

Catalog

2.1 - 2.18 Foundational Math_ Co-ordinate Geometry and Linear Algebra.pdf	1
3.1 - 3.20 Foundational Math_ Calculus for ML and AI.pdf	55
4.1 - 4.12 Python for Data Science_ Introduction.pdf	115
5.1 - 5.6 Inbuilt Data Structures.pdf	152
6.1 - 6.3 Introduction to functions.pdf	191
7.1 - 7.7 Programming Problems 1.pdf	209
8.1 - 8.9 Insertion Sort.pdf	212
9.1 - 9.10 Analysing Algorithms_ Time and Space Complexity.pdf	224
10.1 - 10.9 Problems_ Big O, Theta, Omega Notation .pdf	248
11.1 - 11.29 Recursion and Dynamic Programming.pdf	258
12.1 - 12.9 Introduction to Modules_Libraries for DataScience.pdf	327
13.1 - 13.7 Miscellaneous topics in Python.pdf	381
14.1 - 14.10 Object Oriented Programming for ML_AI.pdf	409
16.1 - 16.7 Introduction to Regular Expressions.pdf	432
17.1 - 17.4 Multiprocessing _ Multithreading.pdf	433
19.1 - 19.32 SQL.pdf	445
20.1 - 20.15 Plotting for Exploratory Data Analysis (EDA).pdf	456
21.1 - 21.19 Probability Theory.pdf	490
22.1 - 22.33 Applied Probability _ Stats.pdf	550
25.1 - 25.10 Dimensionality reduction -- Foundations.pdf	618
26.1 - 26.10 PCA (Principal Component Analysis).pdf	646
27.1 -27.7 (t-SNE) embedding.pdf	666
28.1 - 28.16 Real World Problem_ Predict Rating given product reviews on Amazon.pdf	686
29.1 - 29.32 Classification and Regression Models_ K-Nearest Neighbors.pdf	705
30.1 - 30.9 Performance Measurement of Models.pdf	769
31.1 - 31.21 Classification Algorithms in various situations.pdf	793
32.1 - 32.17 Naive Bayes.pdf	832
33.1 - 33.17 Logistic Regression.pdf	890
34.1 - 34.4 Linear Regression.pdf	935
35.1 - 35.11 Solving Optimization Problems.pdf	942
36.1 - 36.15 Support Vector Machines(SVM).pdf	964
37.1 - 37.15 Decision Trees.pdf	1017
38.1 - 38.17 Ensemble Models.pdf	1066
45.1 - 45.17 Featurization and Feature Engineering.pdf	1093
46.1 - 46.10 Miscellaneous topics.pdf	1118

2.1 Introduction



- In this chapter we will be learning about basics in Linear algebra (lines ,planes,vectors,matrices e.t.c)



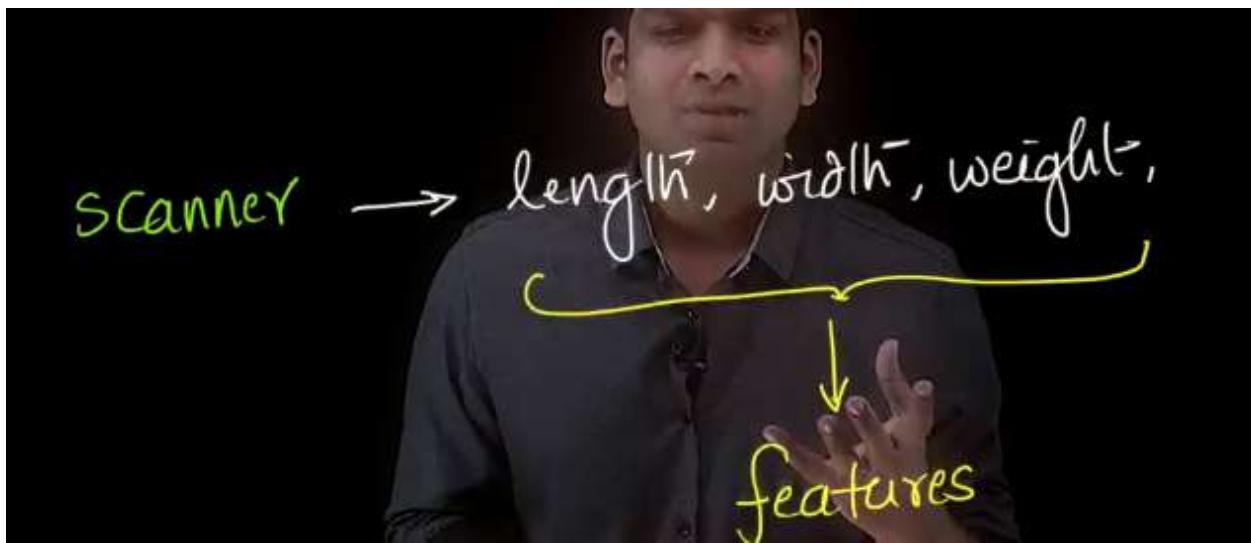
- We will discuss concepts which are relevant to MI/AI and solve them using real world problem context.

2.2 Real world ML classification problem: Sorting Fish in a Factory

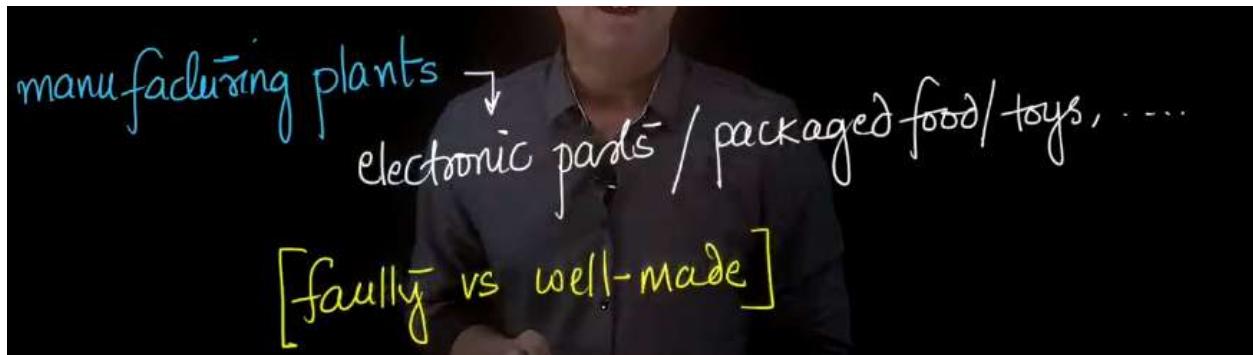
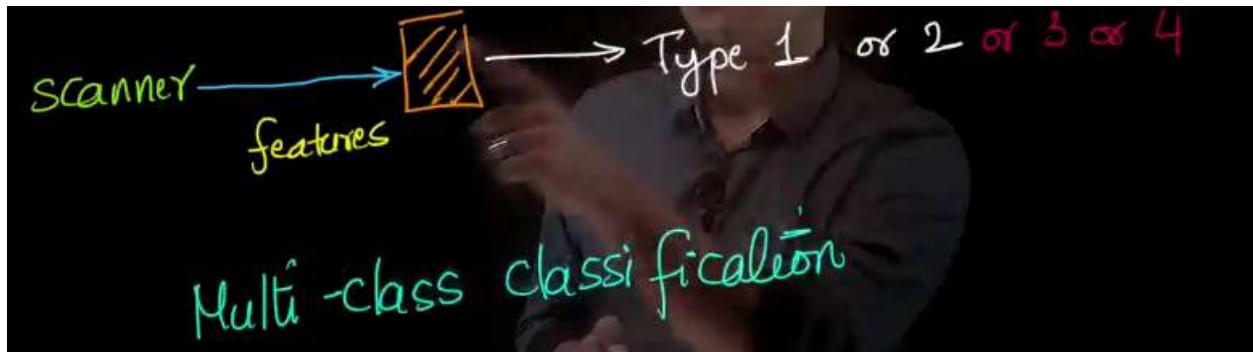
Below is an example shown at timestamp 2.28 in the video



Labour intensive & need for human experts



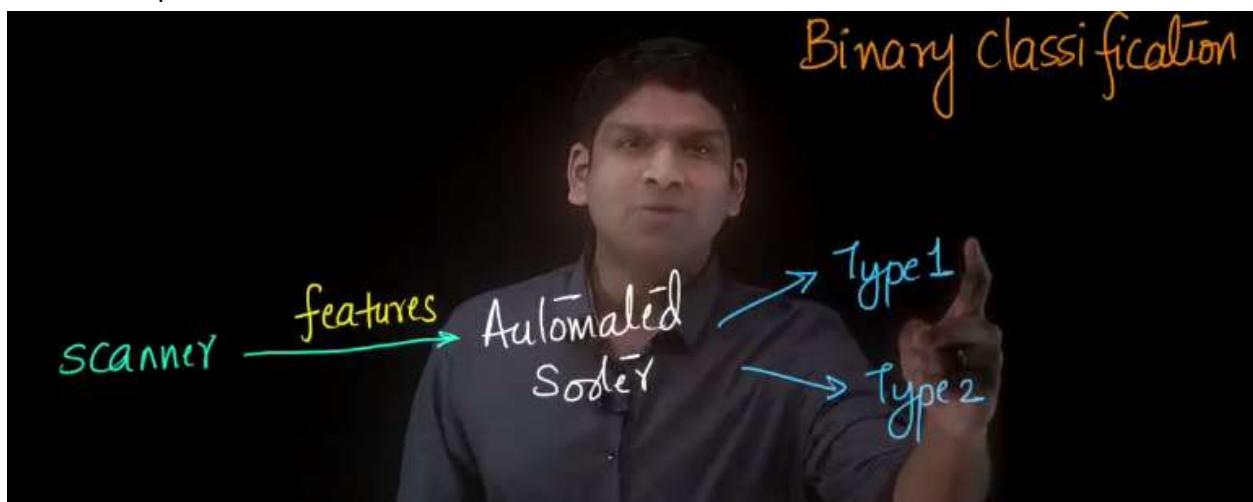
- Sorting fish problem can be treated as classification problem where we have to classify whether a given fish is of type 1 or type2. It is clearly an labour intensive task.
- We try to automate the task by identifying the features of the fish such as length, width, weight and using these features we classify each fish whether it is type 1 or type 2



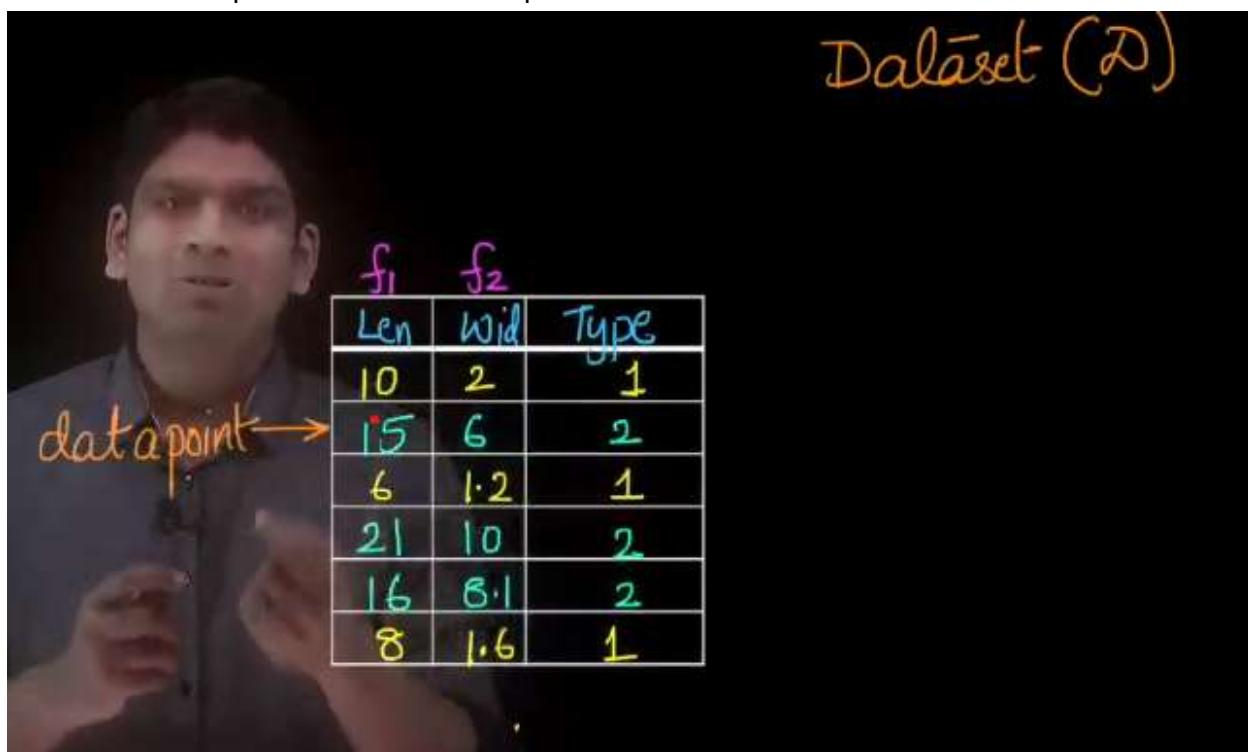
- Our task here is a binary classification task since we have two types of fish but this can be extended to multiclass classification as well.
- Similar classification tasks are performed across the manufacturing plants for identifying faulty vs good products.

2.3.Dataset & Plotting

At timestamp 0.30 in the video



Below is an example shown at timestamp 2.02 in the video

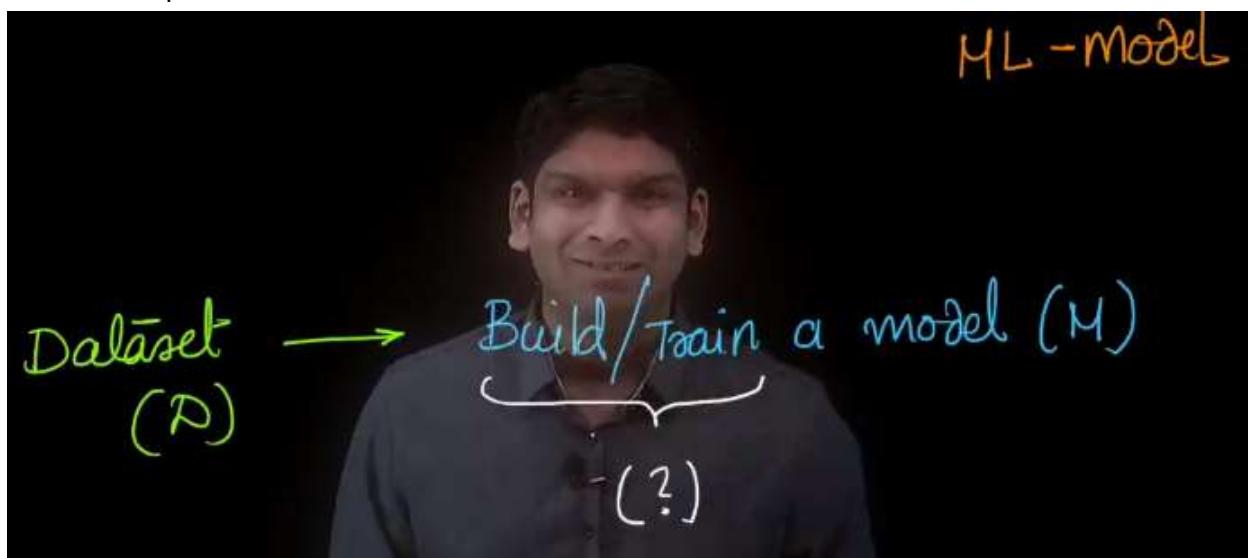


The image shows a man in a grey shirt pointing towards a table on a screen. The table is titled "Dataset (D)" and contains six rows of data. The columns are labeled f_1 , f_2 , Len, Wid, and Type. The data points are colored yellow for Type 1 and green for Type 2.

f_1	f_2	Len	Wid	Type
10	2	1		
15	6	2		
6	1.2	1		
21	10	2		
16	8.1	2		
8	1.6	1		

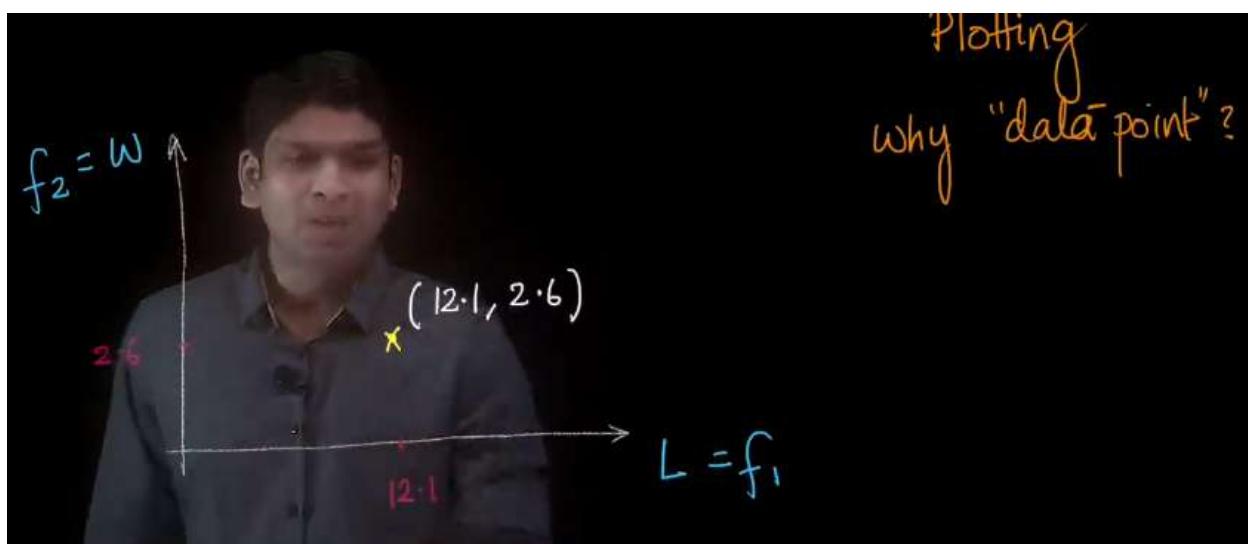
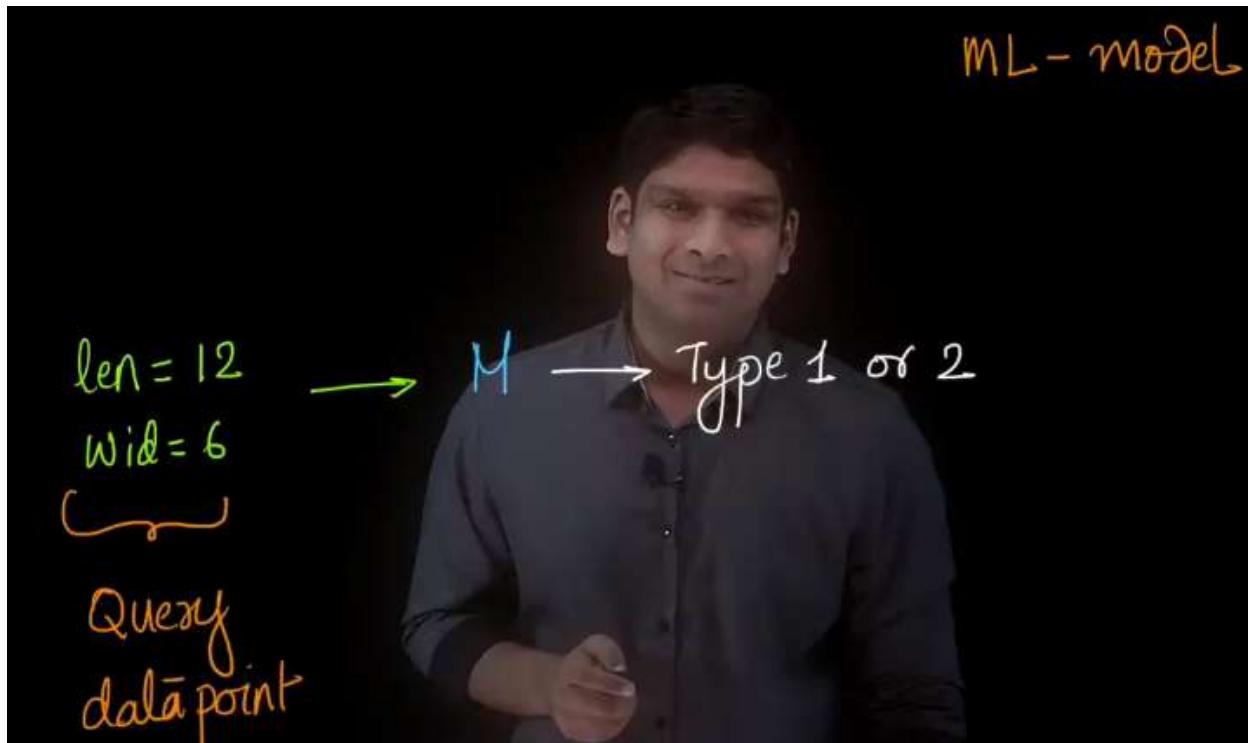
- The problem we are about to solve is a binary classification problem and we use the features of the fish for classification.
- We have a dataset(D) in which each row describes the length,width,type of a fish.We call this information of each fish as a datapoint which is collected manually.
- The yellow data points represent fish of type1 and green data points represent fish of type 2.

At timestamp 3.31 in the video

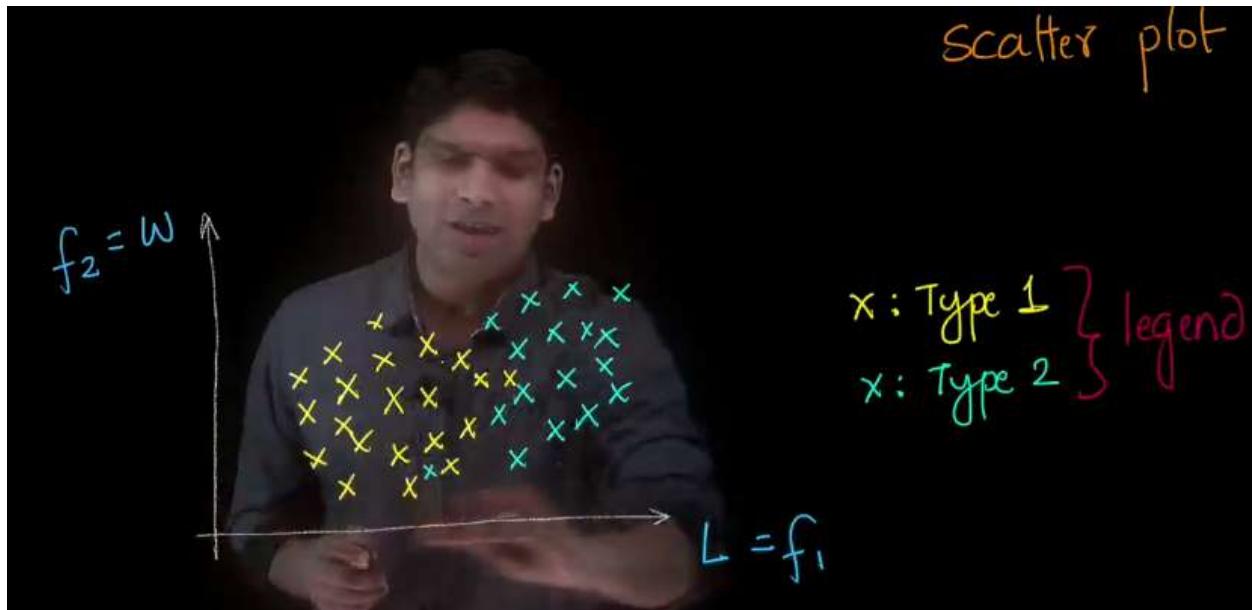


- Given the dataset D we are going to build or train a machine learning model (M).

At timestamp 6.03 in the video



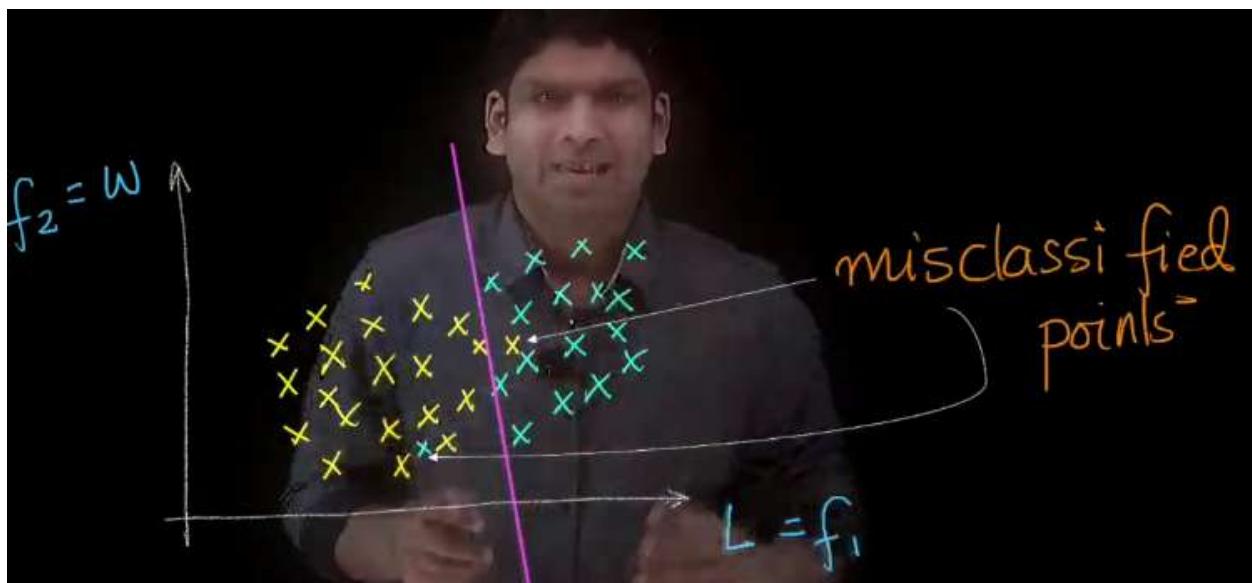
- Assume we have built model M ,consider we have a fish with features length=12,width=6 and we want to find whether it is of type 1 or type2. We call this a query datapoint.we can visualize the data point as shown.
- feature 1(length),feature 2(width) are represented along the X,Y axis respectively.(f1,f2) gives data about 1 fish.



- We can visualize the entire dataset in 2D using a scatter plot since we have 2 features ,and we can add legend to the plot for separating data points.

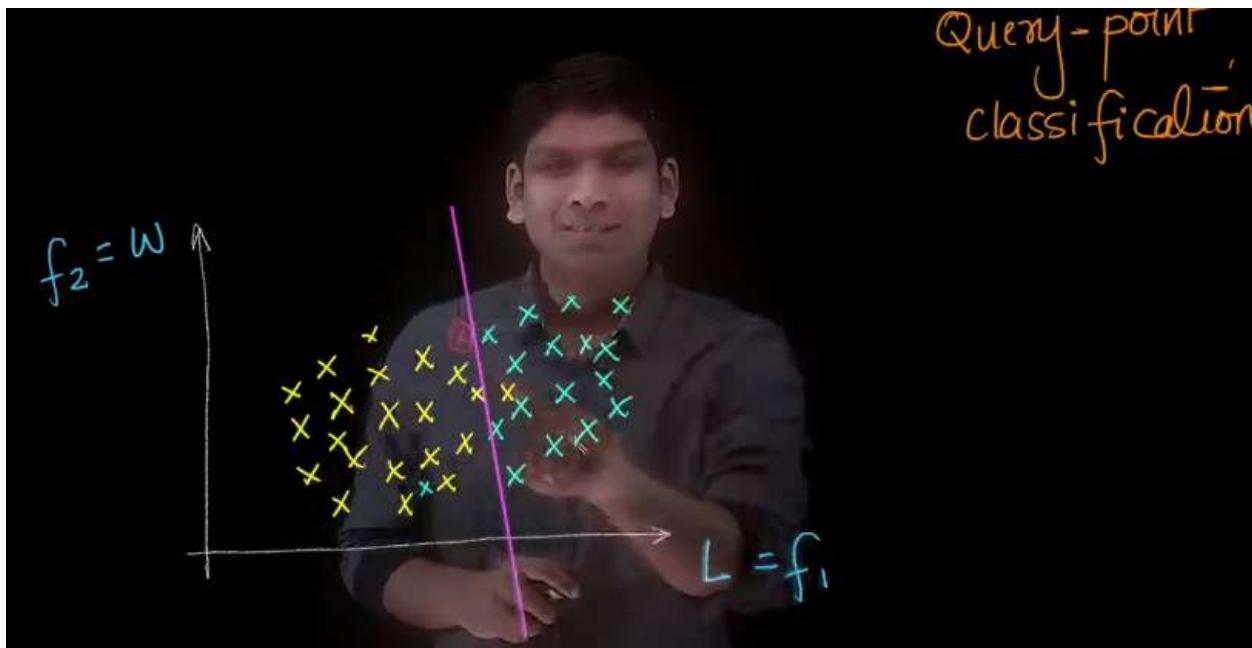
2.4 Lines, Circles and Ellipses for ML Classification

At timestamp 3.18 in video



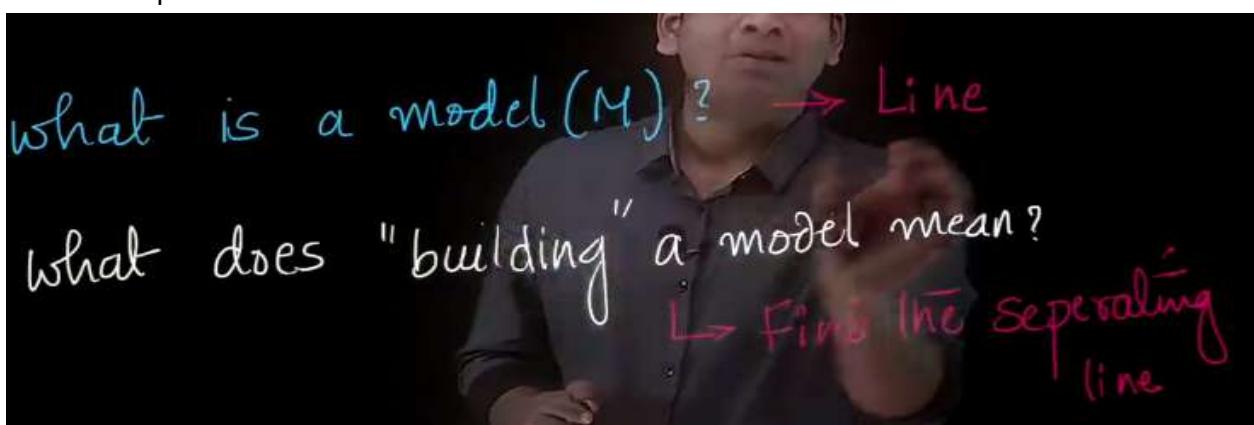
- You can clearly observe that the pink line is more or less separating the two types of fish. Let's say that the line is our model. It is separating our yellow points and green points fairly well except for the two misclassified points. Majority of the points are classified correctly.

At timestamp 6.13

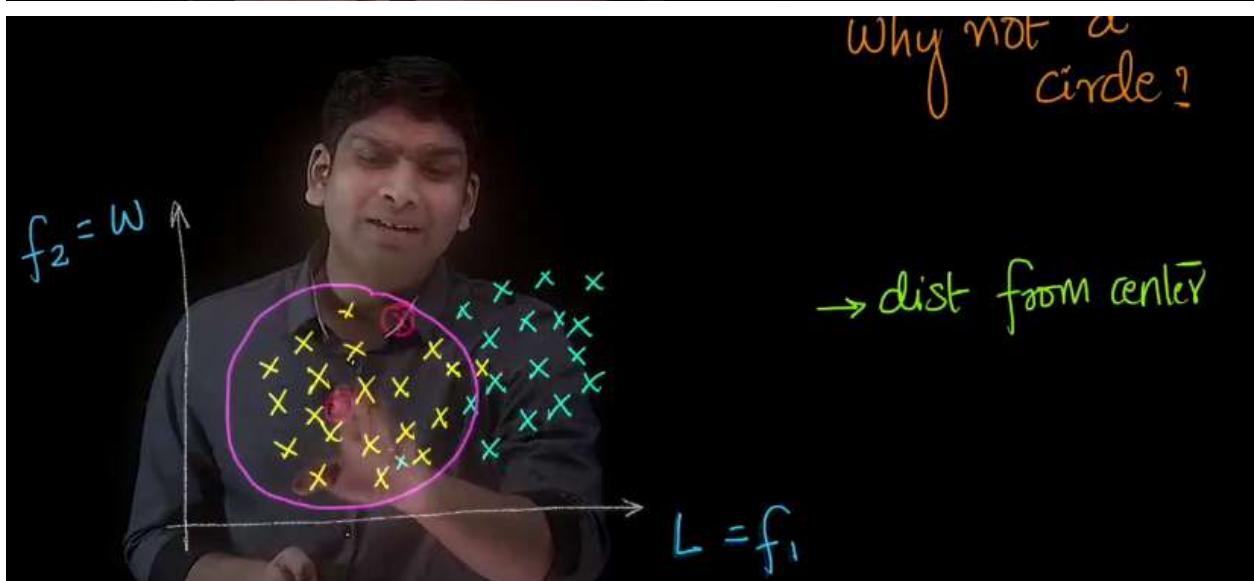
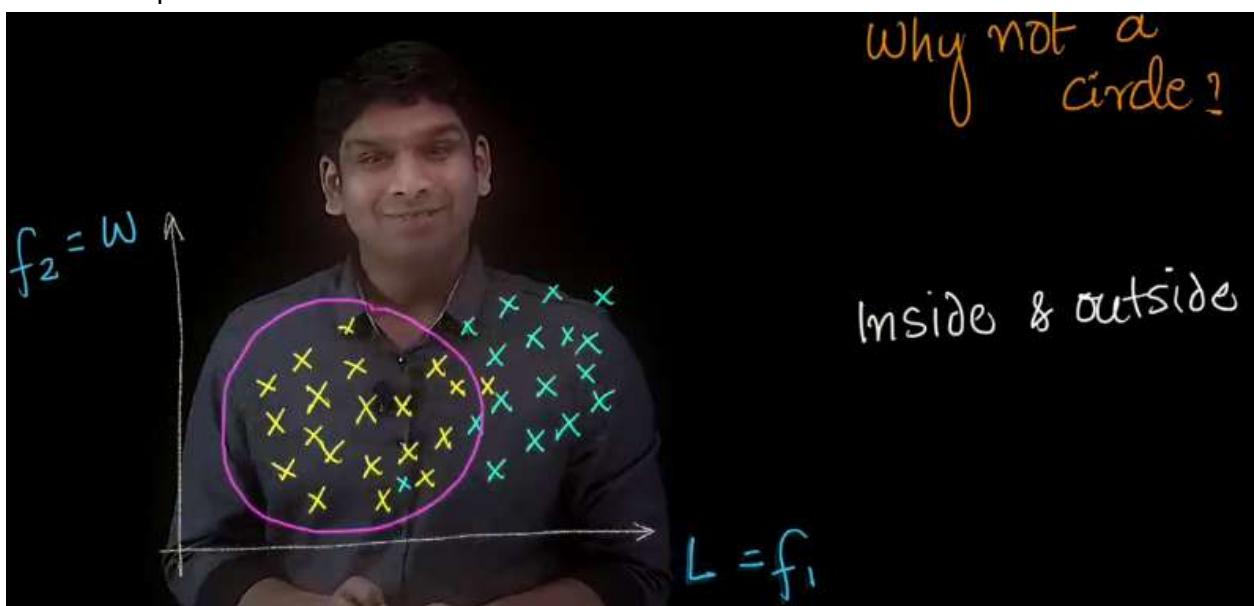


- Now that we have our line as a model ,we can have a query point and determine whether it is type 1 or type 2 fish.The closer the query point towards the line our model is less certain about its type.If the point is farther away from the line then model will be confident about the type of fish.
- Finding the classifier or model is nothing but finding a good line that separates the points.
- We classify a query point based on two things ,
 1. Which side of line point lies
 2. Distance of point from the line

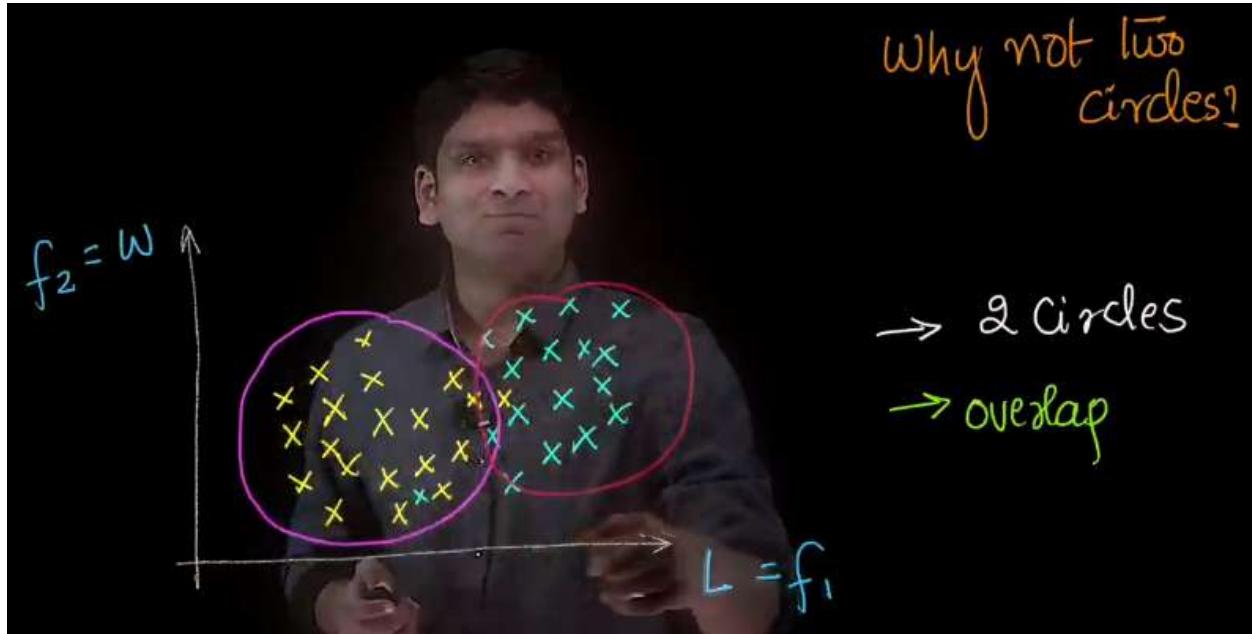
At timestamp 6.13



At timestamp 10.32



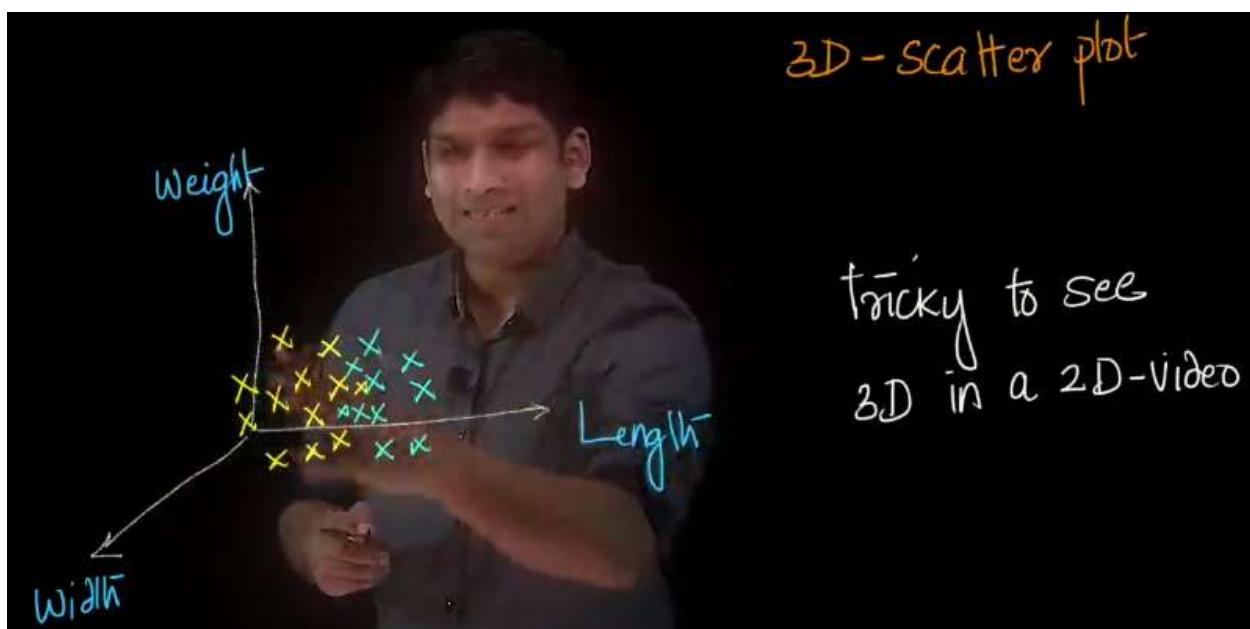
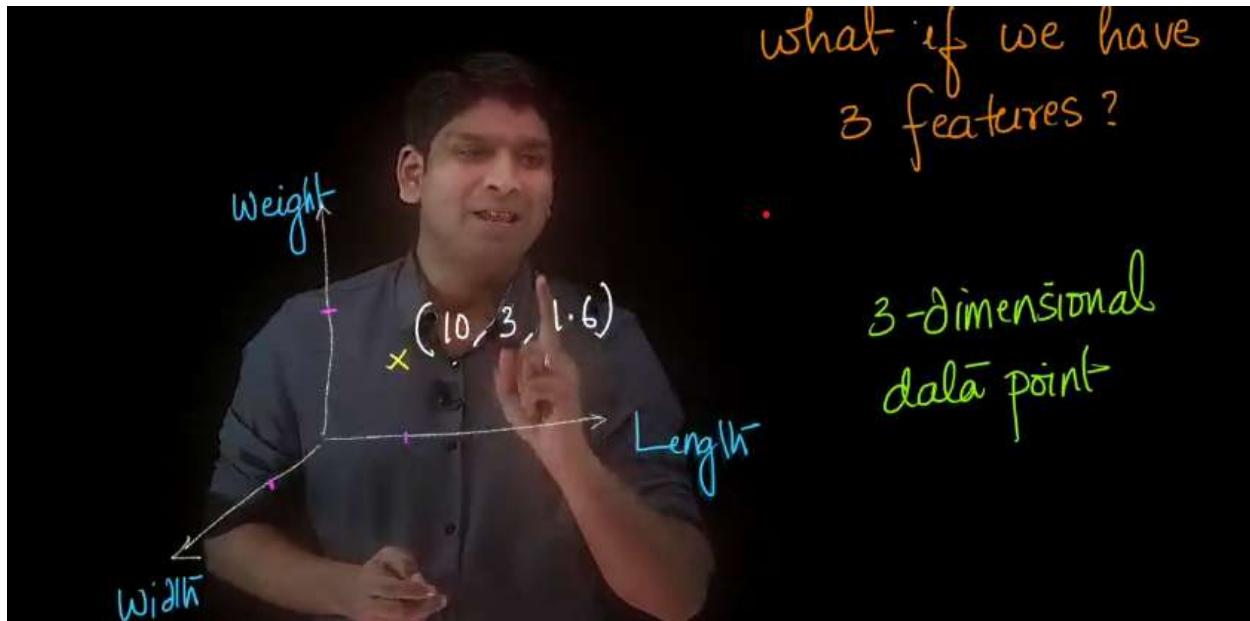
- The model can also be a circle and we classify the data points based on whether the data point lies inside or outside of the circle. We do this by calculating the distance of the point from the centre of the circle.
- The closer we are to the centre the more certain we are about the type of fish vice versa.



- We can also have a model which is represented using two circles
- For every model there are pros and cons
for example
 1. What if a point lies exactly on the line, circle
 2. What if the point lies in the region where circles overlap, and if point lies outside the circles
- Similarly we can use ellipse also for our binary classification problem.

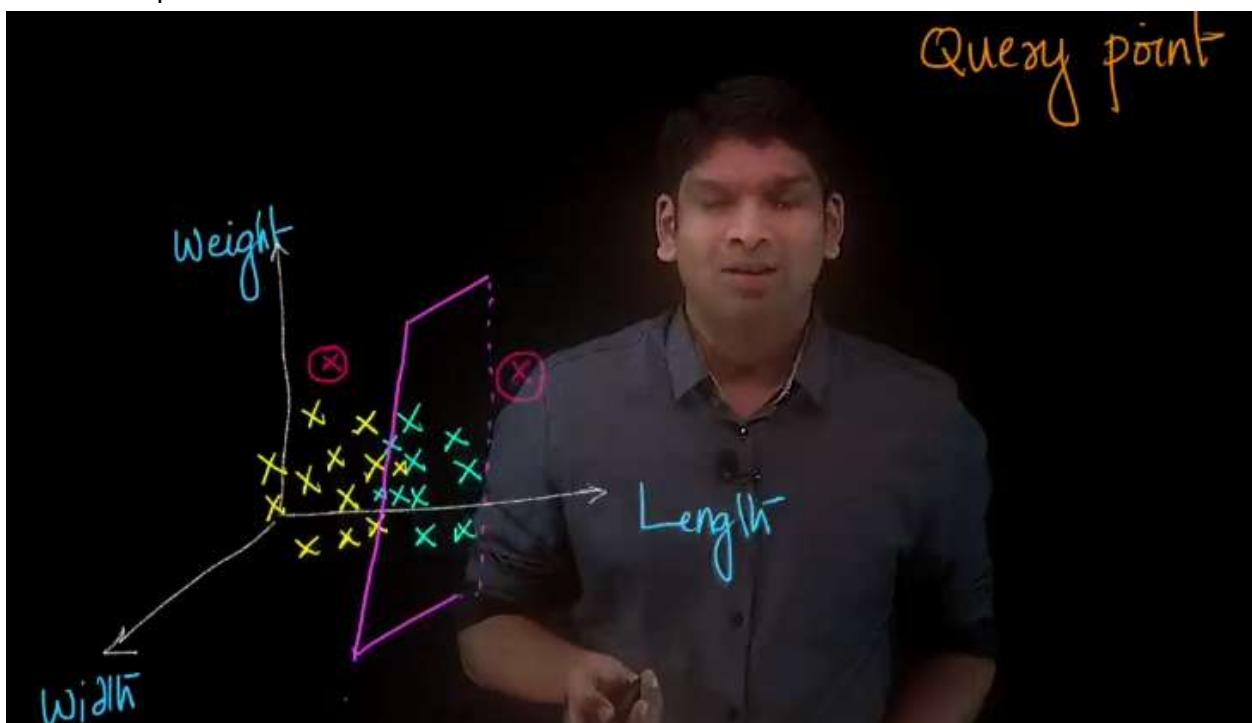
2.5 Planes, Spheres and Ellipsoids for ML classification

At timestamp 1.53 in the video



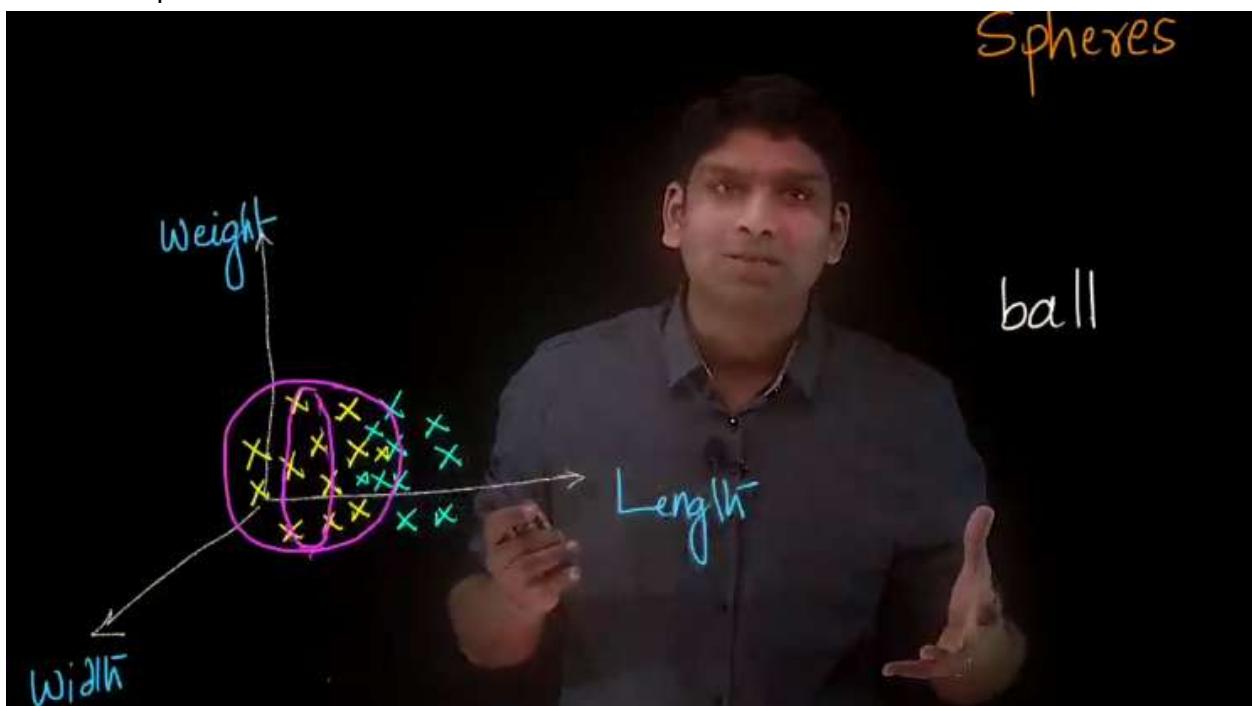
- The same way we visualize a datapoint in 2D we can visualize a data point in 3 D space if we have 3 features to represent a data point along the x,y,z axis respectively.
- We can also visualize the entire dataset in 3D space.

At timestamp 8.20 in video

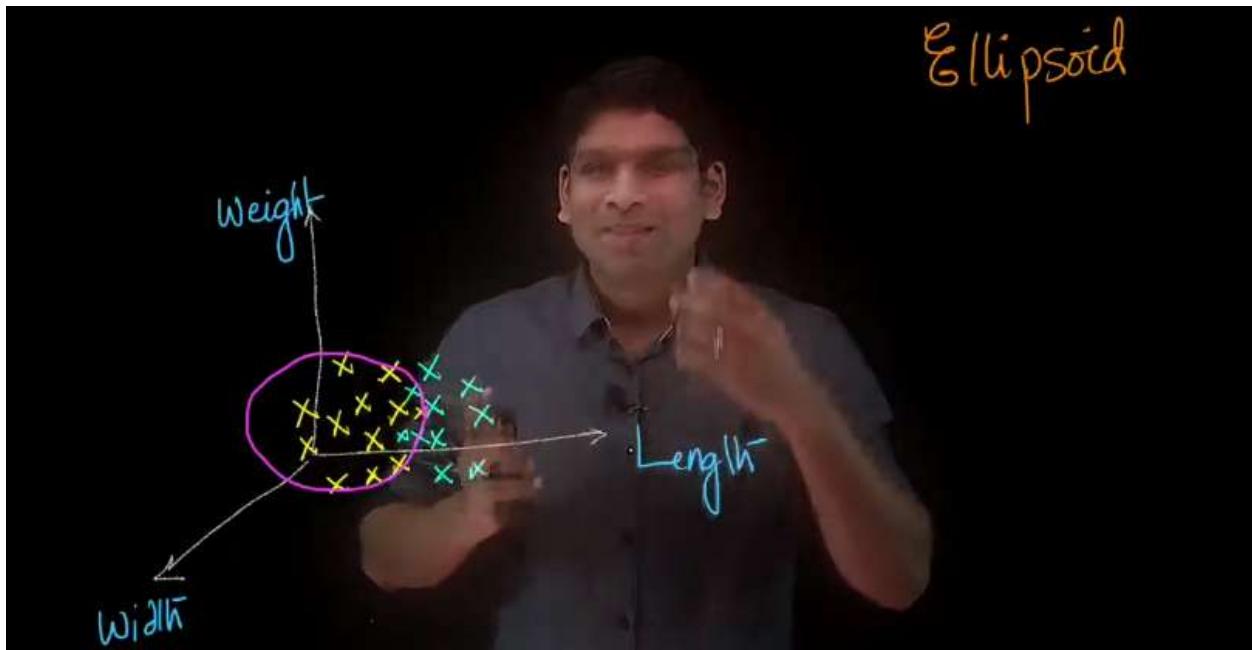


- We cannot use a line to classify the data points in 3D space; we need a plane for separating them. Plane is the simplest model in 3D
- Plane separates the 3D space into two regions ,given a query point we find the distance of the point from the plane and which side of the plane the point belongs .
- This is similar to what we have discussed with line in 2D

At timestamp 9.43 in video



- Just like we have extended the concept line in 2D to plane in 3D ;we can extend circles in 2D to spheres in 3D .Everything inside the sphere is one class of data points and outside will be another class.How close the point is to the centre of the sphere determines how certain we are that the point belongs to a particular class.



- Similarly we can extend ellipse in 2D to ellipsoid in 3D for the classification.

3-D data \Rightarrow use planes/spheres/ellipsoids

At timestamp 13.2 min

what if 4-D,
5-D, 10-D, 100-D?

Let's say we have > 3 features

At timestamp 14.34 min

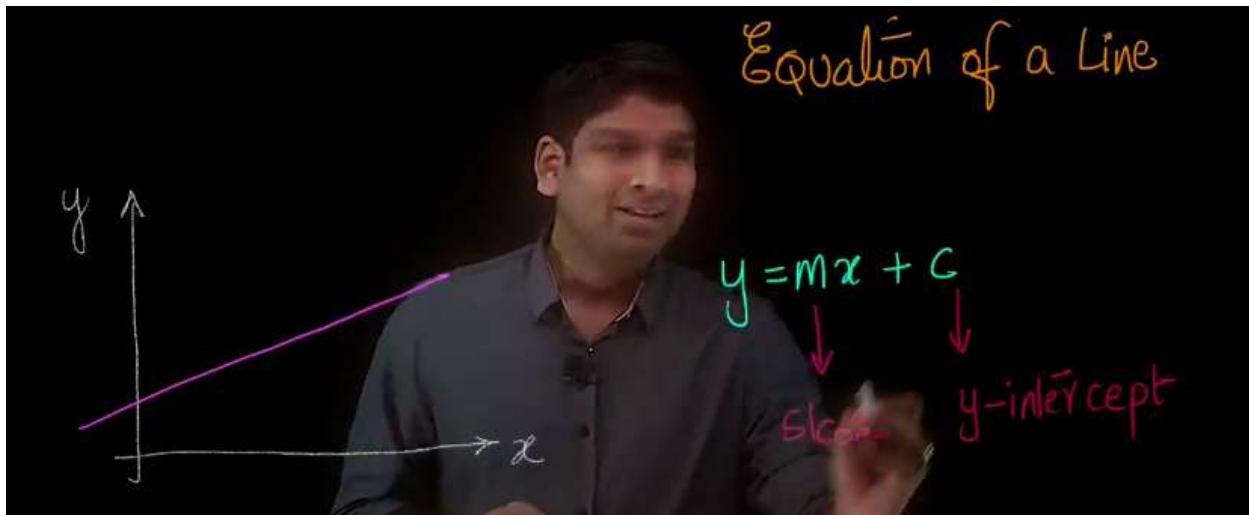
Hyper-X :

hyper-plane, hyper-sphere, hyper-ellipsoid
↓
new concepts [use Linear Algebra]

- To summarize we use planes, spheres, ellipsoids in 3D space and when we have more than 3 features we extend the same concept to higher dimensions as well. But we cannot visualize data in higher dimensions like we did in case of 2D and 3D; so we use linear algebra to understand it mathematically. We call them hyper planes, hyper-spheres and hyper ellipsoids in higher dimensions.

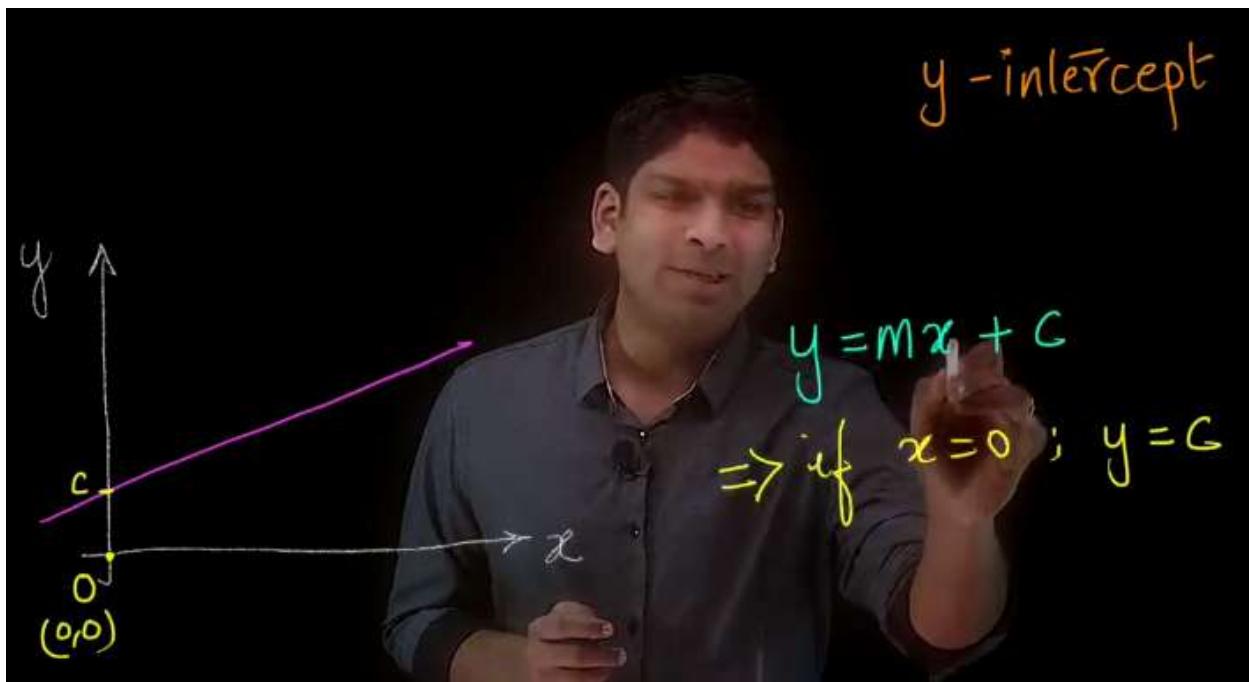
2.6 Equation of a line

At timestamp 0.22



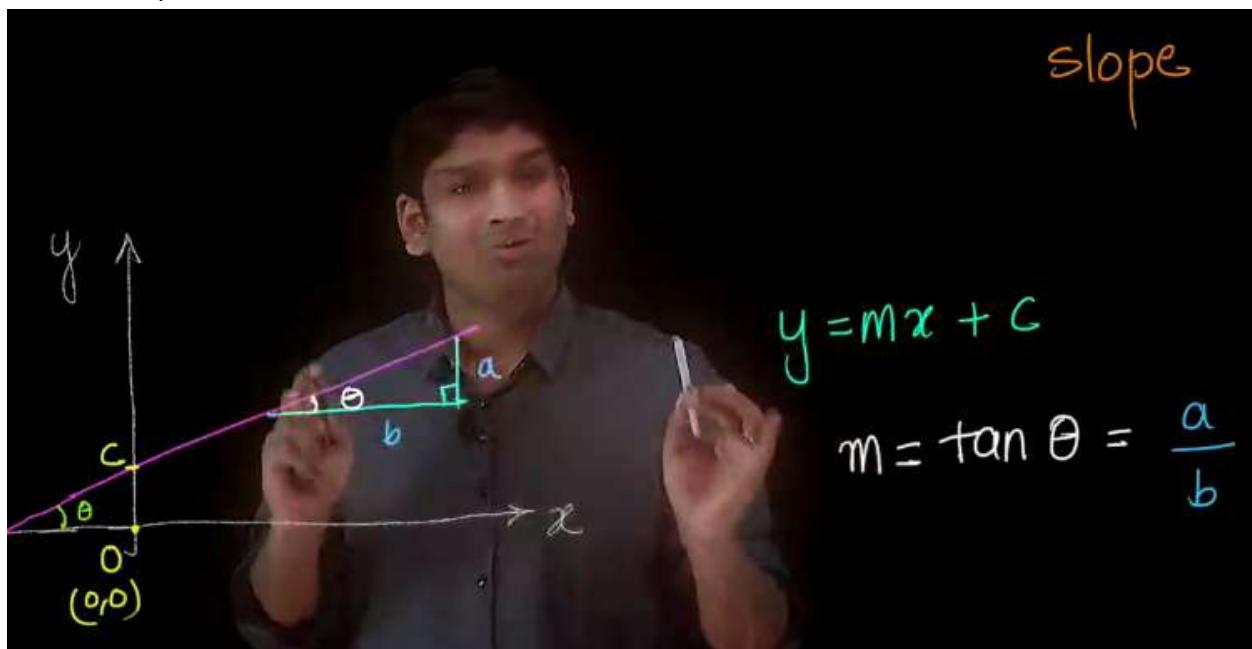
- Slope intercept formulation of a line is $y = m \cdot x + c$, whereas m is the slope and c is y-intercept.
- y intercept is the point at which the line intercepts the y axis.

At timestamp 2.32



- For the line shown above the line is c units away from the origin ,hence for this line $y\text{-intercept}=c$.Algebraically at $x=0$ we have $y=c$

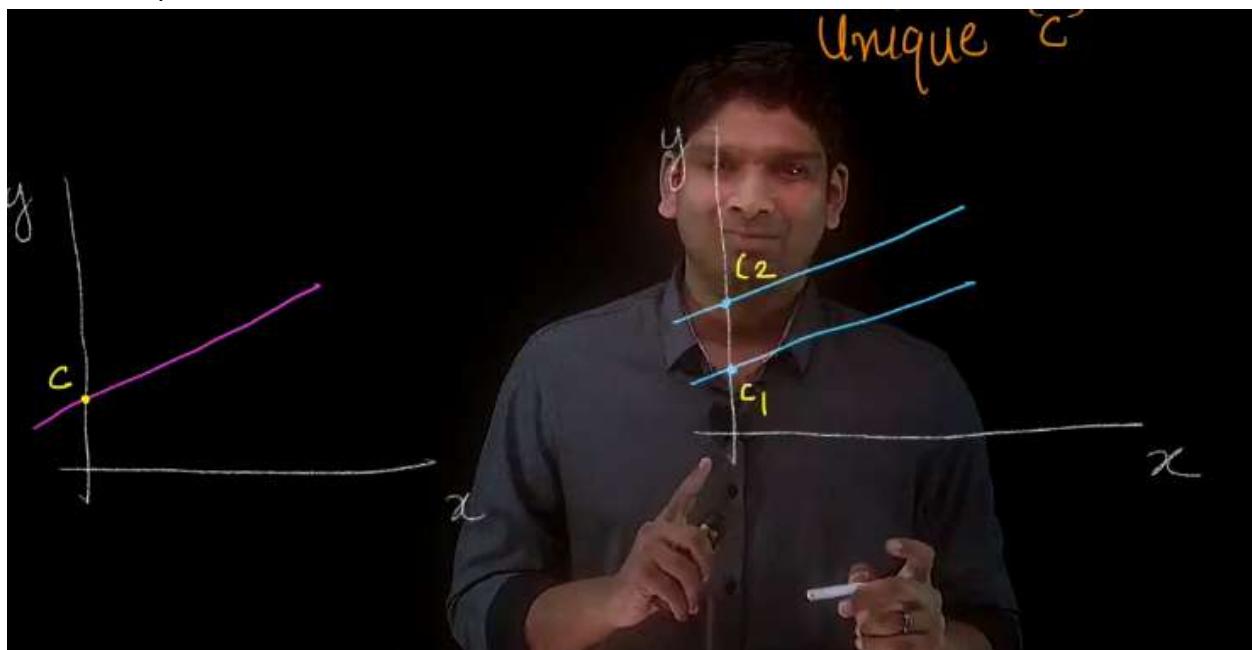
At timestamp 4.5 min



- Slope m is the angle made by the line with x -axis, you can calculate slope $\tan\theta = a/b$ as shown above.

→ Slope is same everywhere on the line

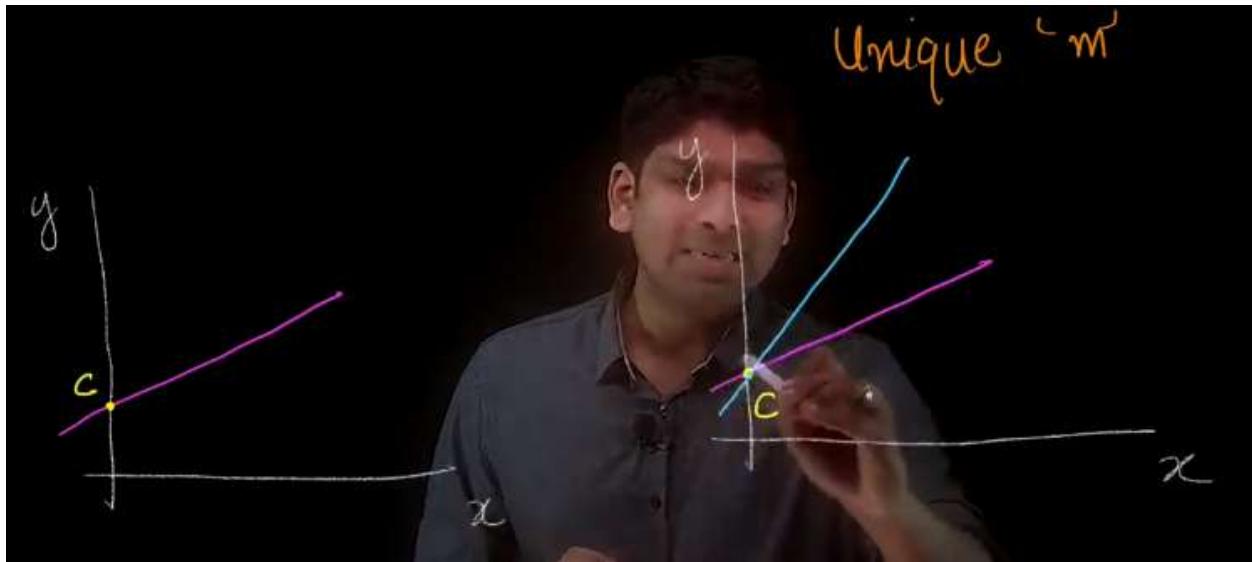
At timestamp 7.02



- Intercept c will always be unique for a given line.

- If we have two lines parallel to each other their slopes will be same but intercept will not be same
- Given unique m and c there will only be one such line .

At timestamp 8.29



- The intercept c can be the same for two lines but their slopes will be different;because the angle made by each line with x axis will be different.

\rightarrow splines = curved lines . Not often used in ML.

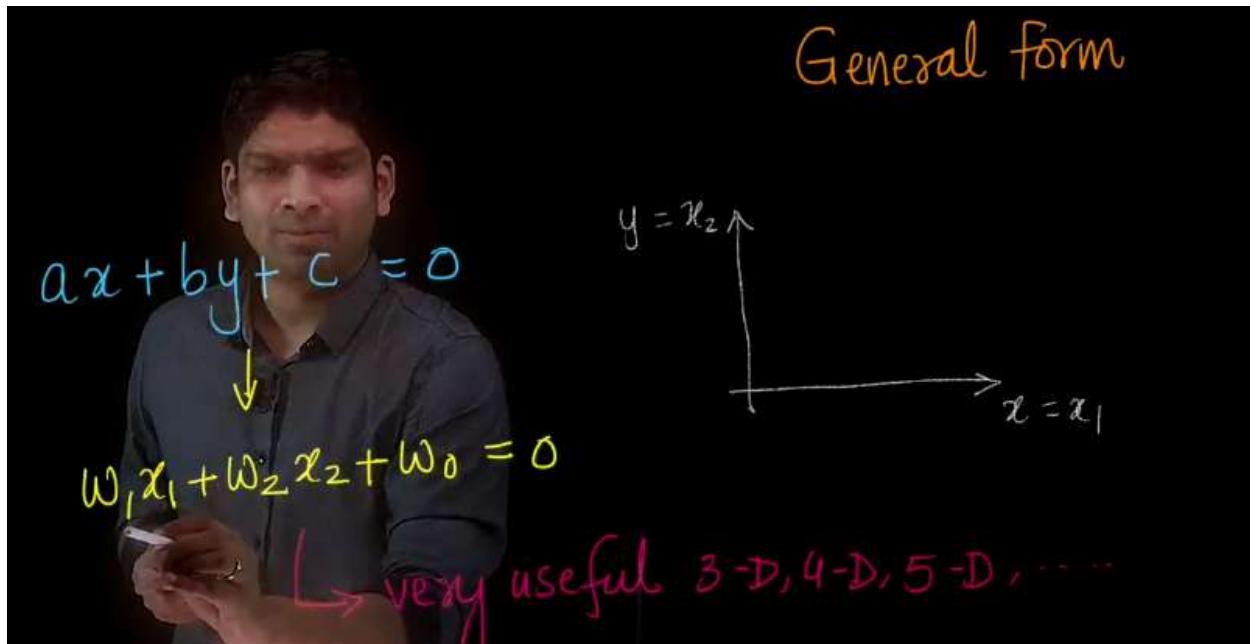
At timestamp 12.31

The image shows a man in a dark shirt standing in front of a whiteboard. He is writing mathematical equations. At the top right of the whiteboard, there is handwritten text: "General form" above "to" and "Slope-intercept" below it. On the left side of the whiteboard, there is handwritten text: "ax + by + c = D". Below this, there are two equations written in white chalk:

$$\Rightarrow y = \frac{-c - ax}{b}$$
$$\Rightarrow y = \frac{-c}{b} - \frac{a}{b}x$$

- General form of a line is given by the equation $a.x+b.y+c=0$. (x,y correspond to x and y coordinates). This equation is important because we use this to represent lines, planes, hyperplanes in ML.
- Given a general form of line we can easily convert it into slope-intercept form as shown above. Slope $m=-a/b$ and y -intercept is $-c/b$.

At timestamp 16.13



- Instead of using x,y as x coordinate and y coordinate from now on we use x1,x2 as the coordinates ;so that we can extend the same concept even if we have n dimensions (x1,x2,x3,x4....).
- Also for representing coefficients we are going to use w1,w2,w0 instead of a,b,c
- For understanding the geometry of line,plane and hyperplane in higher dimensions we are going to represent the equation as $w_1x_1+w_2x_2+w_0=0$

2.7 Equation of a plane & hyper planes

At timestamp 2.10

Equation of plane

$$\Pi : ax + by + cz + d = 0 \rightarrow 3D$$
$$L : ax + by + c = 0 \rightarrow 2D$$

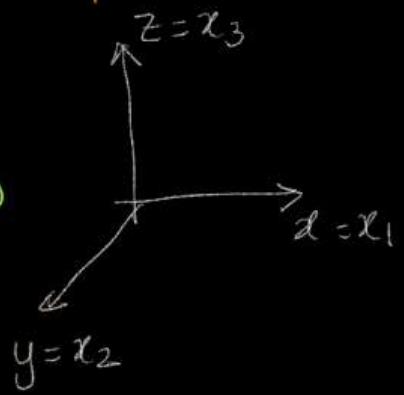
Very similar general form

- Equation of the plane is $ax+by+cz+d=0$.
- We can clearly notice that the general form of line in 2D and plane in 3D are similar, just like line separates two regions in 2D, a plane separates two regions in 3D. In 2D we have only 2 axes and in 3D we have 3 axes.

At timestamp 3.1

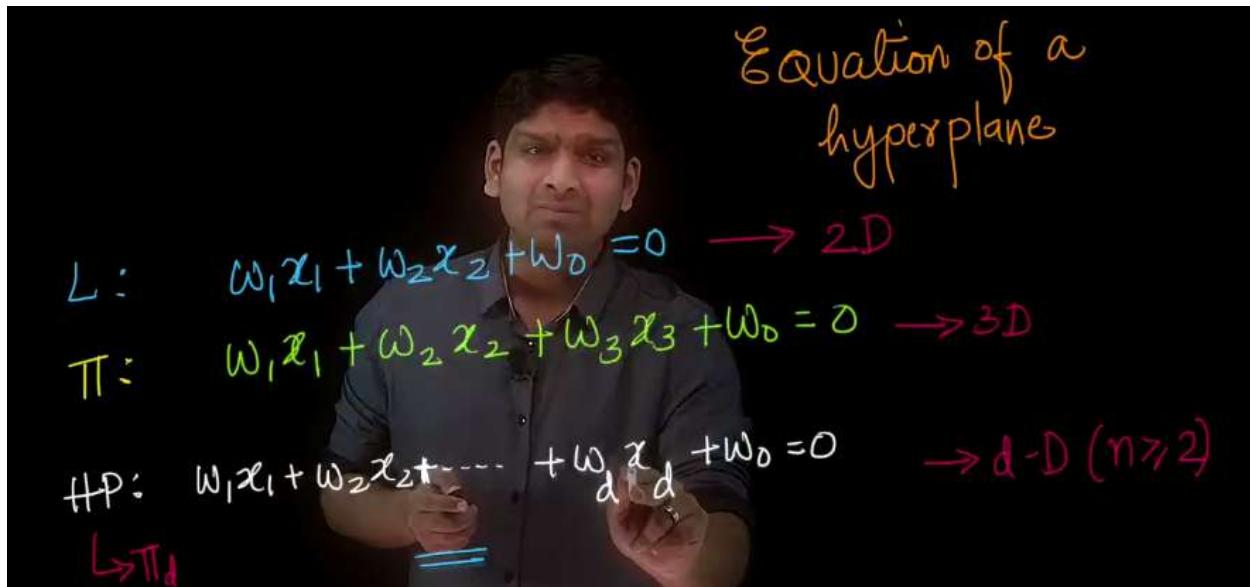
Equation of plane

$$\Pi : ax + by + cz + d = 0$$
$$\Pi : w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0$$



- From now on we use the representation as shown above for representing a plane.

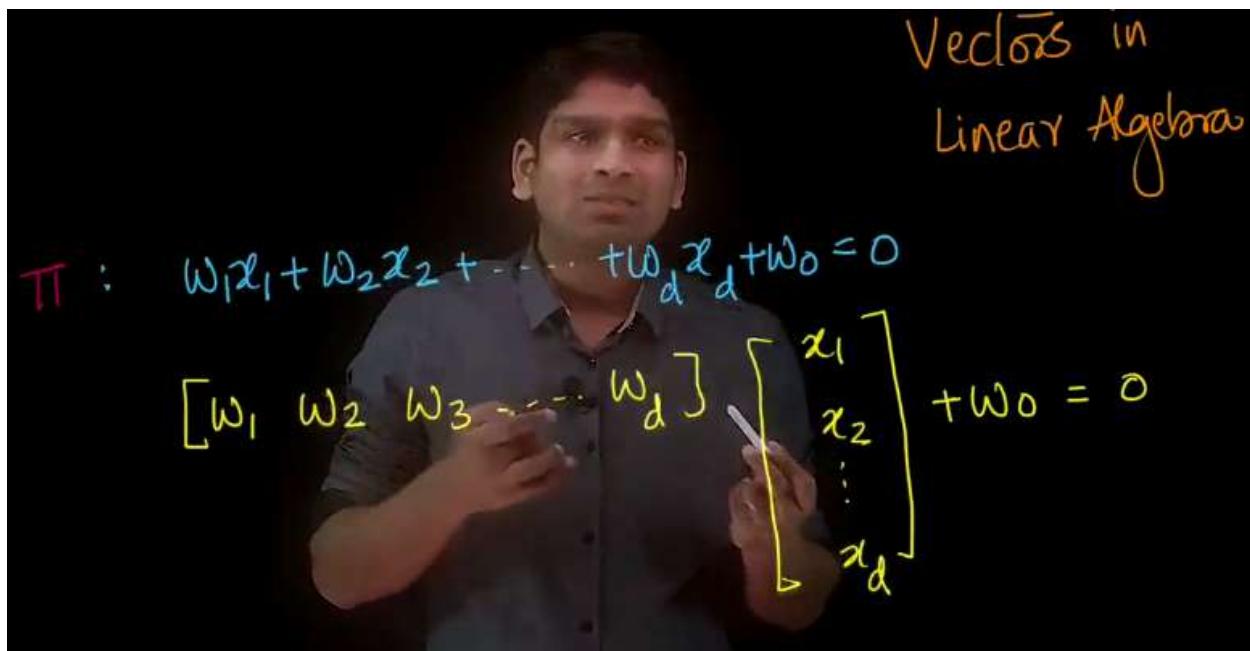
At timestamp 4.26



- Like we have lines in 2D, planes in 3D, we can have d dimensional hyperplanes in dD.
- A hyperplane in d-dimensions separates the d dimensional space into two regions.
- The equation for representing dD hyperplane is shown above.

2.8 Equations using Linear Algebra

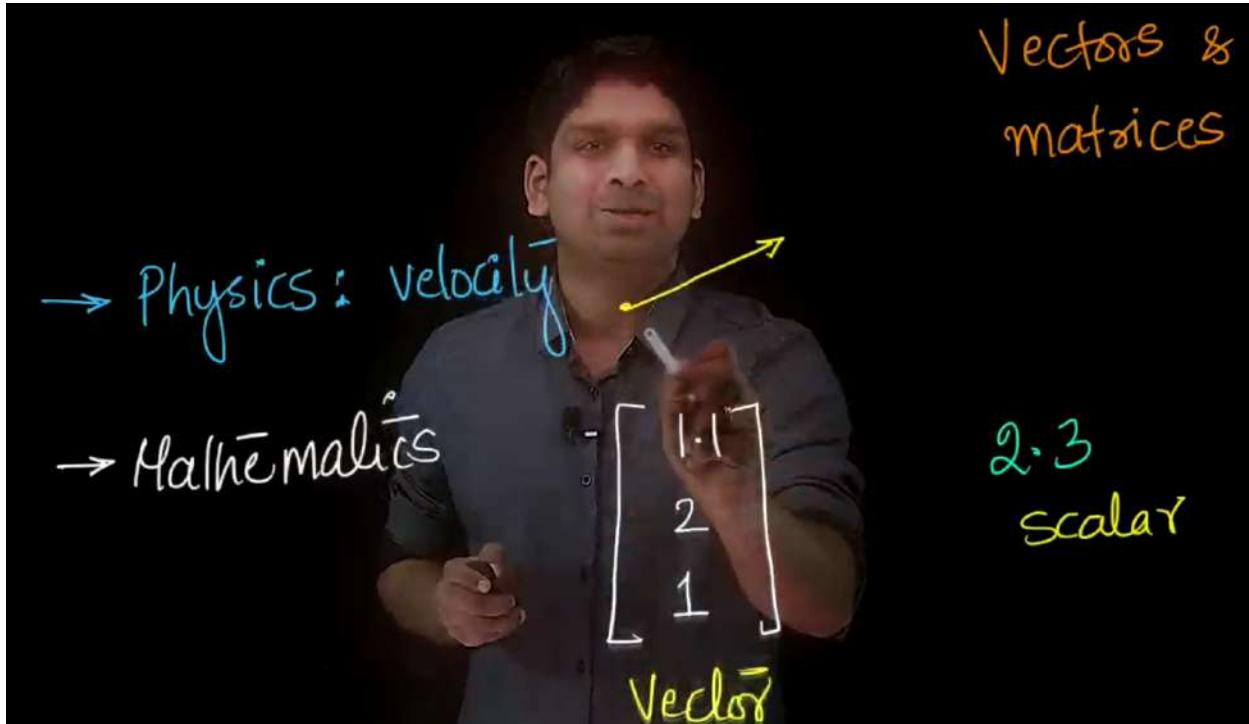
At timestamp 3.10



- We are familiar with the concept of vectors in linear algebra, we can write the equation of a plane in a crisp way using vectors as shown above.

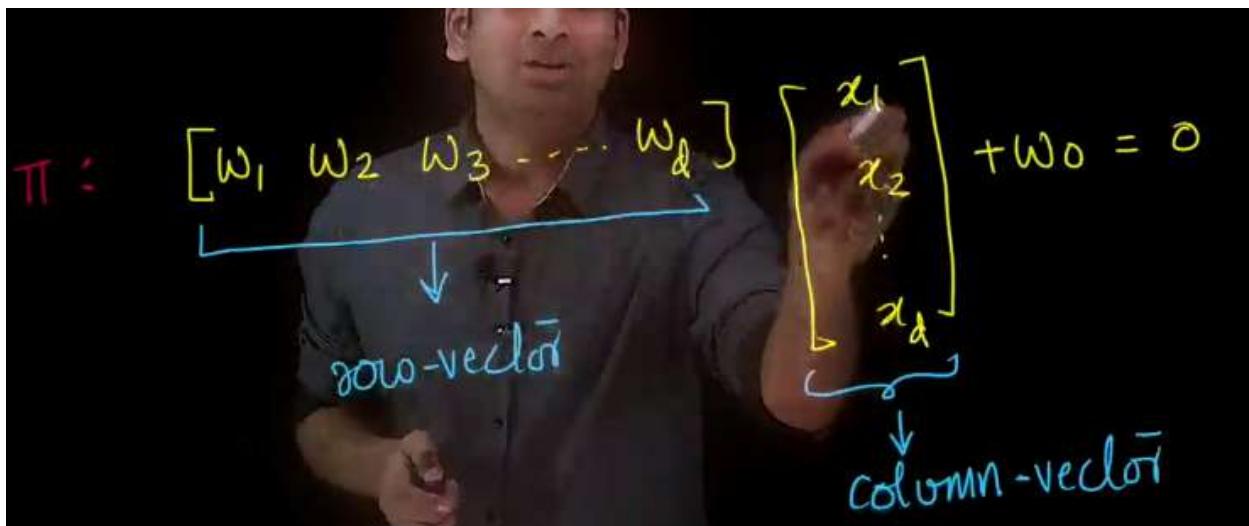
- W is called a row vector and X is called a column vector ,both vectors are of d dimensions. When we perform matrix multiplication for these two vectors we obtain the equation of the plane .

At timestamp 5.48



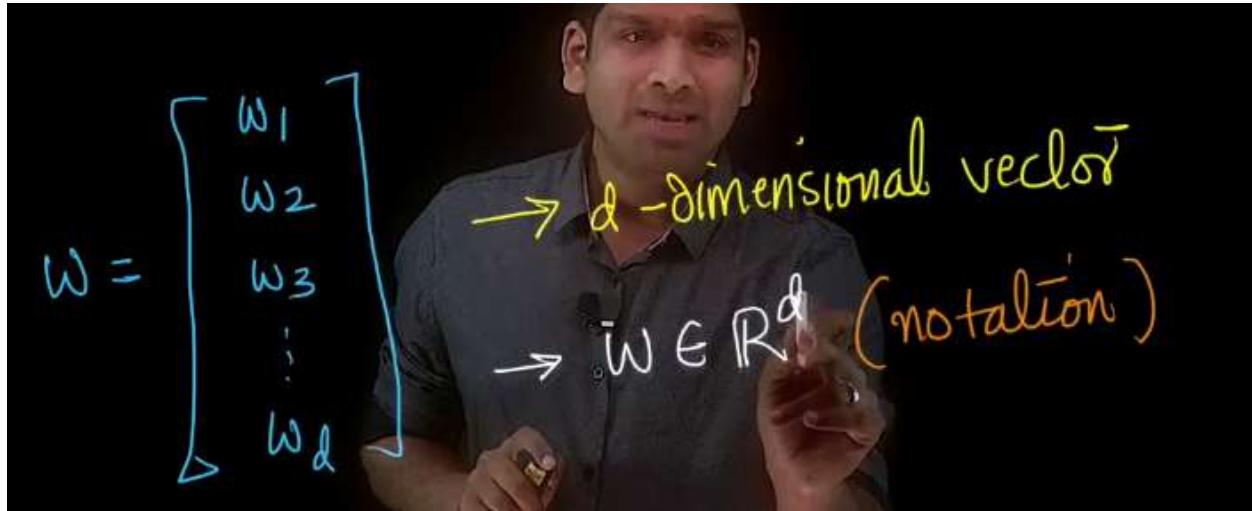
- In physics a vector has magnitude and direction, the concept of vectors arose both in physics and maths. Instead of using the physics approach of vectors we use mathematics interpretation vectors.
- In mathematics a vector and a scalar is represented as shown above.

At timestamp 6.58



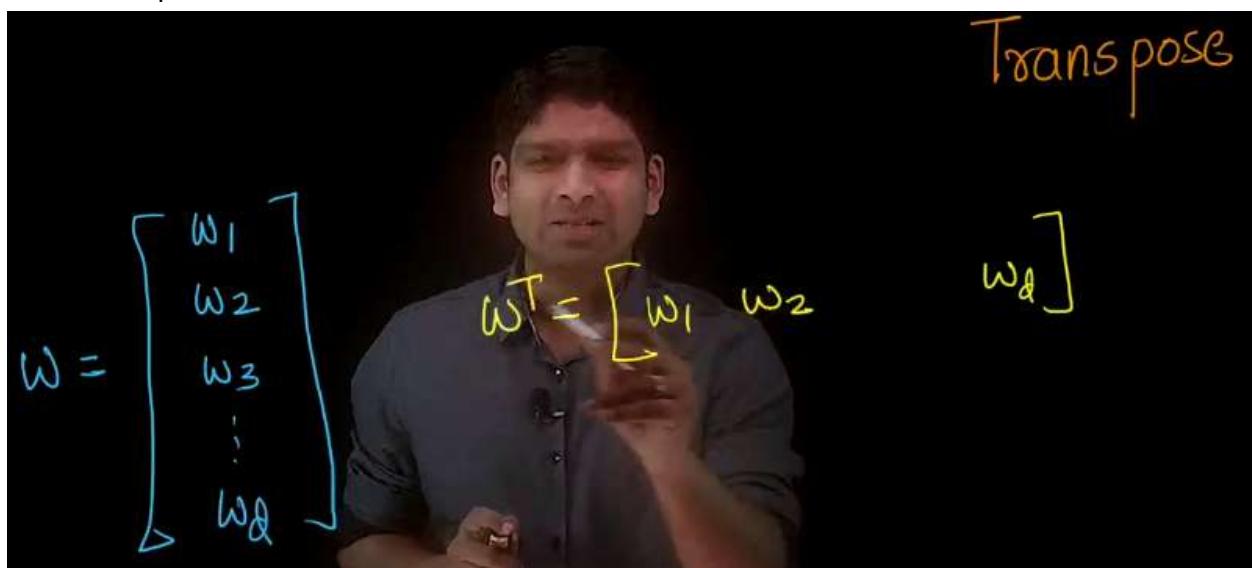
- A row vector and a column vector are represented as shown above, when you multiply a row vector with a column vector the resultant will be a scalar.
- When we say a vector in maths by default it is a column vector. (it is a convention)

At timestamp 9.28



- W is a d dimensional vector and we represent it as shown above. (R is set of real numbers)

At timestamp 10.15



- By default every vector is a column vector, we convert a column vector into a row vector by taking the transpose as shown above. Transpose is an operation we perform on vectors to convert column vectors to row vectors and vice versa.

At timestamp 11.15

Transpose of a matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$
$$A^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

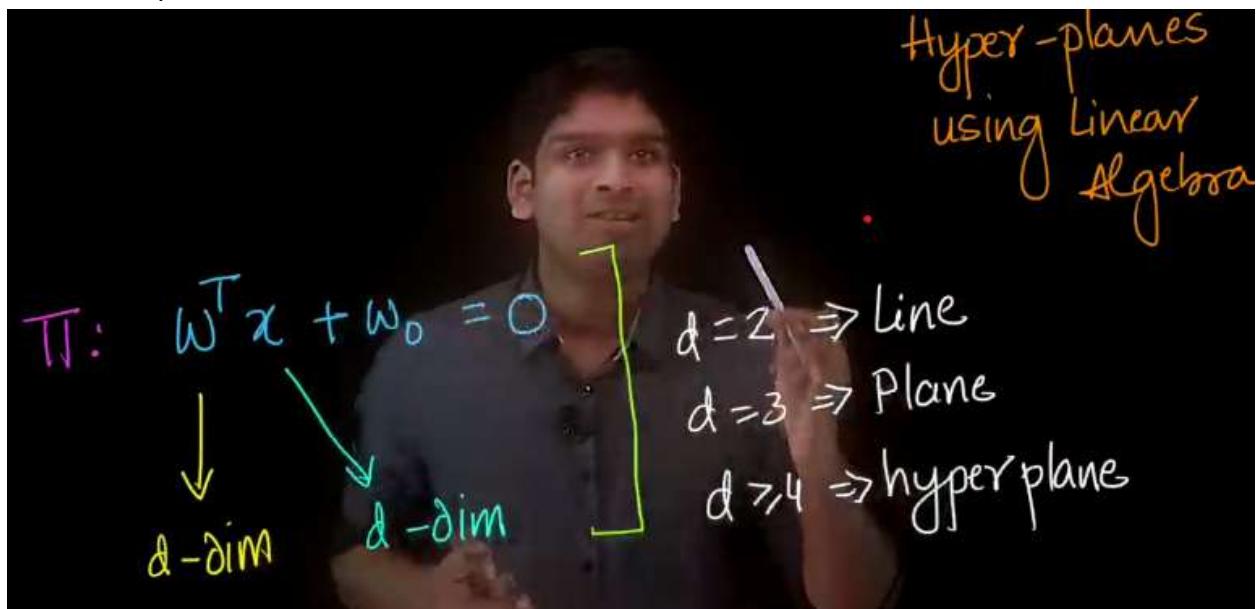
- Transpose can be used in matrices as well as shown above .Rows will be interchanged to columns and vice versa.

$$\Pi: w^T x + w_0 = 0$$
$$\Pi: w^T x + b = 0$$

NOTE: w & x are same dimensional vectors

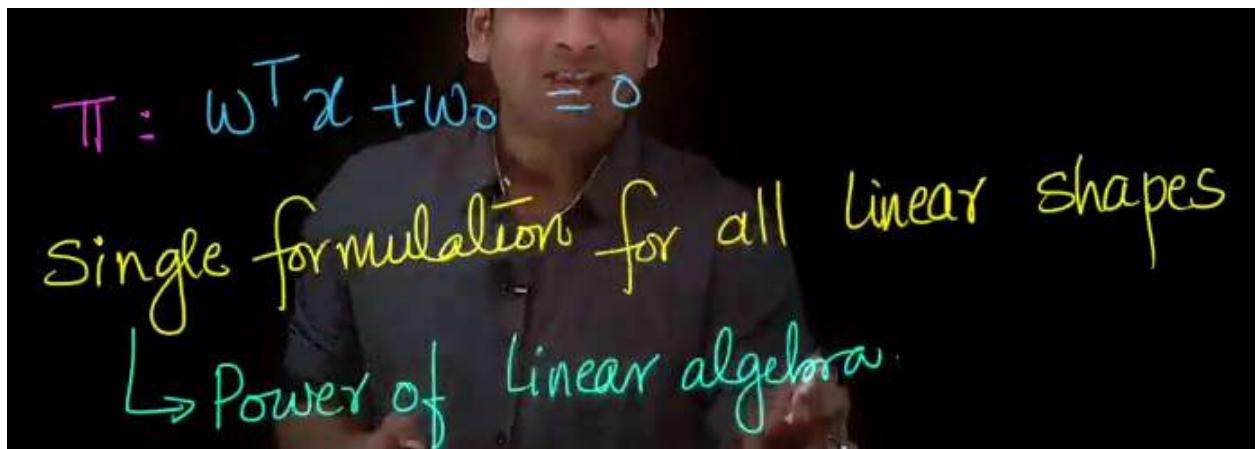
- So now we can represent the equation of the plane as shown above.
- Note that w , x should be of the same dimensions.

At timestamp 15.49



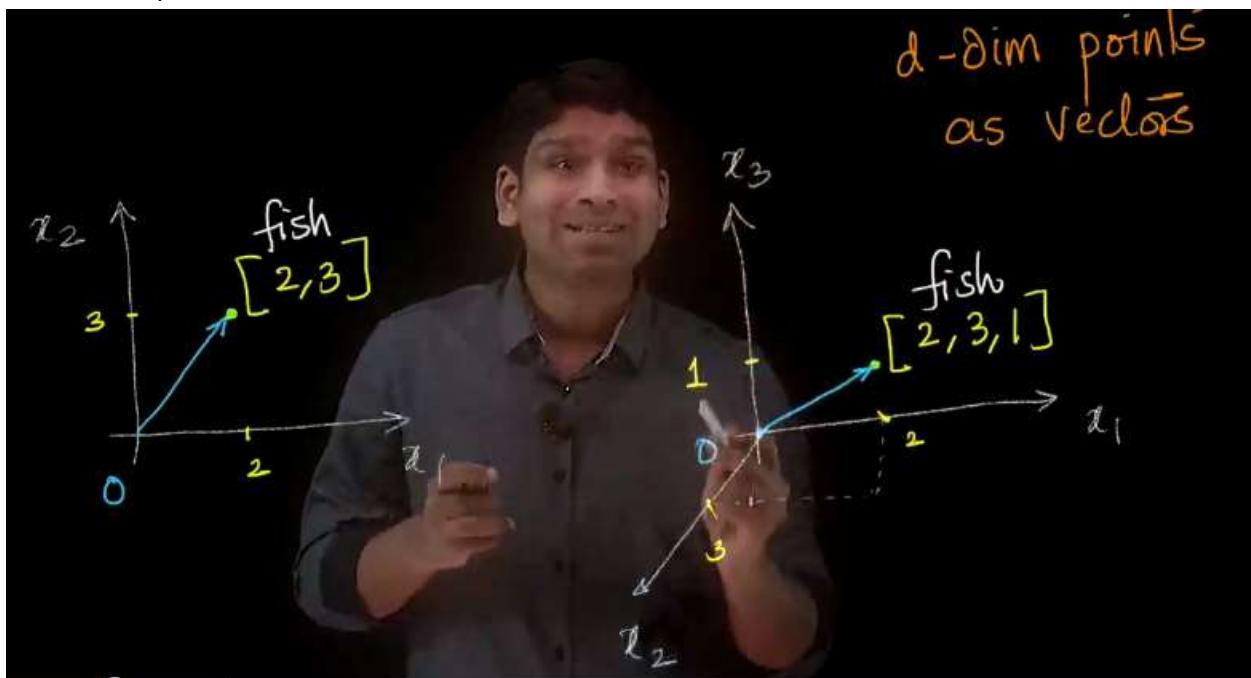
- The above equation when we have 2 dimensions we get a line, in 3 dimensions we get a plane and if $d \geq 4$ we get a hyperplane because w, x are d dimensional vectors.
- It is the most elegant and widely used equation in Machine learning.

At timestamp 16.7



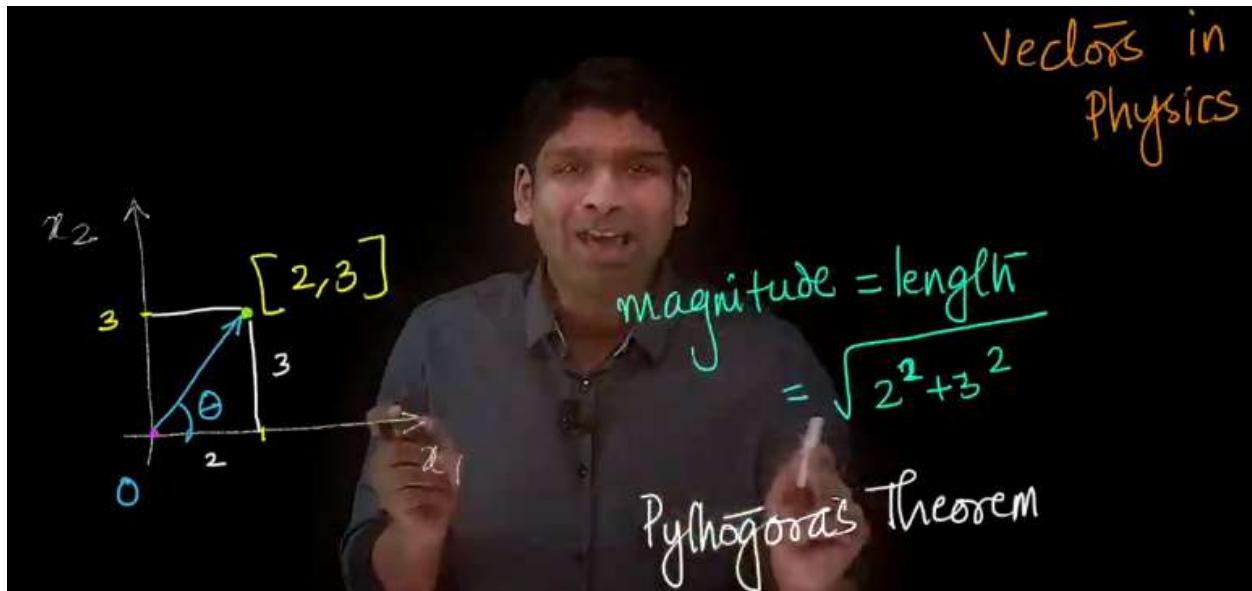
2.9 Dataset using vectors

At timestamp 2.11

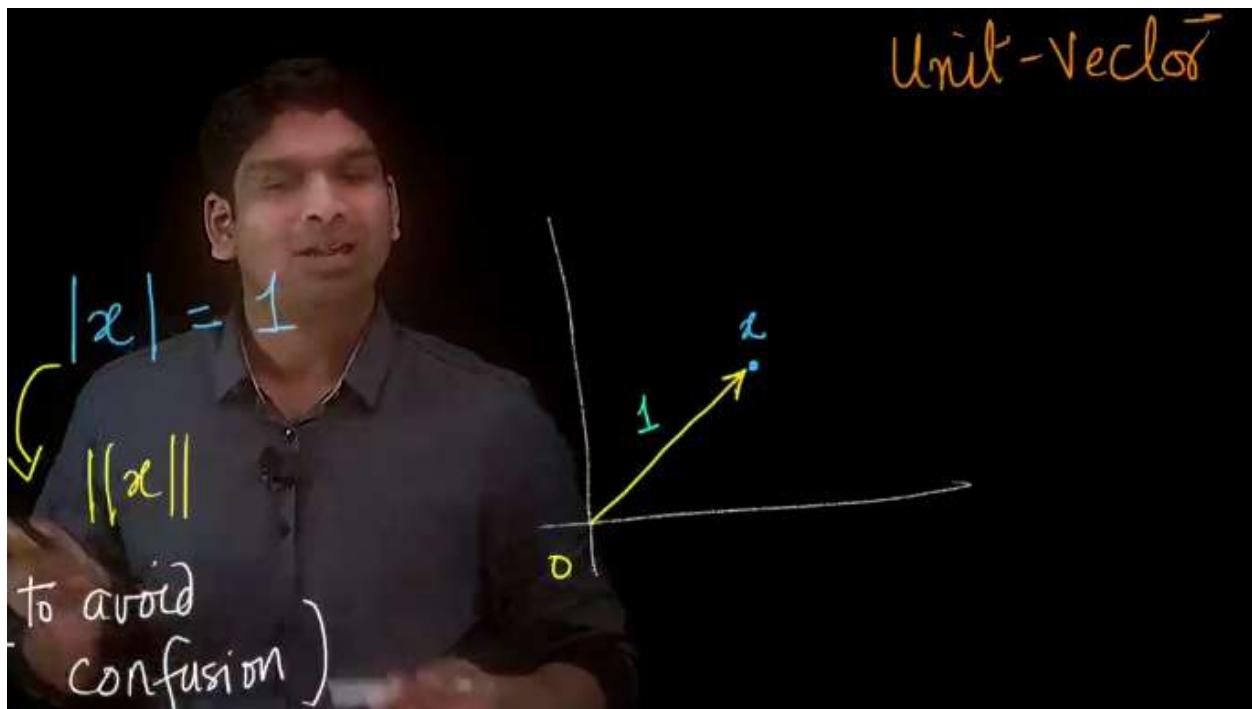


- We can use vectors to represent our dataset. Suppose we have two features to represent a fish(length,width) . Then we can represent a fish using the two values as a vector(fish is represented a point).We can think of d dimensional points as vectors.similarly we can represent a fish with 3 features.
- The first feature is 2 units away from x_1 axis,3 units away from x_2 axis and 1 unit away from x_3 axis.so we can represent a point in 3 D space and that point itself can be represented using a vector from origin to that point.

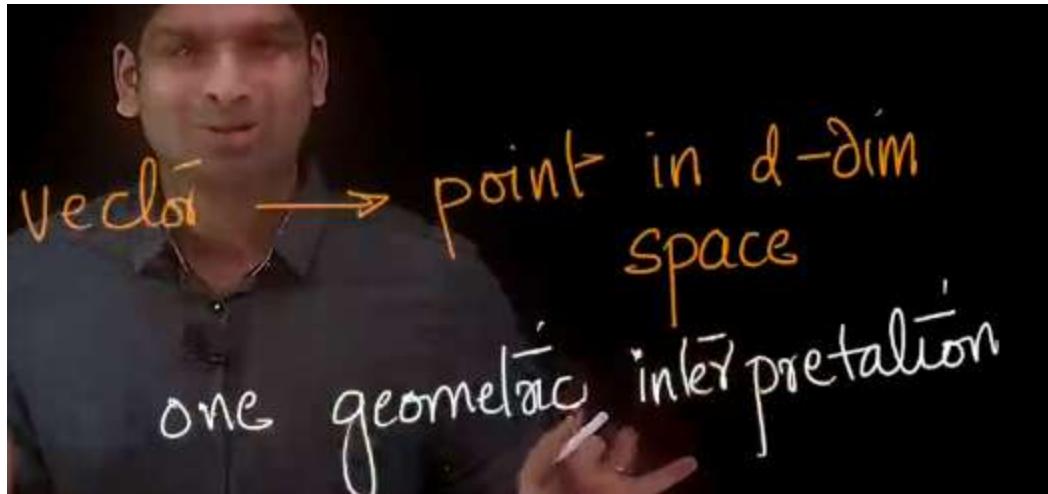
At timestamp 4.51



- Vectors in physics can be represented using magnitude and direction, we get the direction by obtaining the angle made by the vector with x-axis .
- Magnitude is nothing but length of the vector and can be calculated as shown above.\



- The magnitude of a vector is represented as shown above.
- A vector of magnitude or length 1 is a unit vector



- We can represent a vector as a point in d dimensional space.

At timestamp 11.45

Dataset

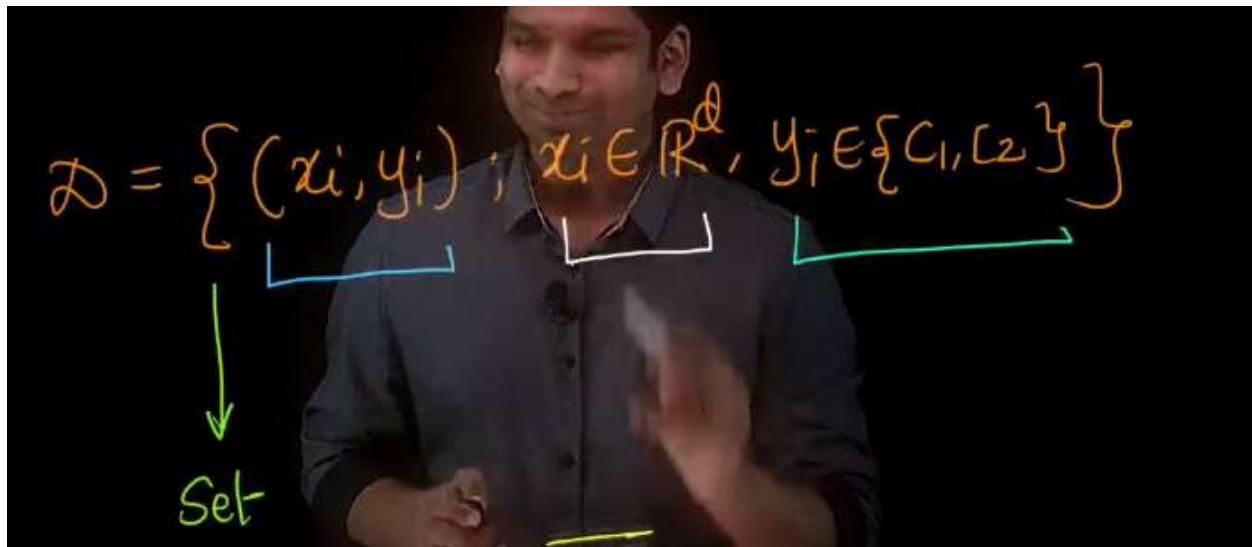
f_1	f_2	f_3	f_4	f_5	f_6	Type
		x_1				y_1
		x_2				y_2
		x_n				y_n

$x_i \in \mathbb{R}^6 \quad \forall i: 1 \rightarrow n$

$y_i \in \{C_1, C_2\}$

- Let our fish dataset has 6 features to represent each fish. So each row of our data set is a d dimensional row vector. The class to which the fish belongs to is given as type y_1, y_2 . Here we have two classes.
- You can see that x,y is data related to each fish
- Each x can be represented as a 6 dimensional vector.

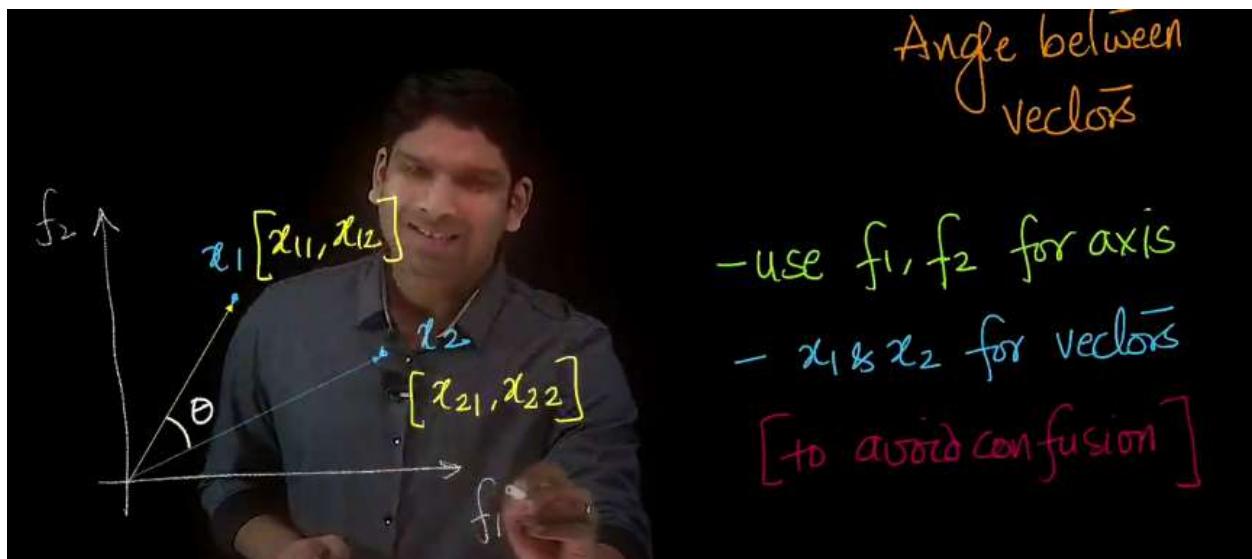
At timestamp 11.53



- We can represent our dataset mathematically as shown above .

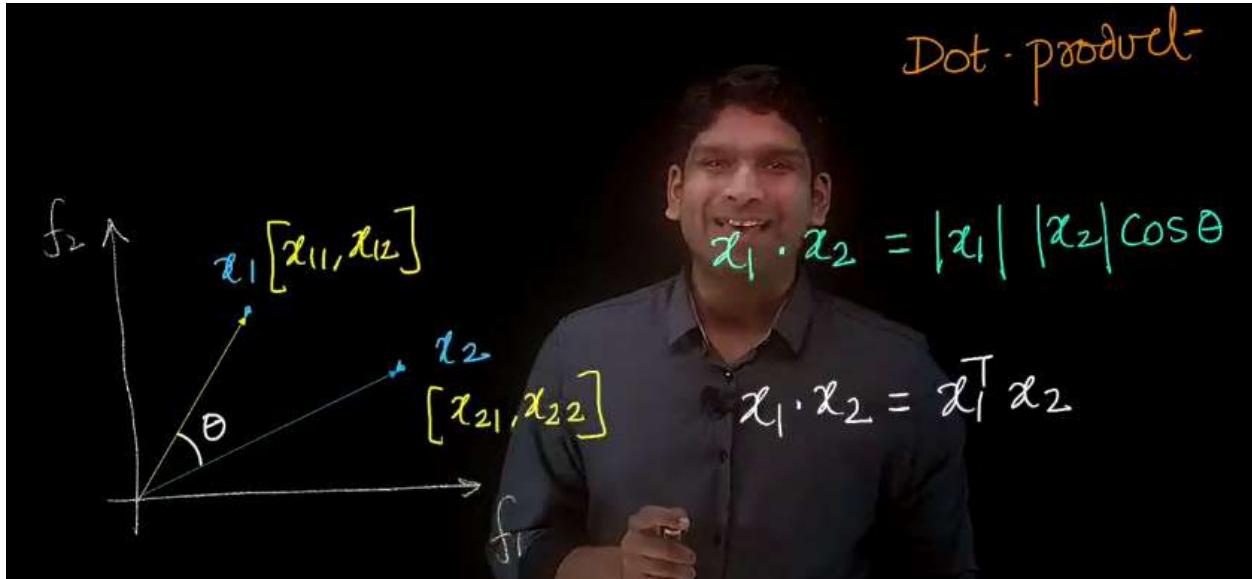
2.10 Dot product and angle between vectors

At timestamp 0.15



- In above snapshot f_1, f_2 are axis and x_1, x_2 are vectors θ is the angle between the vectors.

At timestamp 4.26



- Dot product is an operation between two vectors and is denoted as $x_1 \cdot x_2$ as shown above.
- For computing dot product between two vectors we need length of the two vectors and $\cos\theta$ (θ is angle between the vectors). Another way of interpreting dot product is if we have the vectors x_1 and x_2 (here we have two coordinates since 2D we can the same concept to nD as well) we can compute dot product by multiplying the vectors as shown.
- Given two vectors and their corresponding coordinates we can calculate $\cos\theta$ as shown below.

At timestamp 5.15 in video

$$\Rightarrow x_1^T x_2 = |x_1| |x_2| \cos \theta$$
$$\Rightarrow \begin{bmatrix} x_{11} & x_{12} \end{bmatrix} \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix} = \sqrt{x_{11}^2 + x_{12}^2} \cdot \sqrt{x_{21}^2 + x_{22}^2} \cdot \cos \theta$$

$$\Rightarrow \begin{bmatrix} x_{11} & x_{12} \end{bmatrix} \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix} = \sqrt{x_{11}^2 + x_{12}^2} \cdot \sqrt{x_{21}^2 + x_{22}^2} \cdot \cos\theta$$

$$\Rightarrow \frac{x_{11}x_{21} + x_{12}x_{22}}{\sqrt{x_{11}^2 + x_{12}^2} \cdot \sqrt{x_{21}^2 + x_{22}^2}} = \cos\theta$$

At timestamp 7.57

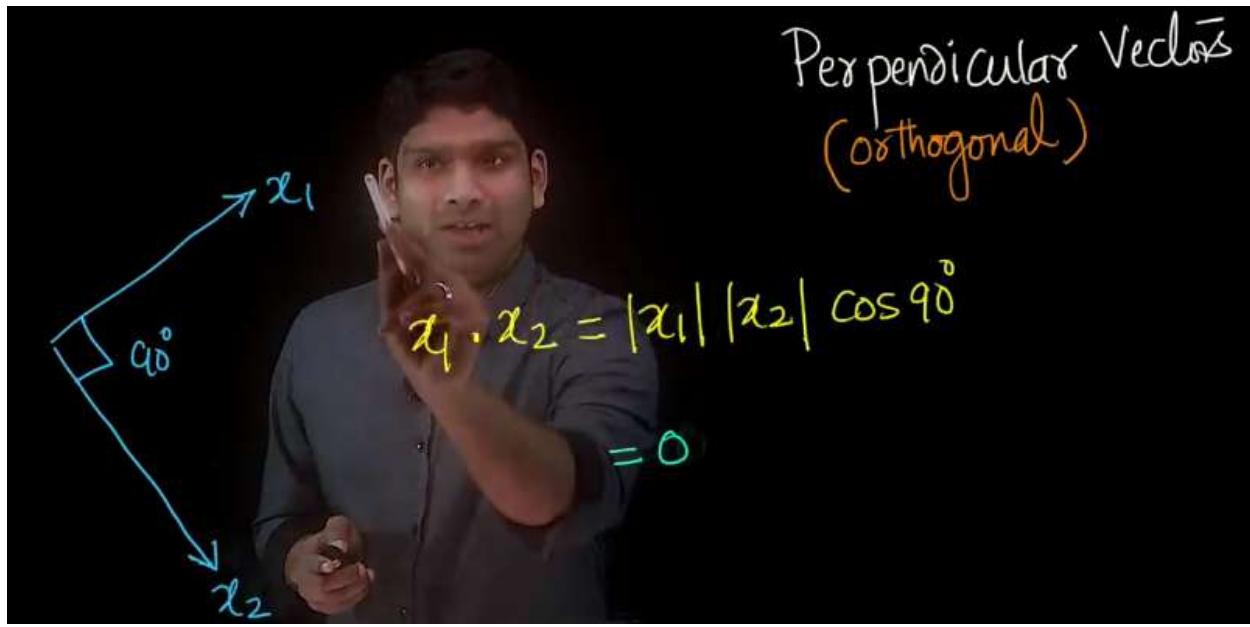
$$x_1 \cdot x_2 = |x_1| |x_2| \cos\theta$$

$$\cos\theta = \frac{x_1 \cdot x_2}{|x_1| |x_2|}$$

$x_1 \in \mathbb{R}^d$
 $x_2 \in \mathbb{R}^d$

- By equating and rearranging terms as shown above we calculated $\cos\theta$. (note that x_1 and x_2 should be of same dimensions because they both have to be in same d -dimensional space)
- What we learnt in 2D can transform it to higher dimensional space easily.

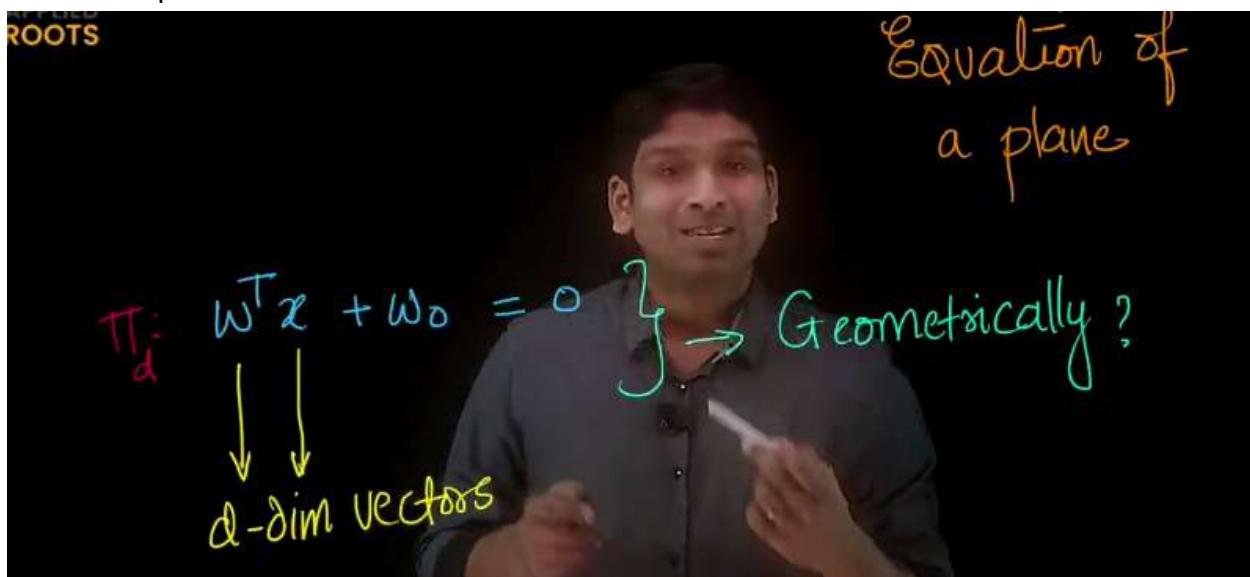
At timestamp 9.26



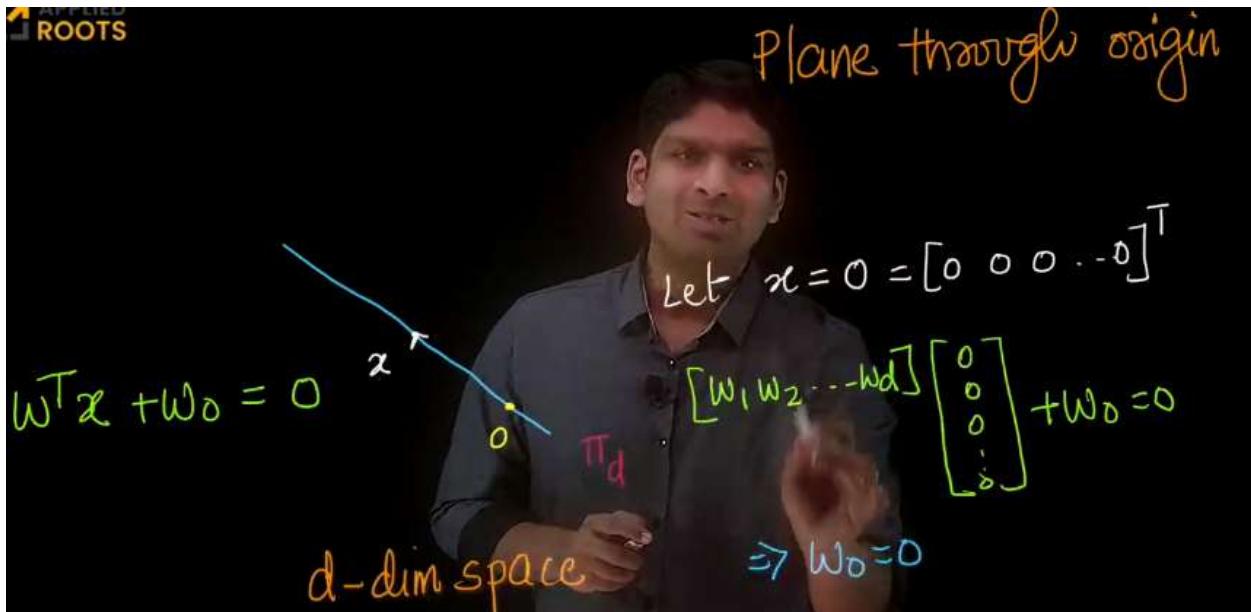
- When two vectors are perpendicular to each other then they are called Orthogonal vectors. (angle between the vectors is 90 degrees)
- The dot product of two perpendicular vectors is 0.
- If the dot product between two vectors is 0 we can conclude that they are Orthogonal vectors.

2.11 Geometry of the Equation of a plane

At timestamp 0.41 in video

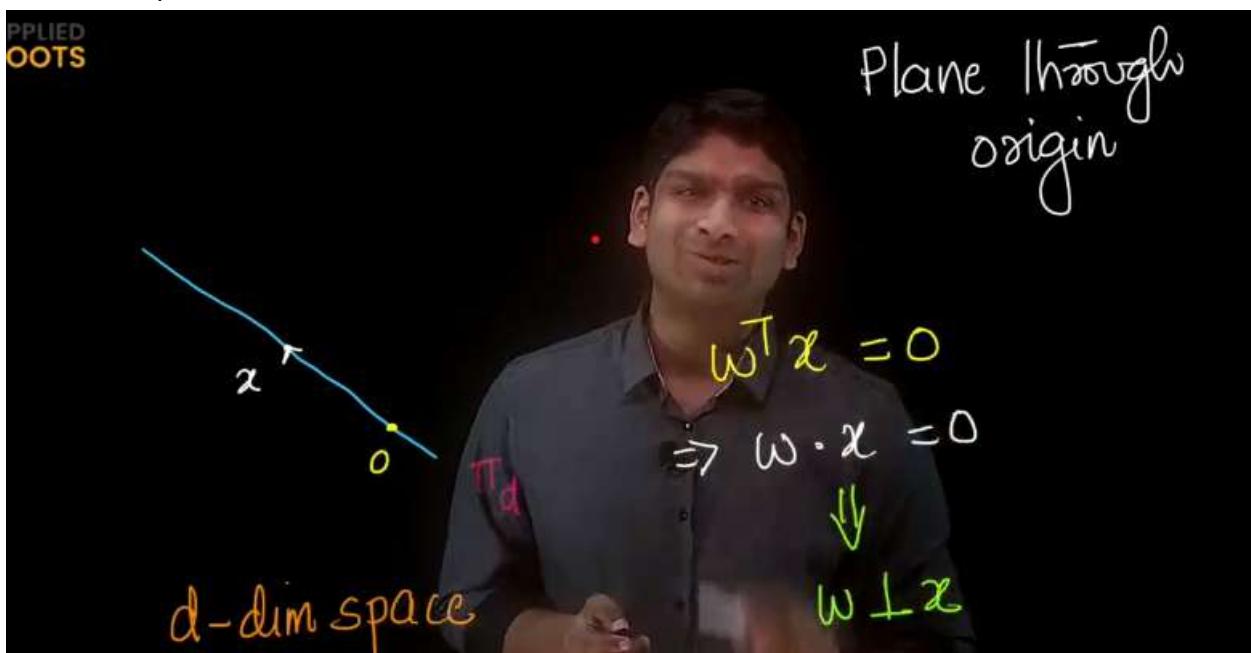


- Just like we understood the geometry of a line we can understand a plane geometrically.
- At timestamp 3.32 in video



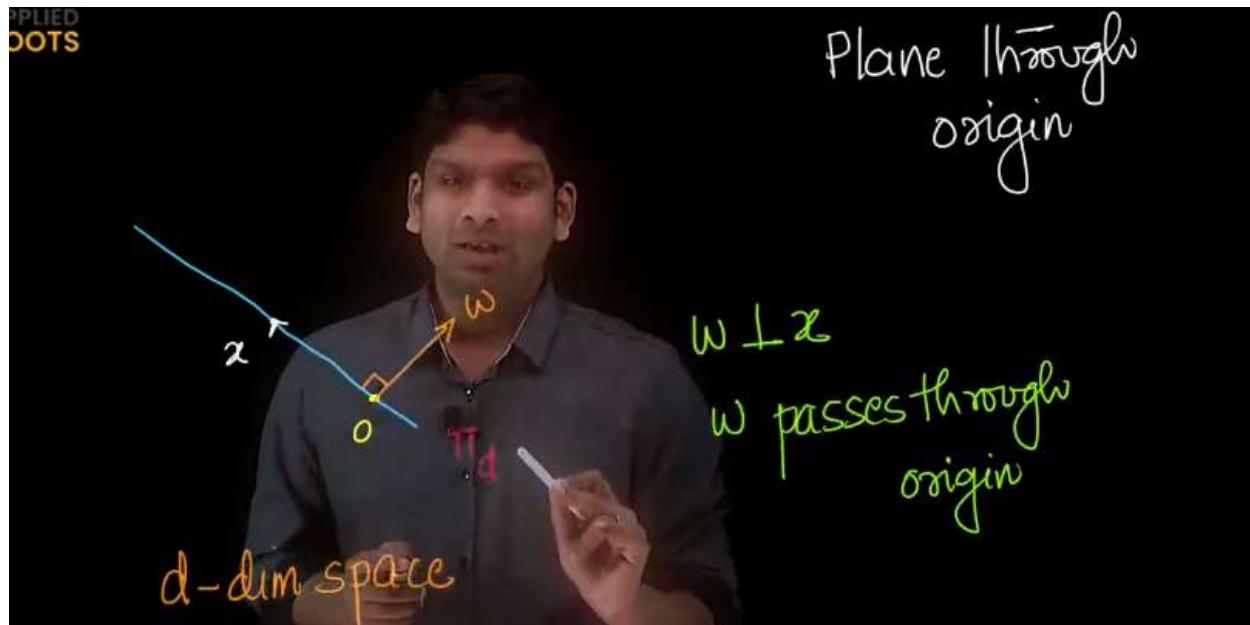
- Let's consider a plane passing through origin and the equation of plane is as shown above. This plane can be in any dimensional space if $d=2$ it's a line, if $d=3$ it's a plane, if $d>4$ they are all hyperplanes.
- Since our plane passes through origin we substitute $x=0$ in the equation, which gives us $w_0=0$. This implies when a plane passes through origin our $w_0=0$.

At timestamp 4.40 in video

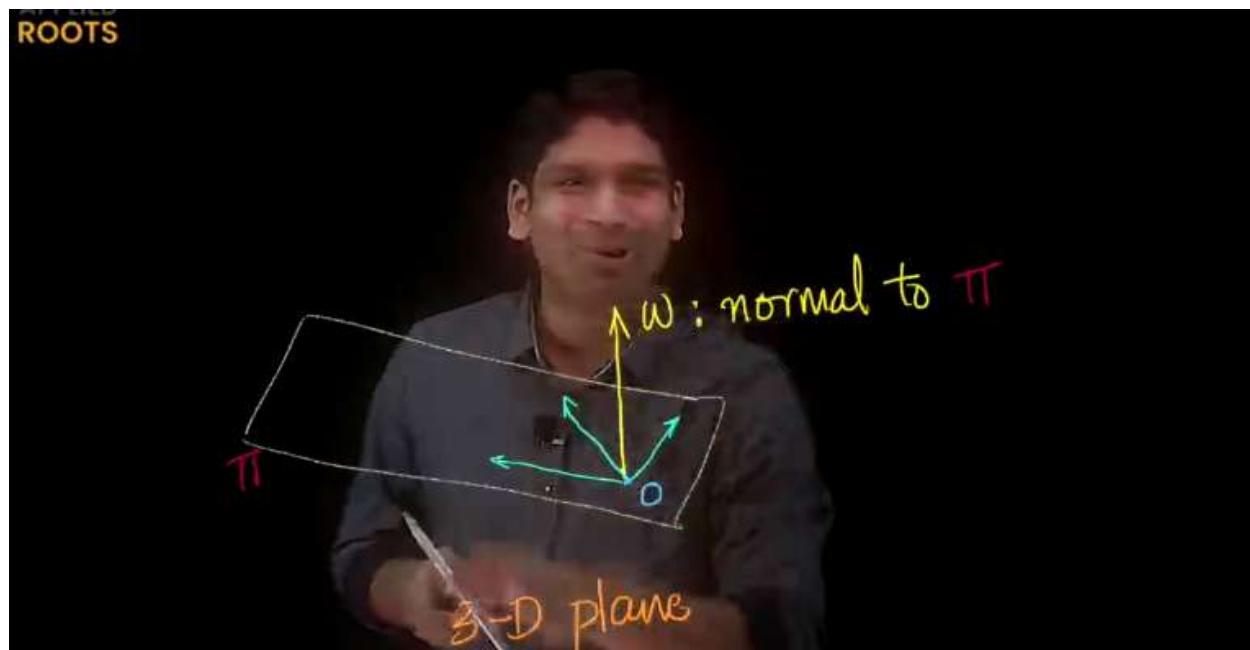


- For a plane passing through origin since $w_0=0$. The dot product of vectors $w \cdot x$ is 0, so they are perpendicular vectors.(angle between x and w is 90 degrees)
- w will be perpendicular to every point x on the plane.

At timestamp 5.50 in video

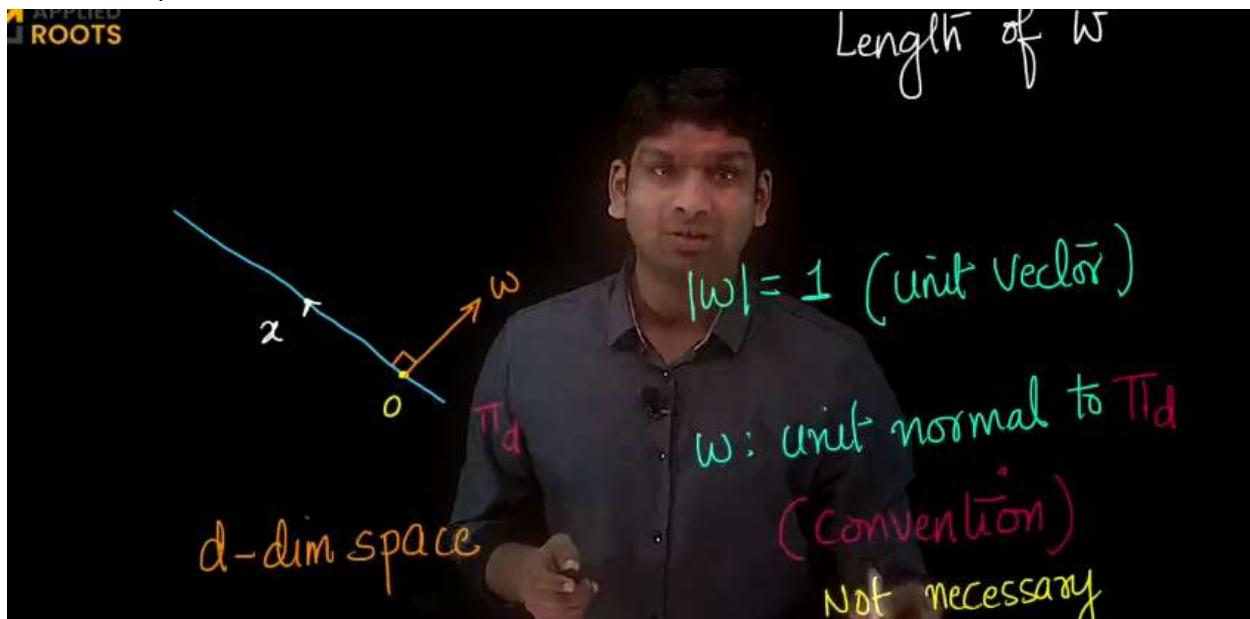


- w will be perpendicular to plane as shown above.



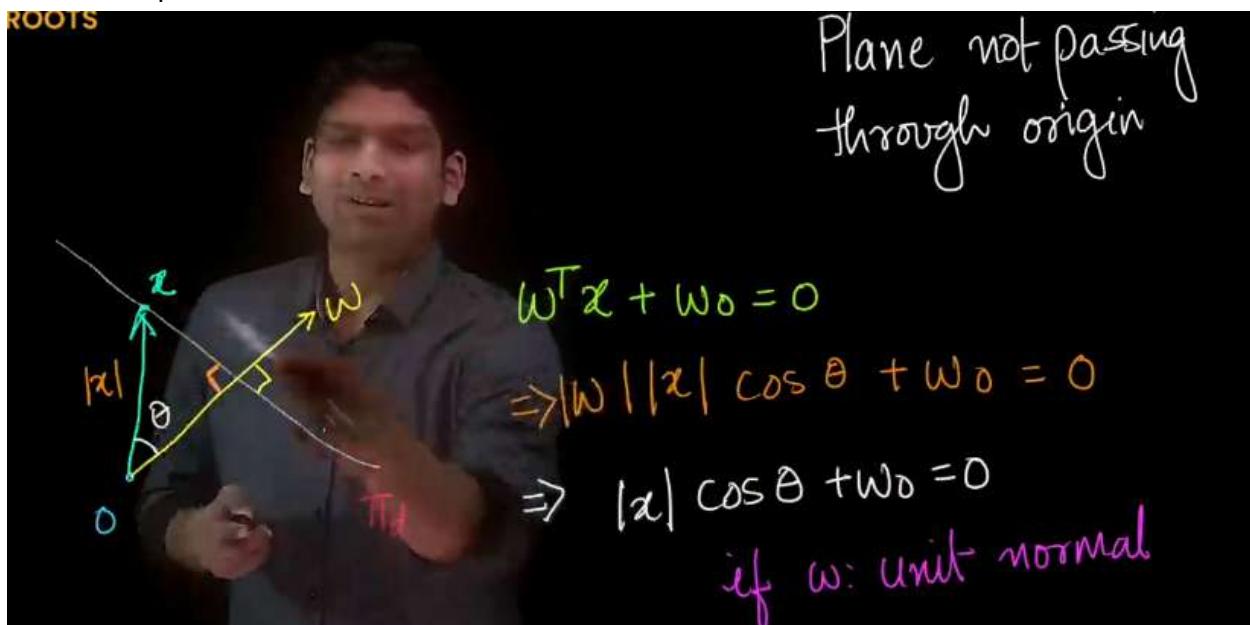
- You can visualize w in 3D as shown above.

At timestamp 8.33 in video



- By convention whenever we use a normal vector to a plane we typically use a unit vector but it is not necessary.
- When the length of vector $w=1$ we refer to it as the unit normal vector to plane.

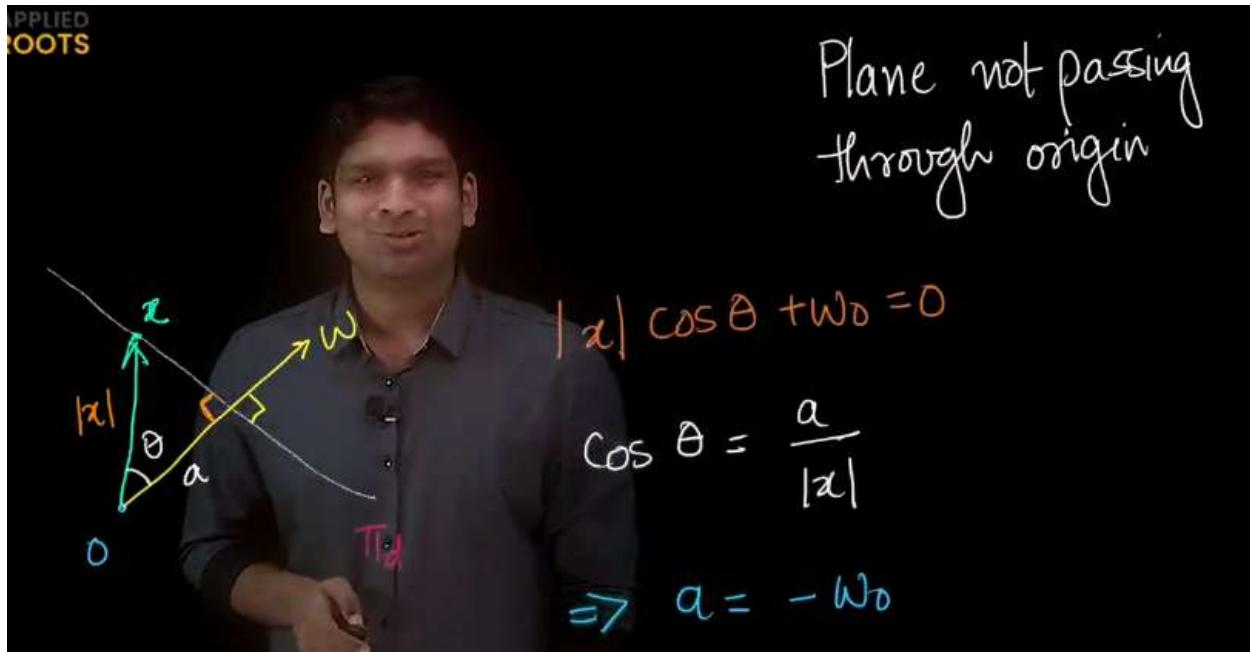
At timestamp 13.8 in video



- For a plane not passing through origin the equation of plane is as shown above. w is normal vector to the plane.Lets consider a point x on the plane .(ox is a vector and w is our normal vector)

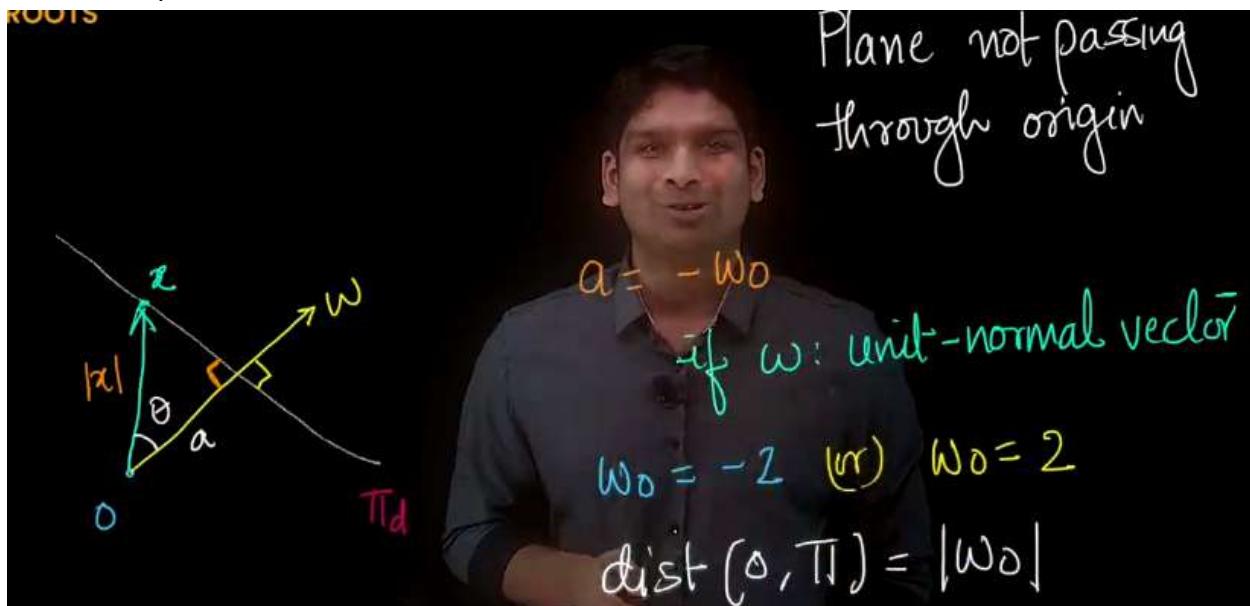
- Now we assume w to be unit vector and find the dot product between vectors x and w as shown above and $\cos\theta$ is not equal to 0. w_0 will not be equal to 0 since plane is not passing through origin

At timestamp 15.8 in video



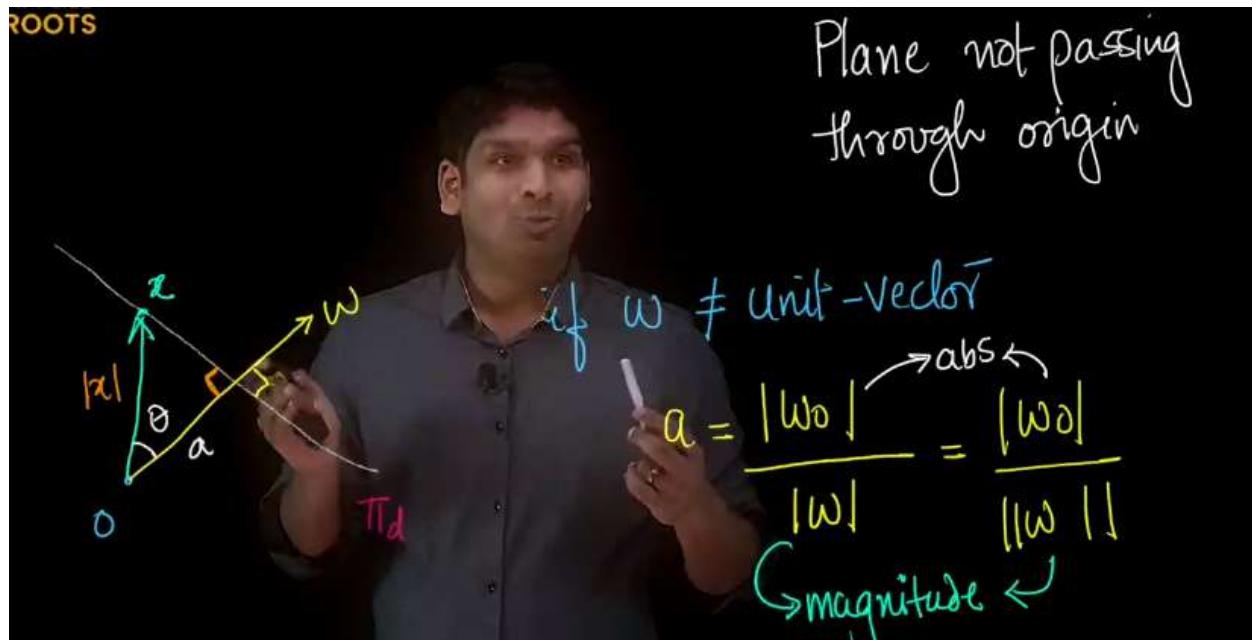
- We can calculate $\cos\theta$ as shown above and we can clearly see that $a=-w_0$.

At timestamp 16.26 in video



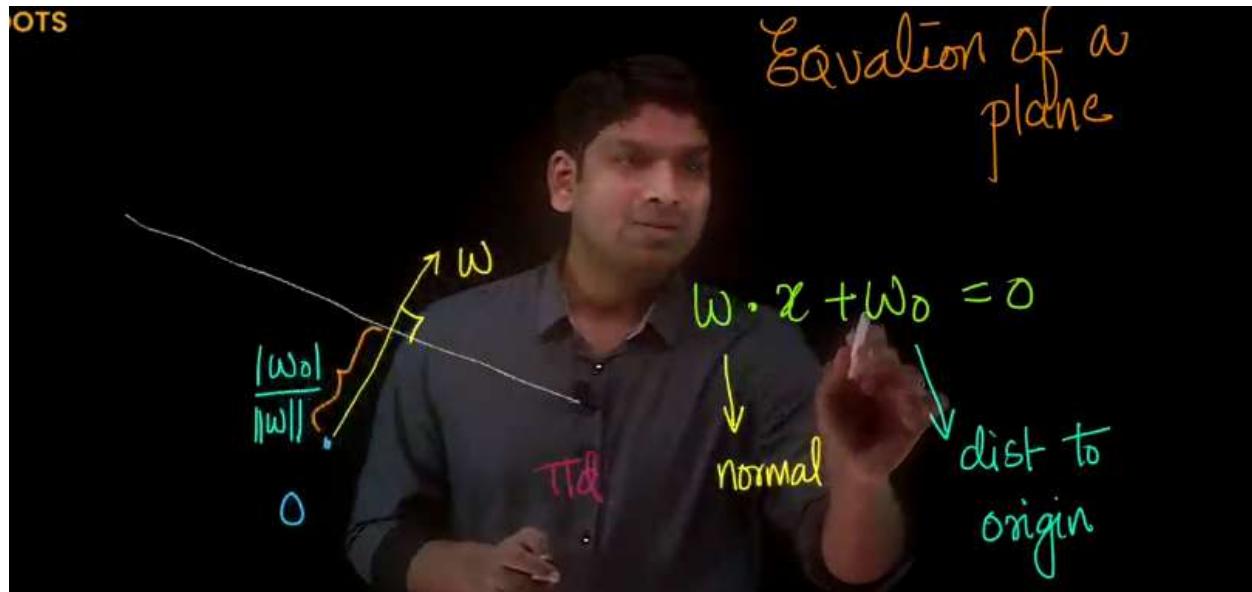
- a is nothing but the shortest distance from origin to the plane and we know that distance cannot have negative values (-ve represents the direction). So, the distance from origin to the plane is $\text{abs}(w_0)$

At timestamp 17.11 in video



- For plane not passing through origin and w is not a unit vector then a can be written as shown above.

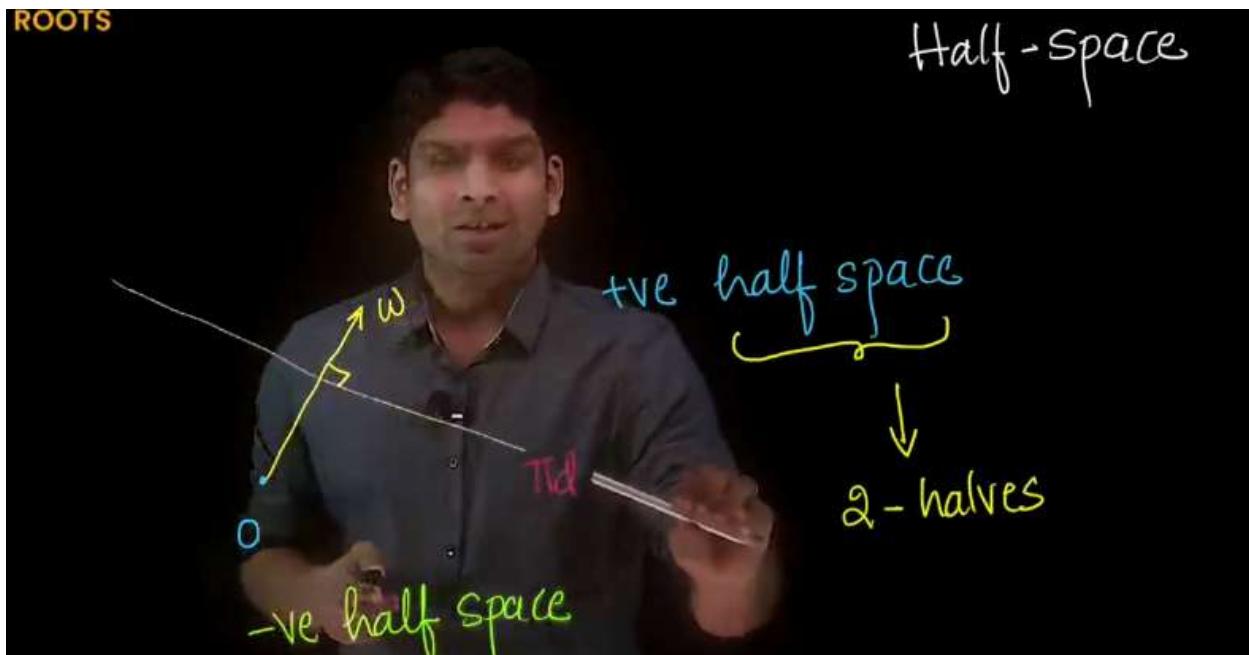
At timestamp 21.53 in video



- Given the equation of plane w is vector normal to the plane and passing through origin, w_0 is the distance from origin to the plane.

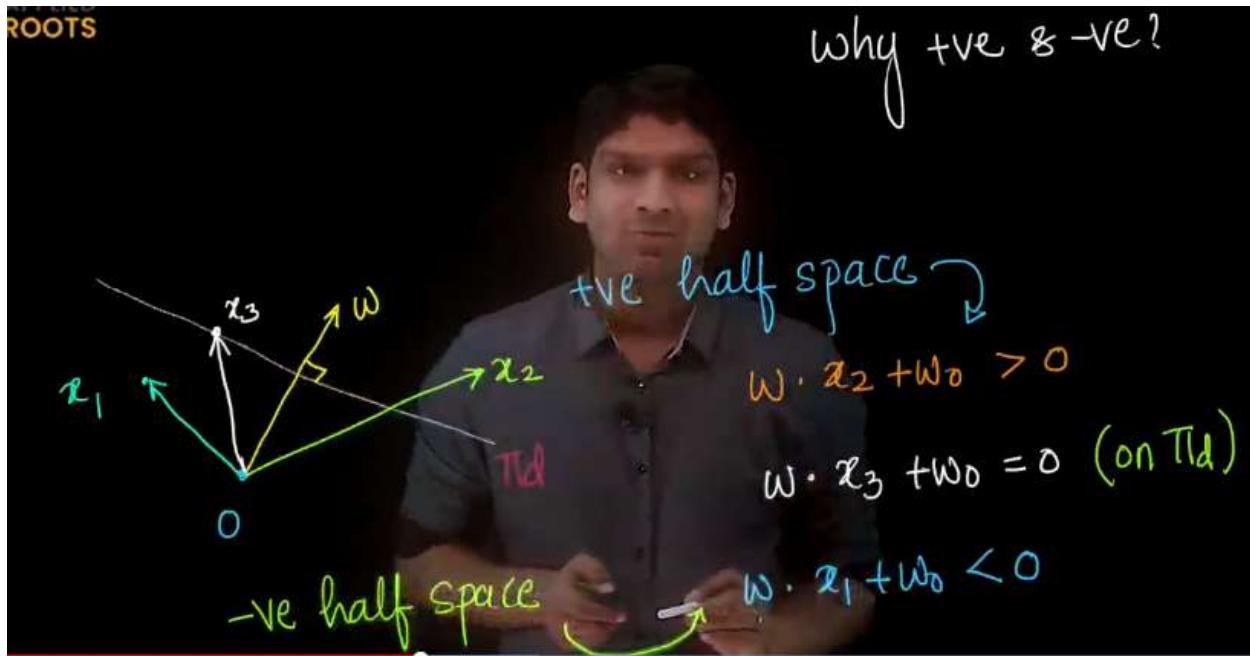
2.12 Half spaces & classification using a plane

At timestamp 0.42 in video



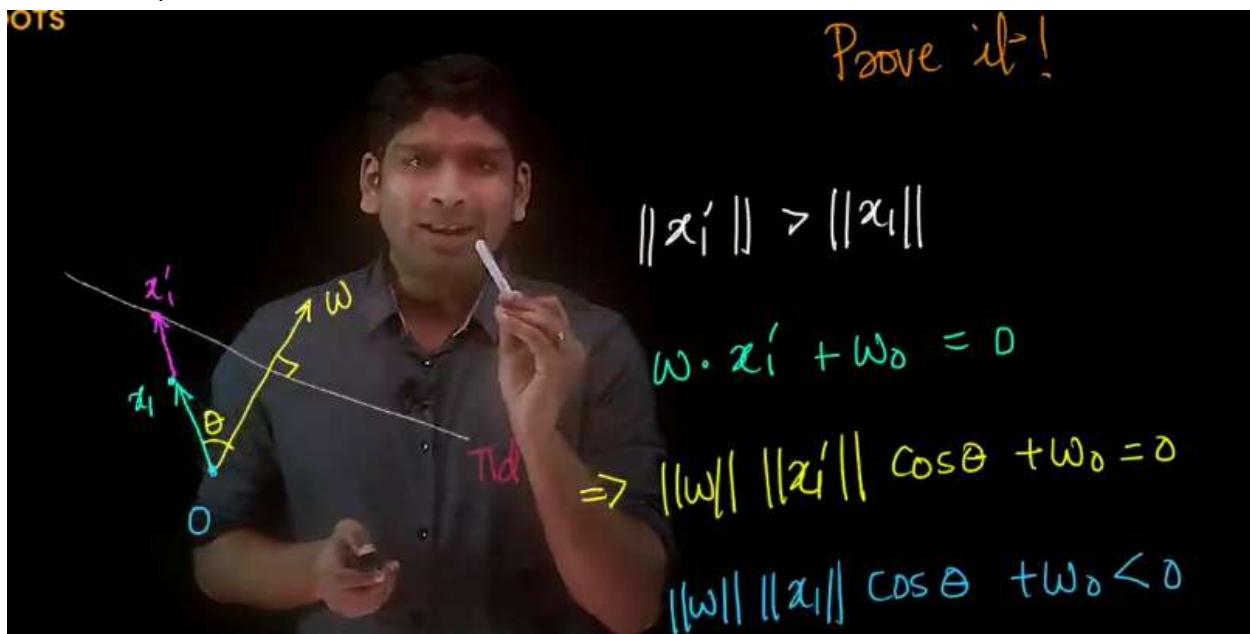
- Now we use the concept of half-spaces to actually classify the points in d dimensional space.
- Given a plane in d-dimensional space and normal vector w as shown above, if we assume it to be a 2D space the line divides the total space into 2 halves everything above the plane and everything below the plane. (+ve half and -ve half). likewise a plane separates the 3D space into two regions.
- The space above the plane in the direction of w is positive half space and space below the plane opposite direction of w is negative half space

At timestamp 4.40 in video



- Let's consider three points x_1 below the plane, x_2 above the plane and x_3 on the plane.
- As shown above for x_2 and all the points above the plane $w \cdot x_2 + w_0 > 0$, and for points below the plane the value is less than 0 and for points lying on the plane it is 0.
- Let's understand each scenario one by one.

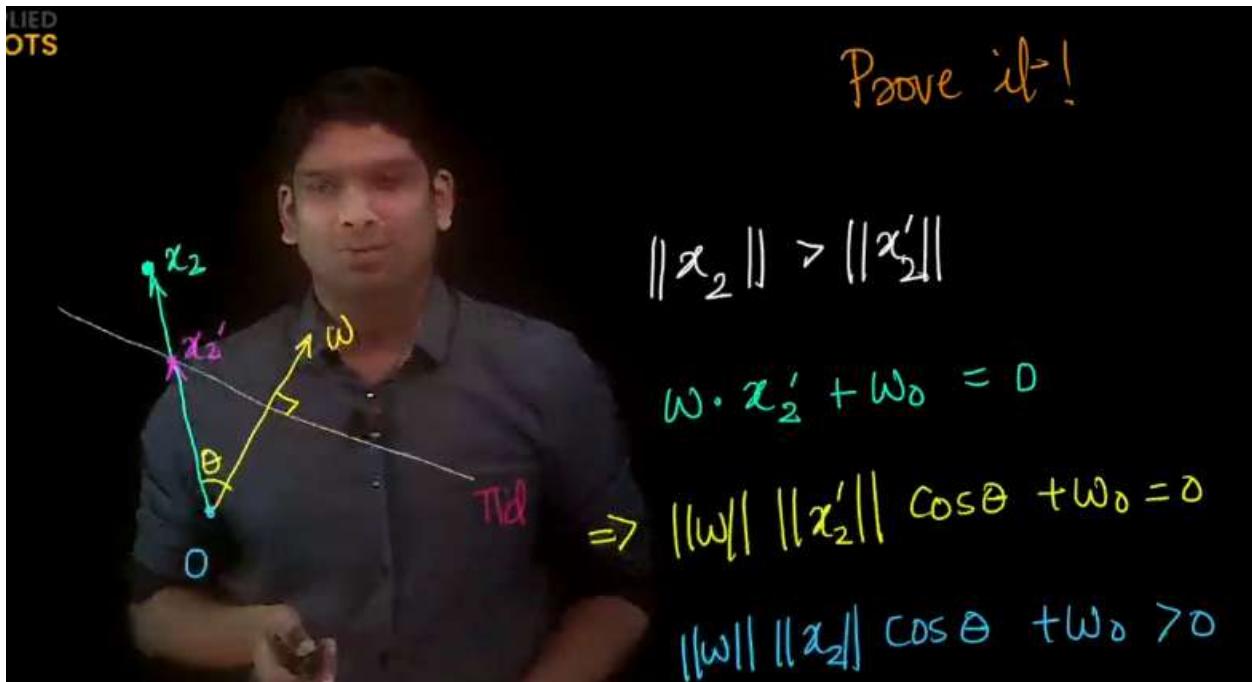
At timestamp 7.27 in video



- As shown above we have our plane and normal vector to the plane w and we have point x_1' below the plane in negative half space.

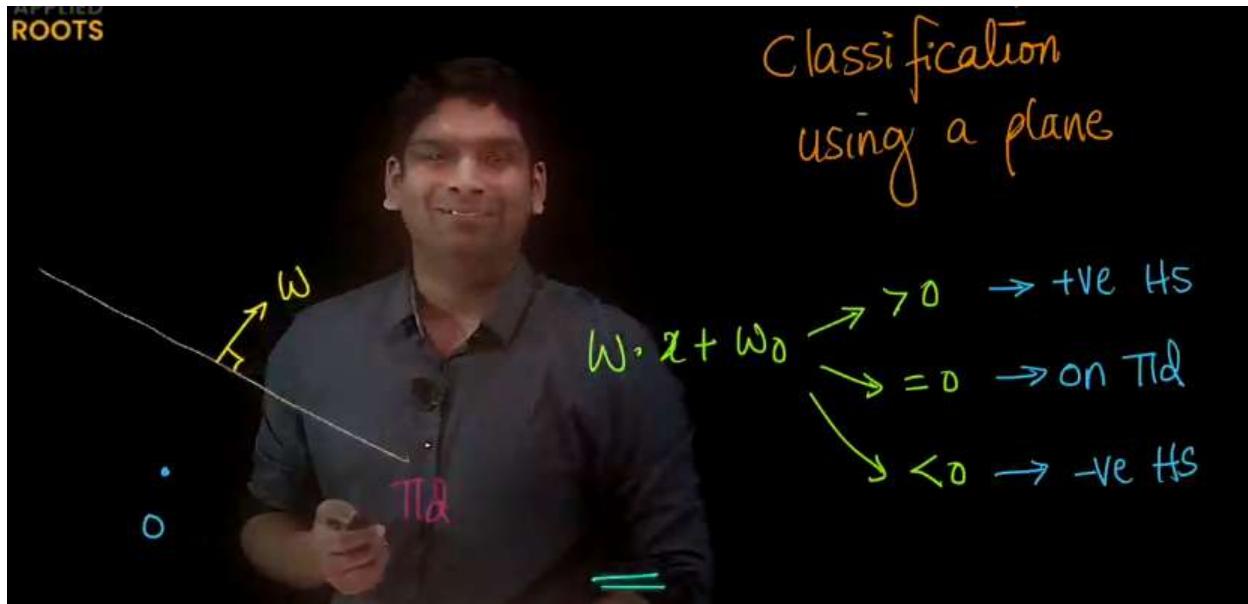
- We extend the vector x_1 slightly till it touches the plane and we call the point x_1' . Now we have two vectors x_1 and x_1' .
- As shown above the magnitude of x_1' is greater than the magnitude of x_1 .
- We know that $w \cdot x_1' + w_0 = 0$ since x_1' lies on the plane. From above equations it's clear that $w \cdot x_1 + w_0 < 0$

At timestamp 8.50 in video



- We now consider a point x_2 in positive half space, let's call the point at which the vector x_2 intersects the plane be x_2' . Now we have two vectors x_2 and x_2' .
- From the above diagram it's clear that length of x_2 is greater than length of x_2' .
- We know that $w \cdot x_2' + w_0 = 0$ since x_2' lies on the plane. From above equations it's clear that $w \cdot x_2 + w_0 > 0$

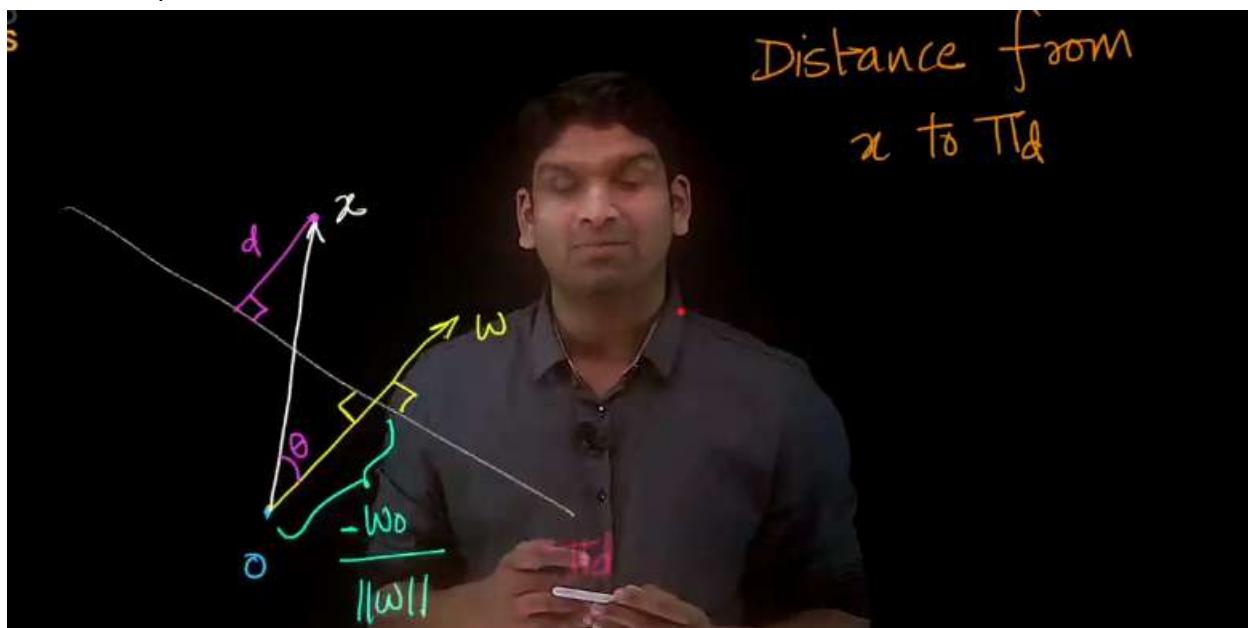
At timestamp 11.53 in video



- To summarize we perform classification using the plane based on above conditions.
- When we use a plane to separate any point in d -dimensional space ,we determine whether the point is above or below the plane.

2.13 Distance of a point from a plane

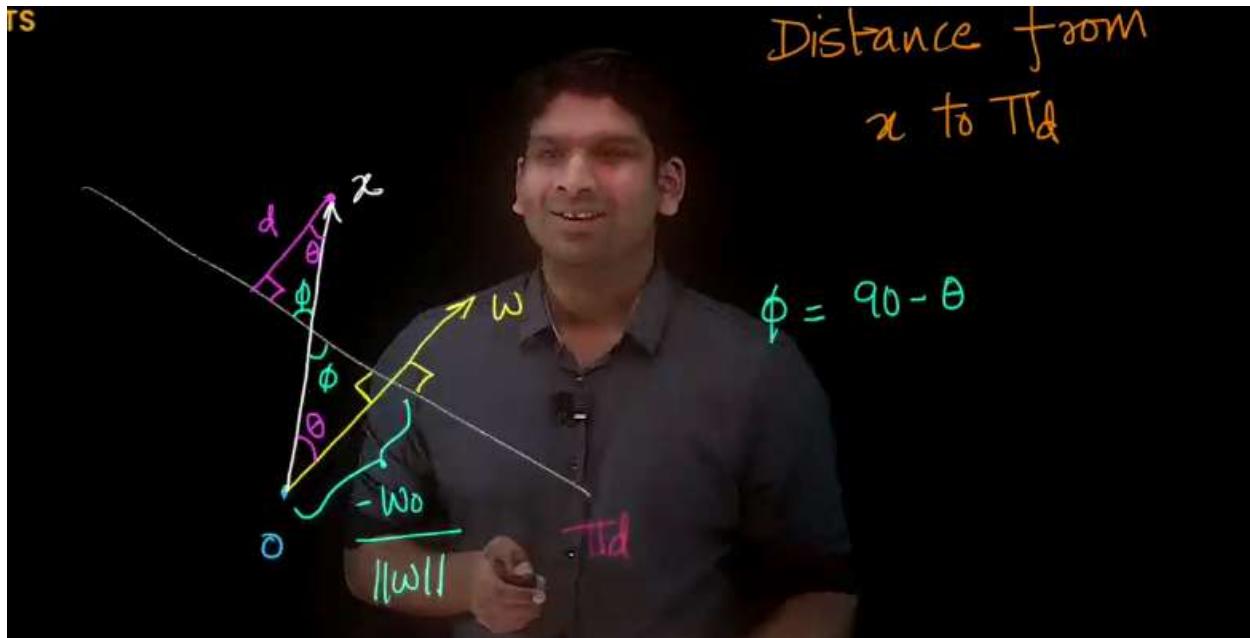
At timestamp 0.13 in video



- Given any point we will find which side of the plane it belongs to and also its important to know how far away it is from the plane.

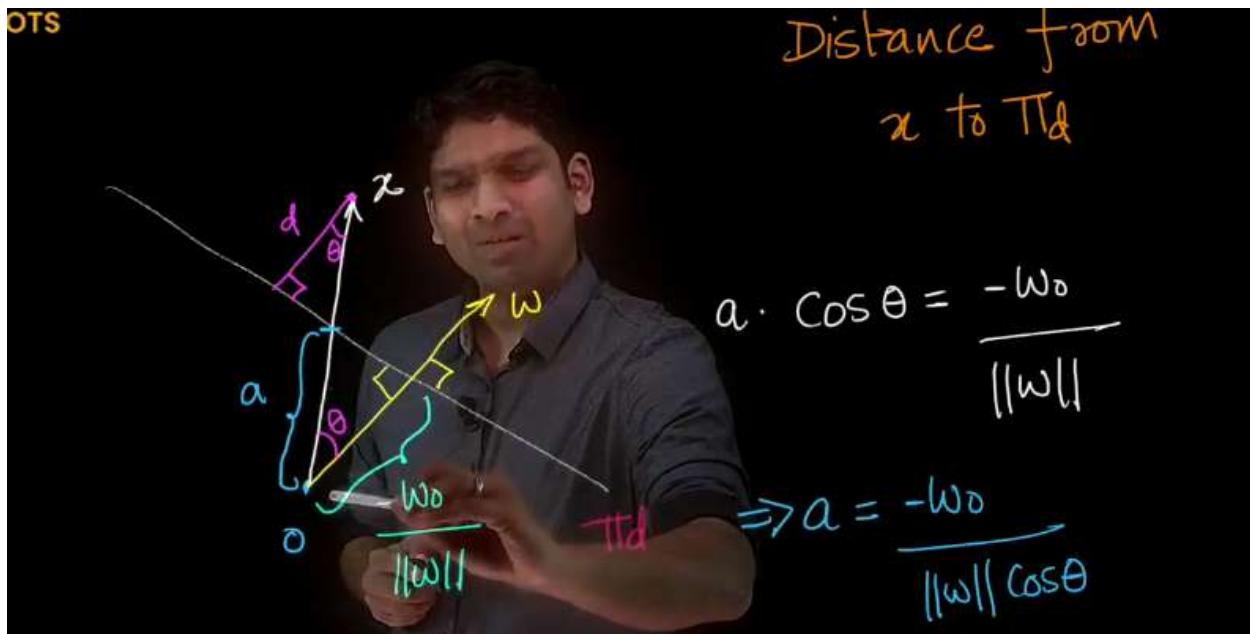
- Given any plane and a point x , we know the normal vector w and also the distance from origin to the plane as shown above. Let's find the distance from point x to the plane
- Distance from plane to the point x is nothing but the length of the line segment that is perpendicular to plane and passing through the point x . (d as shown above)

At timestamp 14.12 in video



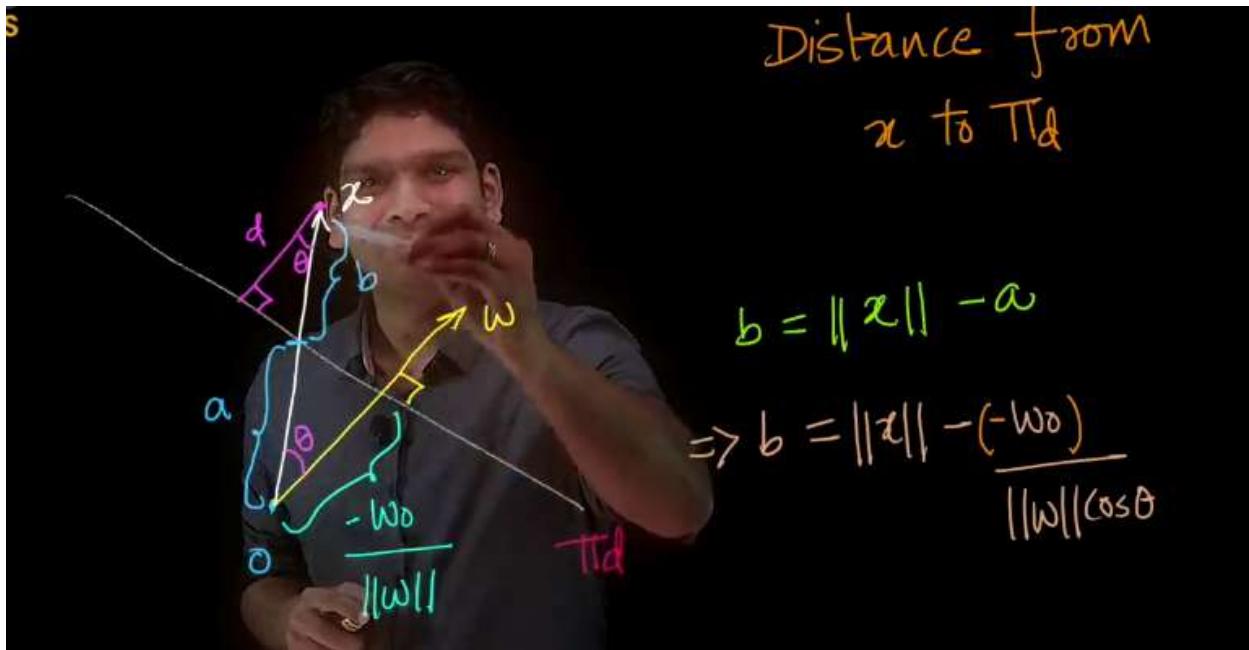
- From the above diagram we can say that $\phi = 90 - \theta$

At timestamp 4.49 in video



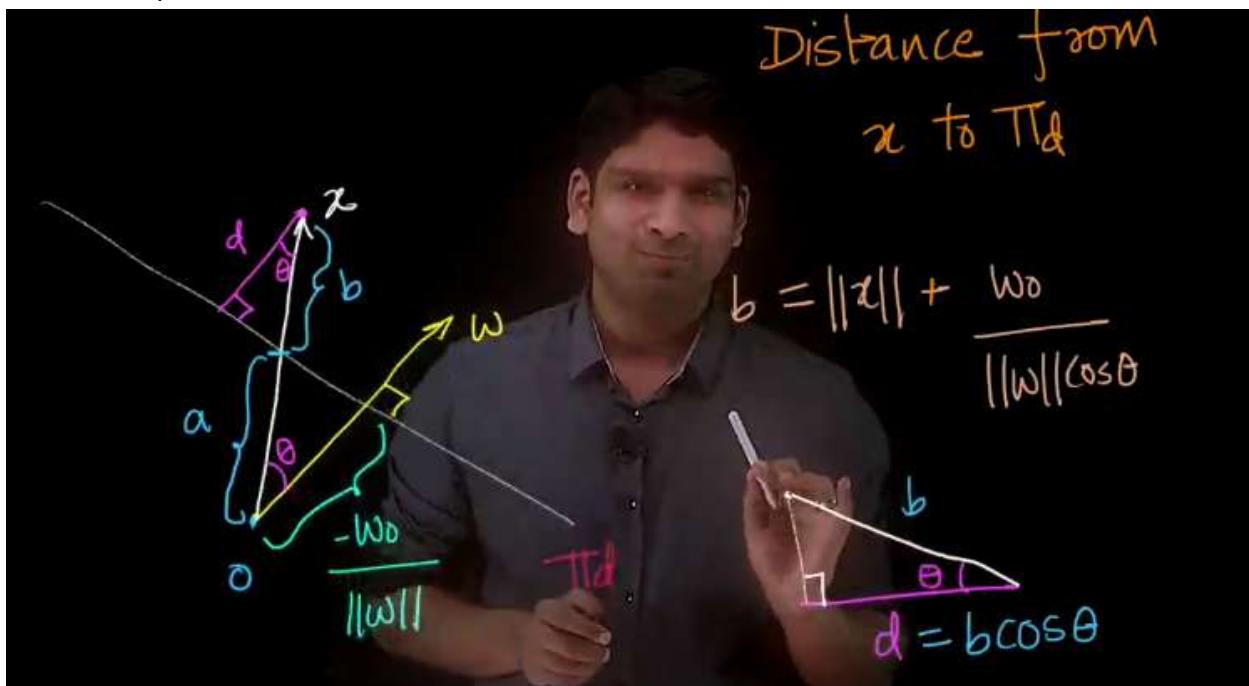
- From the triangle shown above we can calculate a .

At timestamp 5.51 in video



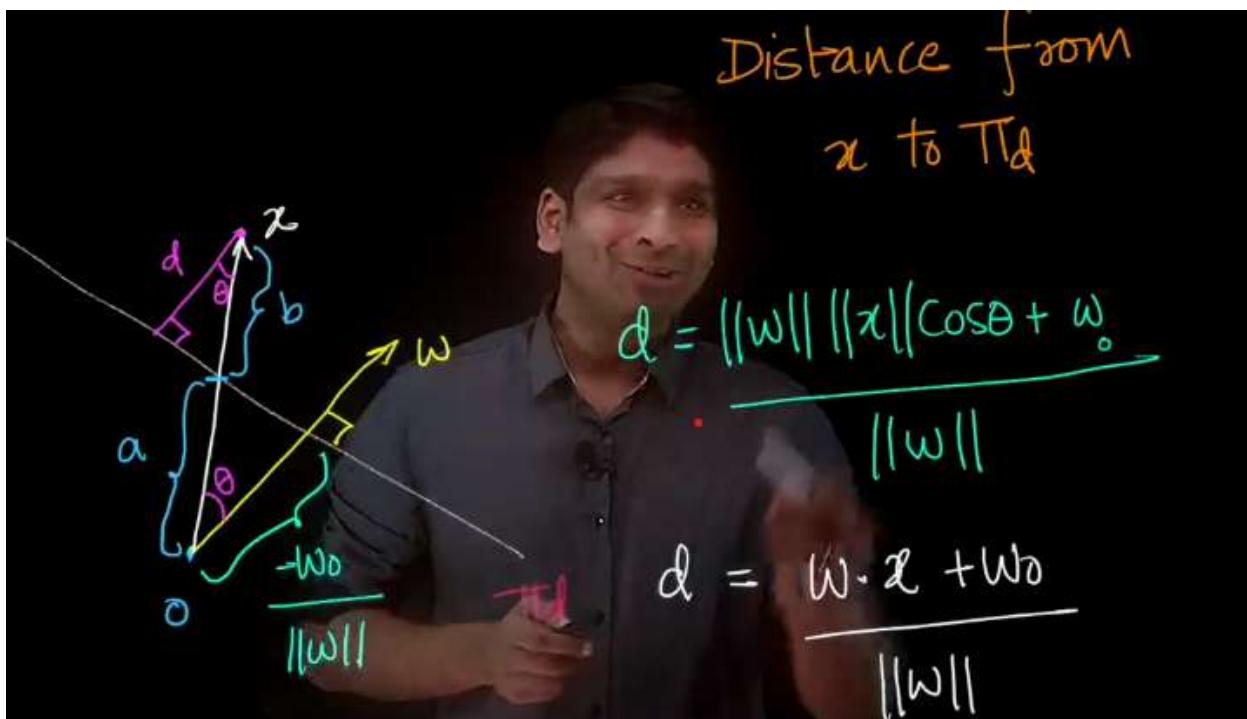
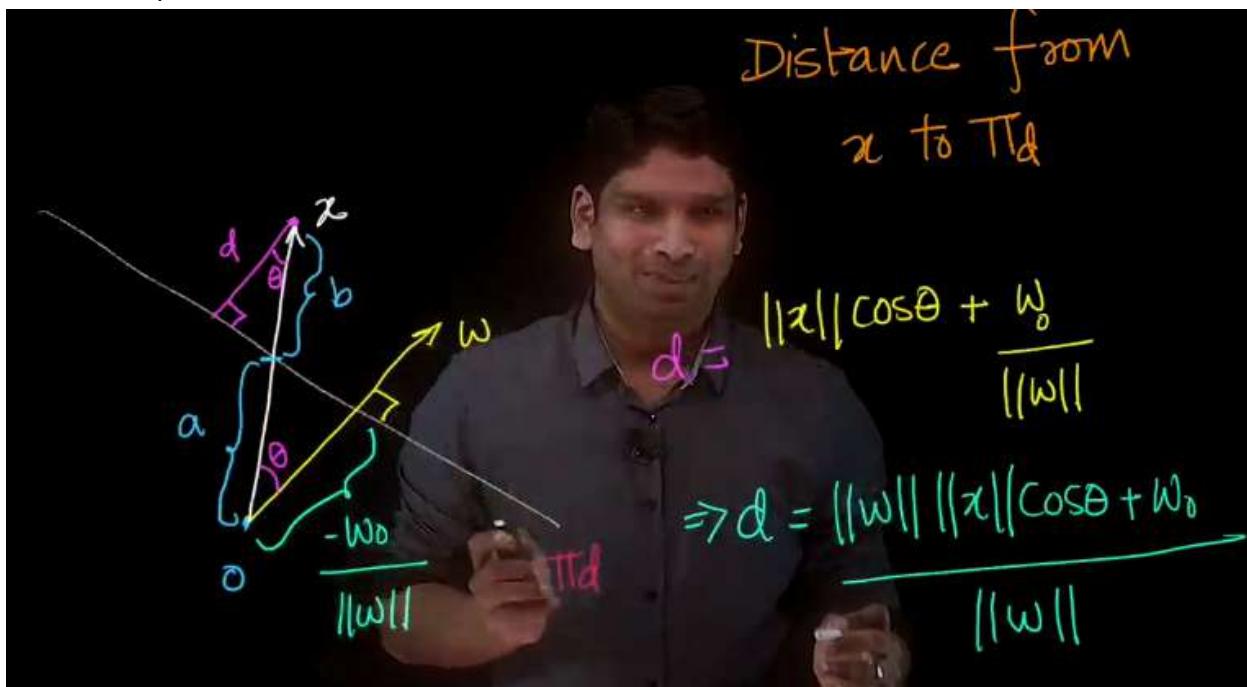
- By subtracting a from the length of vector x we can get b as shown.

At timestamp 7.38 in video



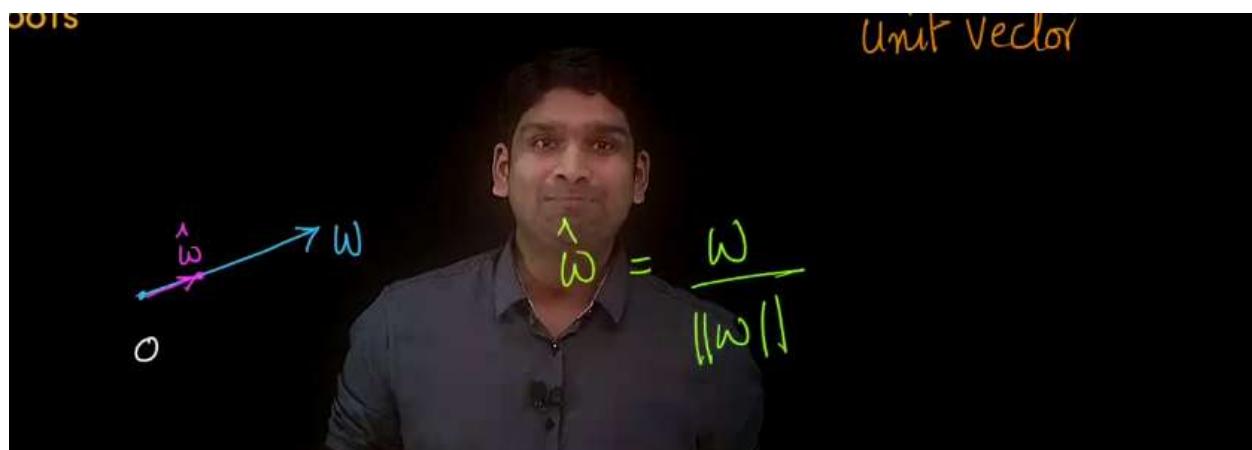
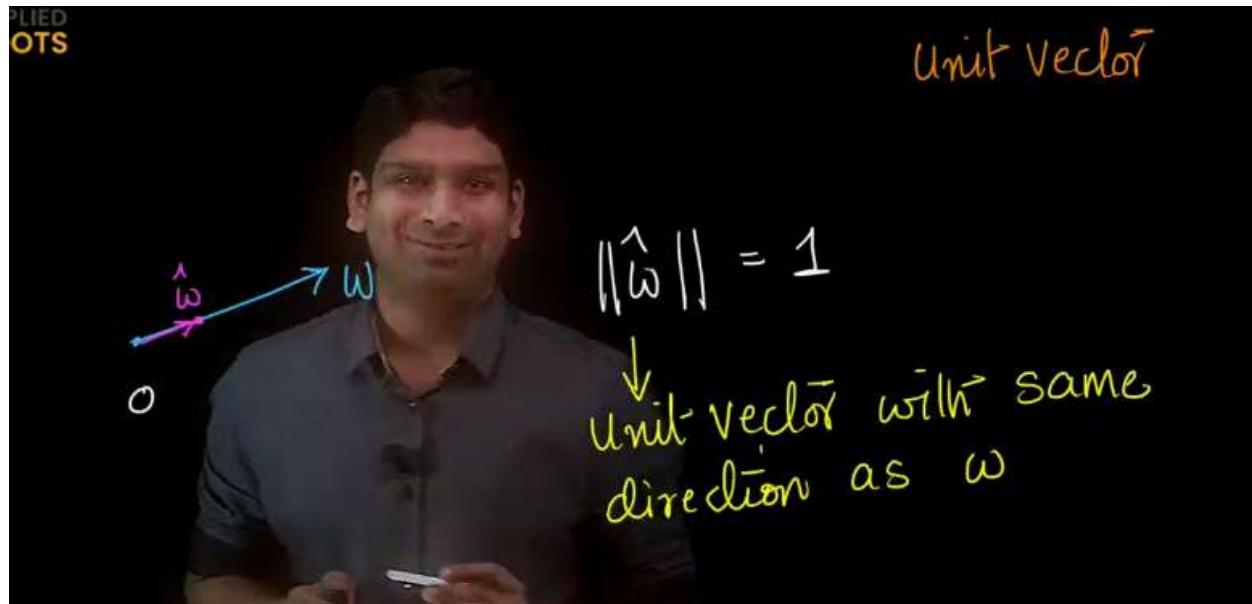
- We can calculate $d = b \cos \theta$ from the above triangle.
- We substitute b value in the above equation and get d value as shown below.

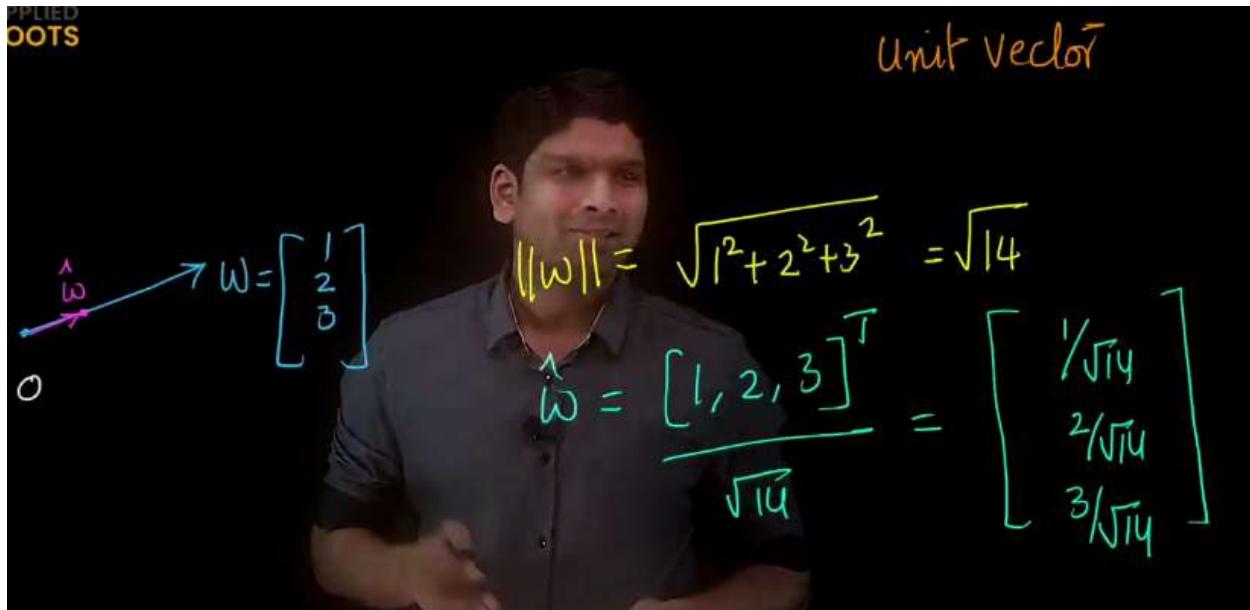
At timestamp 8.53



2.14 Projection & arithmetic operations on vectors

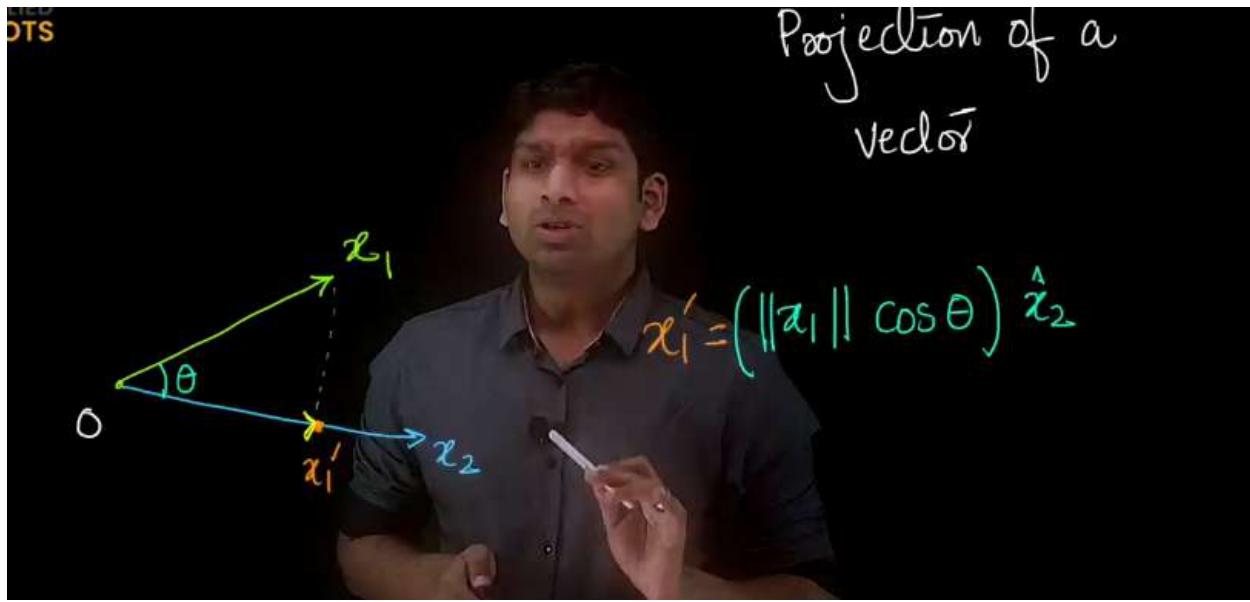
At timestamp 0.40 in video





- Let we have a vector W and its length need not be 1 .If we want to represent a vector whose length is equal to 1 but in the same direction of W we call such a vector as a unit vector.
- When we are given a vector W , we can obtain a unit vector which has the same direction as W as shown above.

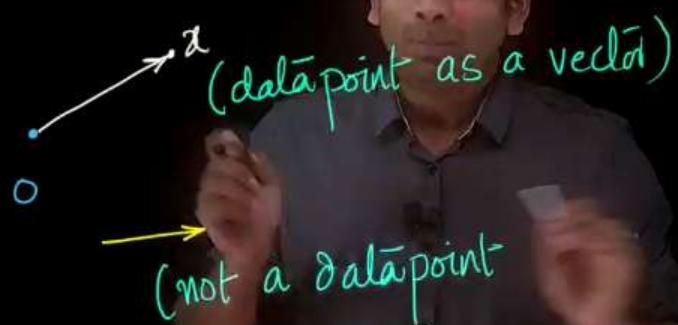
At timestamp 4.6 in video



- Let we have two data points x_1 and x_2 ,if we want to project vector x_1 on to x_2 we can draw a perpendicular line from x_1 onto x_2 and we call it x_1' as shown above.The direction of x_1' is the same as x_2 .

ROOTS

Vectors anywhere
in Space



- Whenever we have a datapoint we draw a line from origin to the point and we call it a vector. But vectors can be anywhere like the yellow line shown above and we cannot interpret it as a datapoint.

At timestamp 8.47 in video

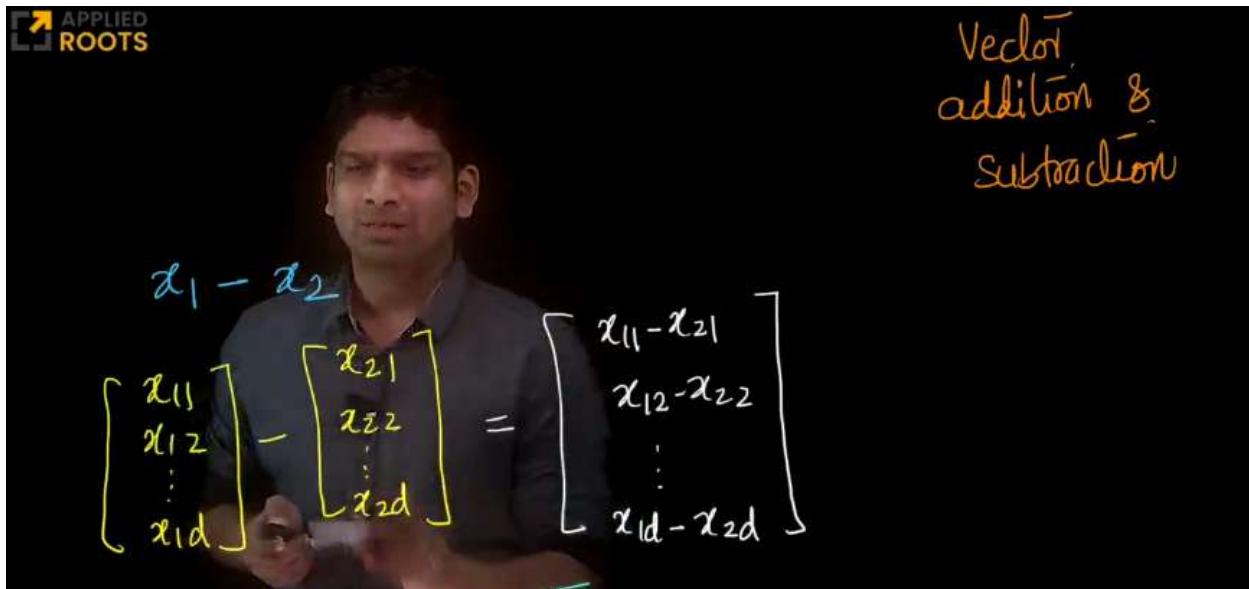
ROOTS

Addition of
Vectors

$$\vec{OP} + \vec{PQ} = \vec{OQ}$$
$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} + \vec{PQ} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$
$$\Rightarrow \vec{PQ} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

- We can perform vector addition as shown above.

At timestamp 10.56 in video



The image shows a man in a grey shirt standing in front of a chalkboard. He is writing a vector subtraction equation. The equation is:

$$\begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1d} \end{bmatrix} - \begin{bmatrix} x_{21} \\ x_{22} \\ \vdots \\ x_{2d} \end{bmatrix} = \begin{bmatrix} x_{11} - x_{21} \\ x_{12} - x_{22} \\ \vdots \\ x_{1d} - x_{2d} \end{bmatrix}$$

Handwritten text above the board reads "Vector addition & Subtraction". The logo "APPLIED ROOTS" is visible in the top left corner of the chalkboard.

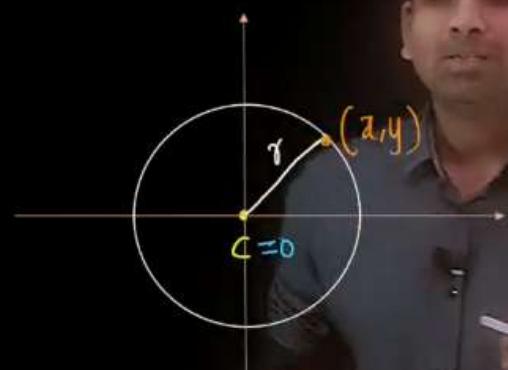
- Vector addition and subtraction be done even in d dimensions as shown .
- Note that for vector addition,subtraction and dot product the vectors should be of the same dimensions.

2.15 Equations of circle, sphere & hypersphere

At timestamp 0.37 in video

LIED
OTS

Circle (2D)

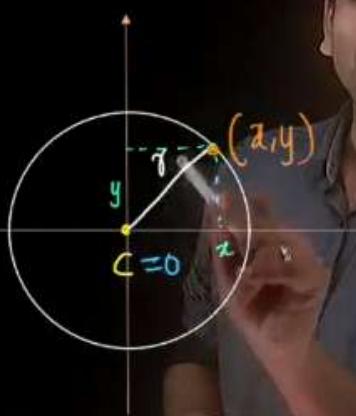


center = origin

r = radius

TS

Circle (2D)

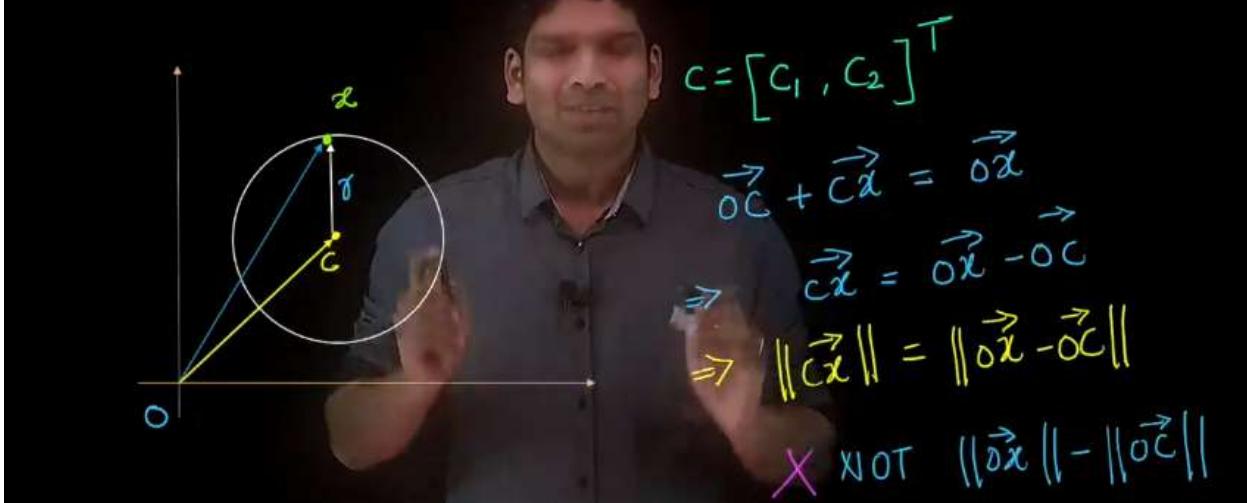


$$\text{dist} = \sqrt{x^2 + y^2} = r \\ \Rightarrow x^2 + y^2 = r^2$$

- Circle in 2D can be represented as shown above with radius r , centre $c=0$. To represent any circle we need its centre c and radius r .

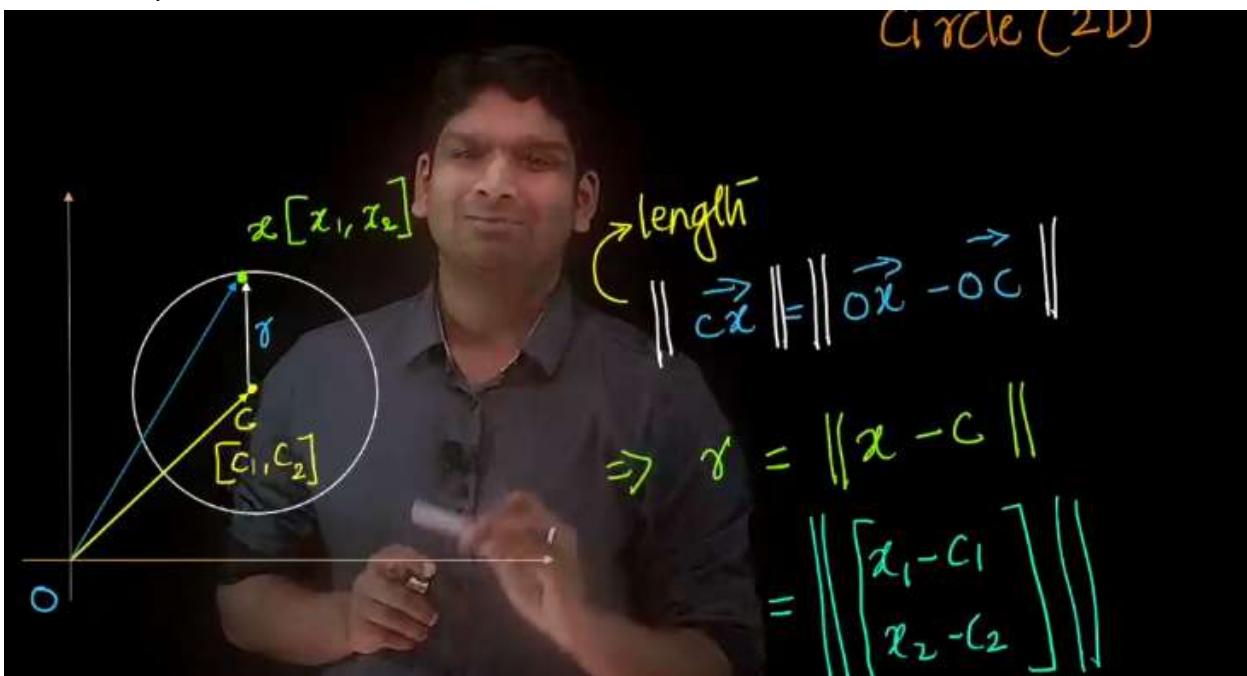
At timestamp 4.52 in video

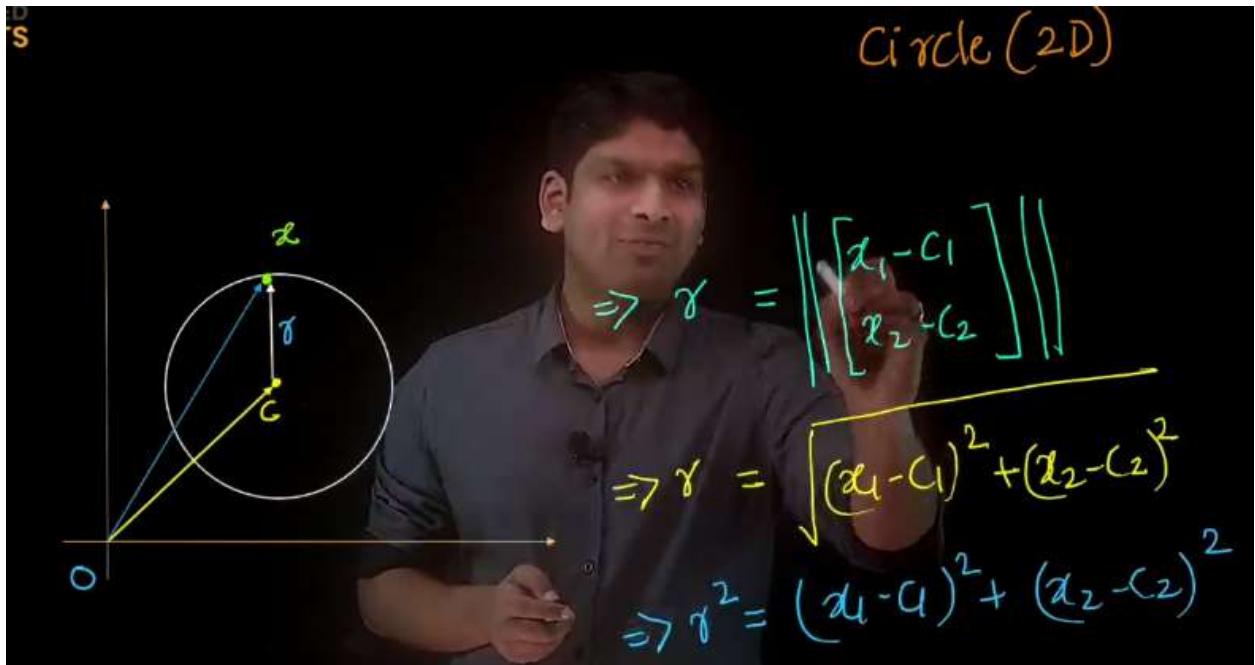
Circle (2D)



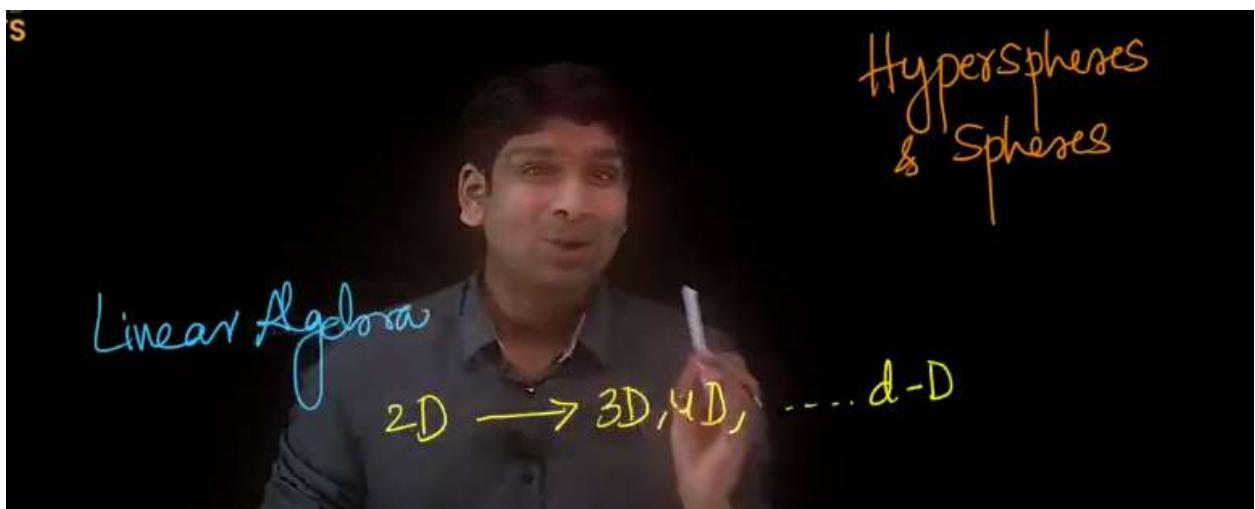
- From the definition of circle the length from center of circle to any point x is always the same and it is r .
- Using algebra we derived $cx=ox-oc$.

At timestamp 5.33





- We already know that the length of cx is radius r. Coordinates of ox are (x_1, x_2) and coordinates of oc are (c_1, c_2) . By doing component wise subtraction of two vectors we get radius r as shown.

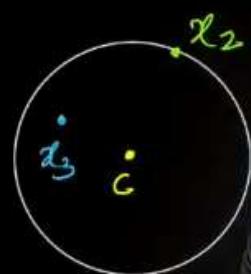


- Whatever we learnt in 2D can be extended to higher dimensions as well.

2.16 Classification using hypersphere

At timestamp 1.04 in video

Classification using Spheres



inside -outside

- Let's understand how we classify or data using spheres and hyperspheres. We assume three points x_1, x_2, x_3 lying outside, on and inside the sphere. Typically we use inside/outside as a mechanism to determine whether a point belongs to which class.

$$\|x_1 - c\| > r$$

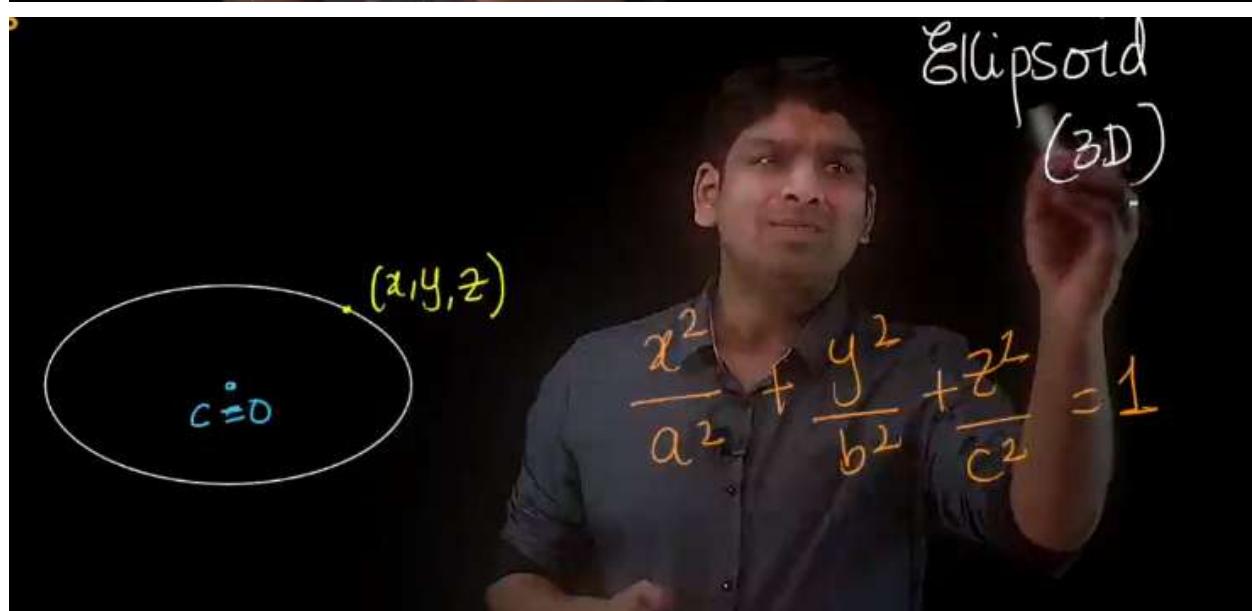
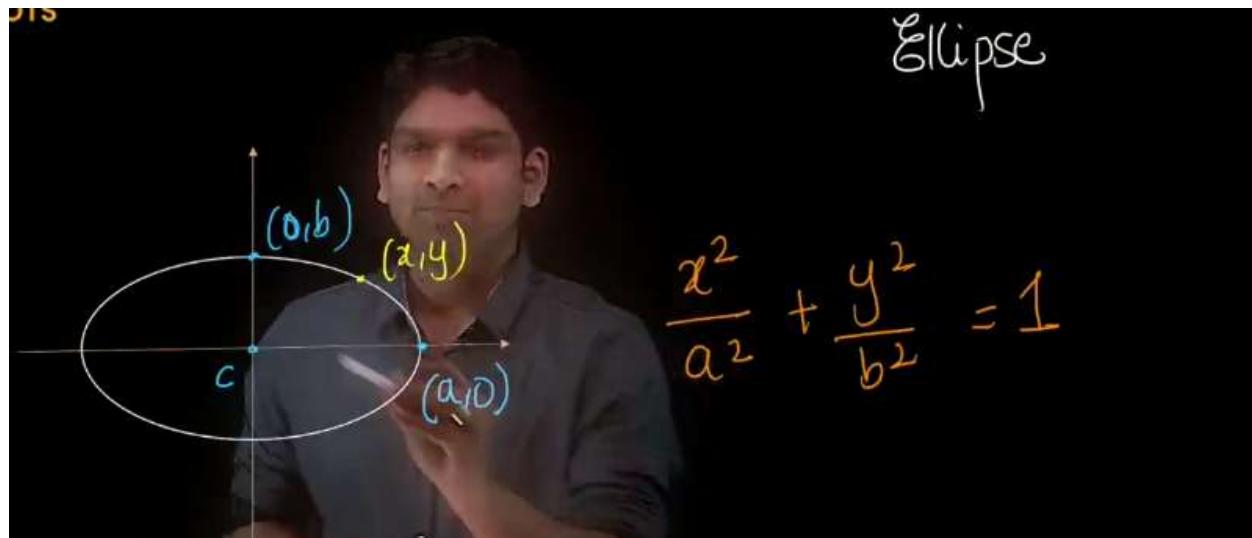
$$\|x_2 - c\| = r$$

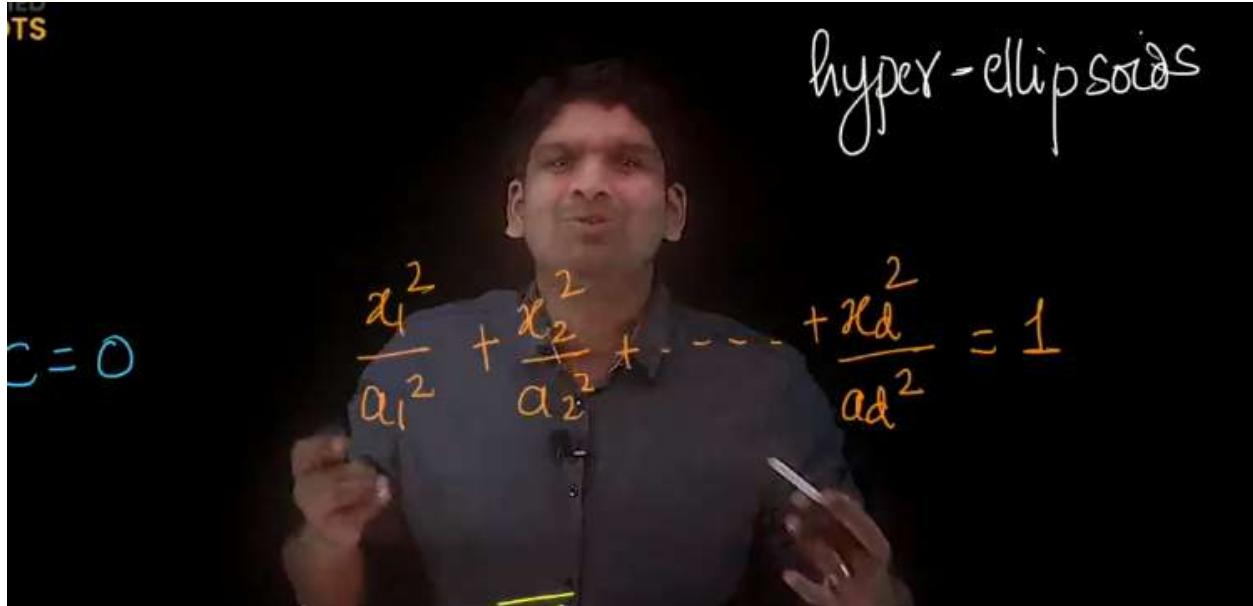
$$\|x_3 - c\| < r$$

- We have to calculate the length of the vector from centre c to to determine where the datapoint lies. We have to check the conditions shown above.

2.17 Ellipse, Ellipsoid, Hyper-Ellipsoid

At timestamp 1.18 in video

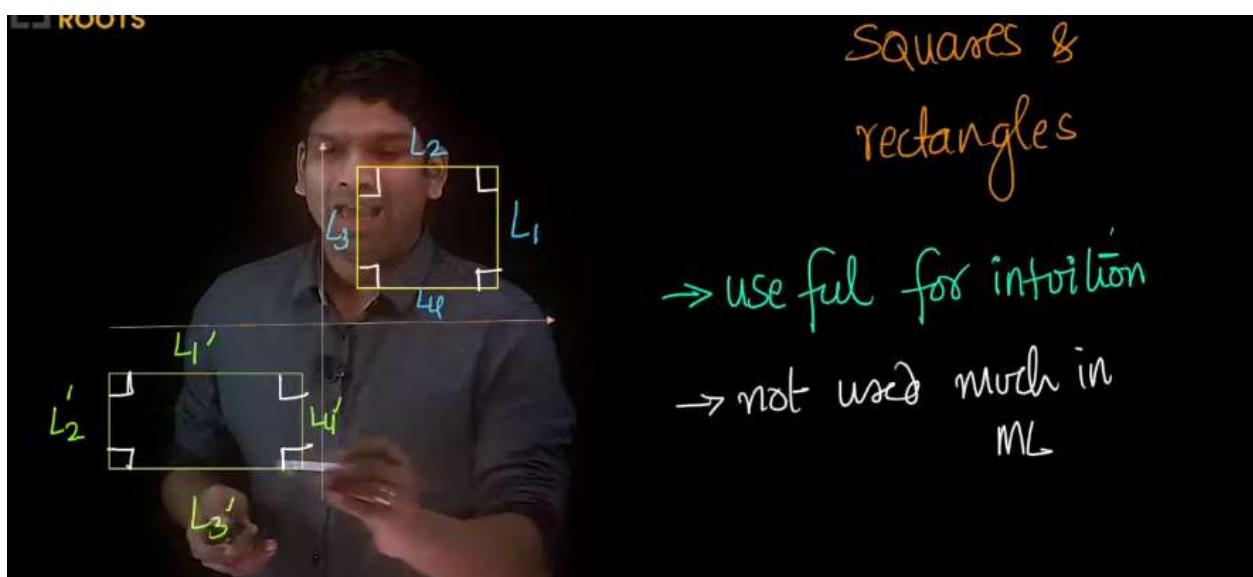




- We represent an ellipse which is centred at origin as shown above ,we call the x axis as major axis and y axis as minor axis here
- We can extend the same concept to 3D and higher dimensions as shown above.

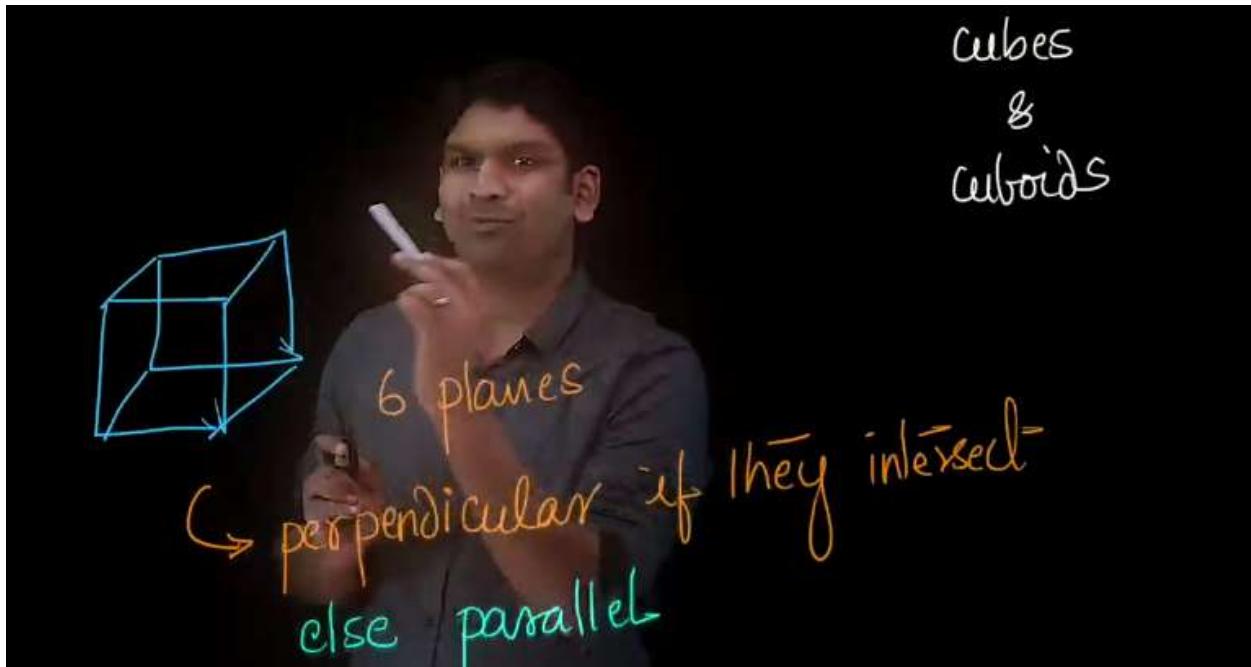
2.18 Squares, Rectangles, Hypercubes and Hyper-cuboids

At timestamp 1.42 in video



- We can represent a square using 4 lines in 2D such that every pair of lines that intersect will be perpendicular and if they don't intersect they have to be parallel to each other .
- Rectangle is similar to a square except that opposite sides will be the same length and all sides need not be of the same length.

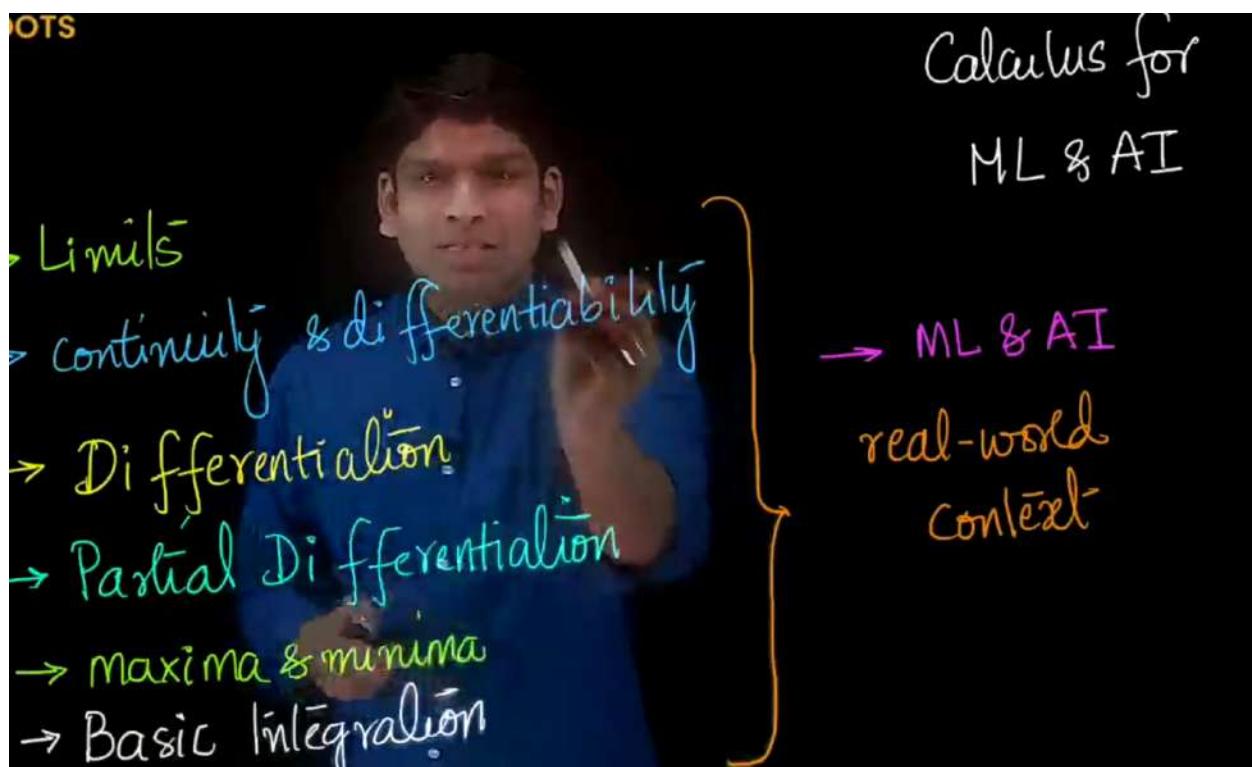
At timestamp 2.19 in video



- We can extend the same concept to higher dimensions as well.
- A point lies inside or outside of the rectangle/square if and only if its x-coordinate lies between the x-coordinate of the given bottom-right and top-left coordinates of the rectangle and y-coordinate lies between the y-coordinate of the given bottom-right and top-left coordinates.

3.1 Introduction

At timestamp in video



- In this chapter we study calculus for machine learning and AI applications in real world context.
- We cover all the above concepts

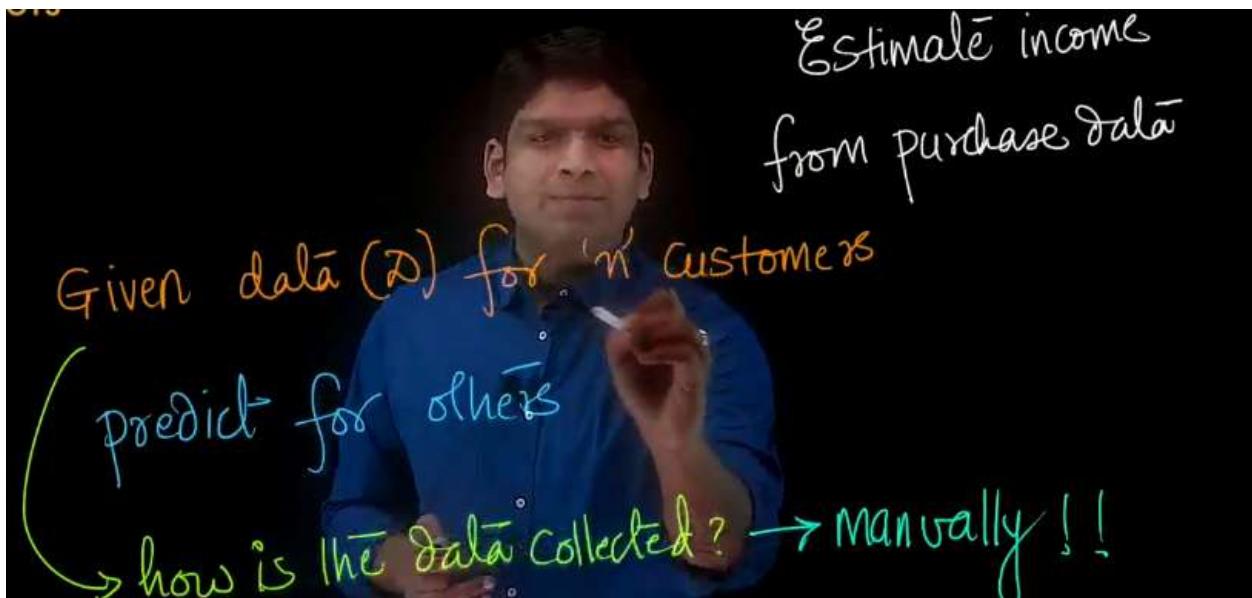
3.2 Real world problem: Estimate income using purchase data

At timestamp 1.24 in video



- In this chapter we estimate income from purchase data of an ecommerce company.

At timestamp 2.24 in video



- We collect customer data manually . Given data of n customers ,we train our model and try to predict the income of others.

At timestamp 4.30

DATA (D)

$x_i \in \mathbb{R}^4$

$y_i \in \mathbb{R}$

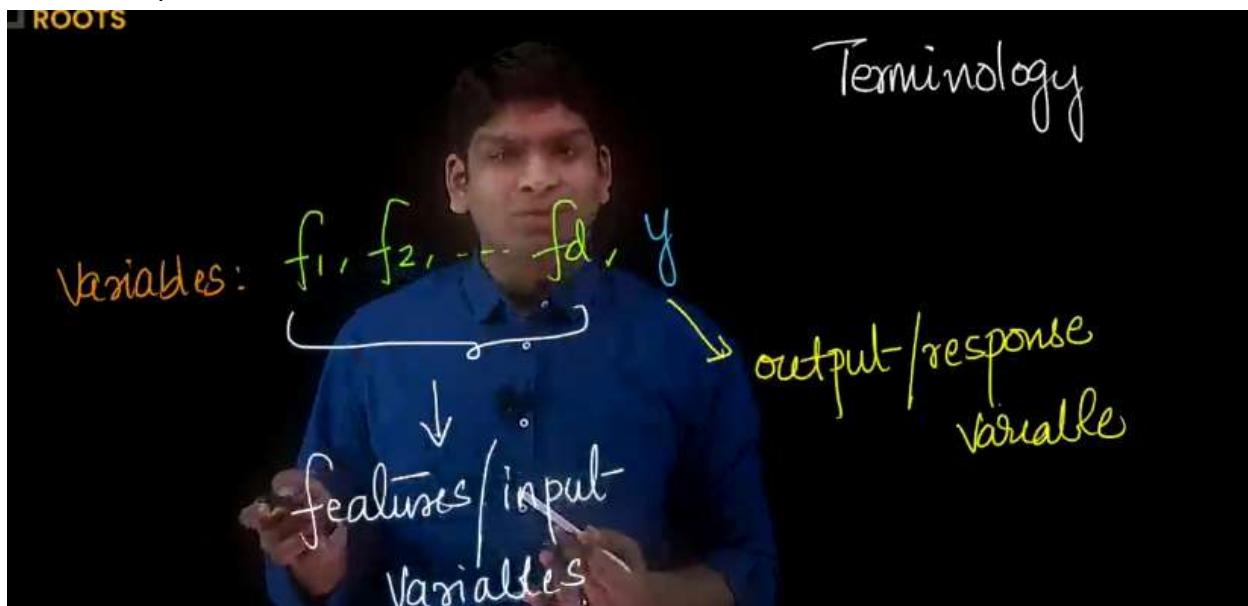
$f_i : i \rightarrow n$

f_1	f_2	f_3	f_4	income
x_1				y_1
x_2				y_2
.				y_3
.				.
x_n				y_n

4-features

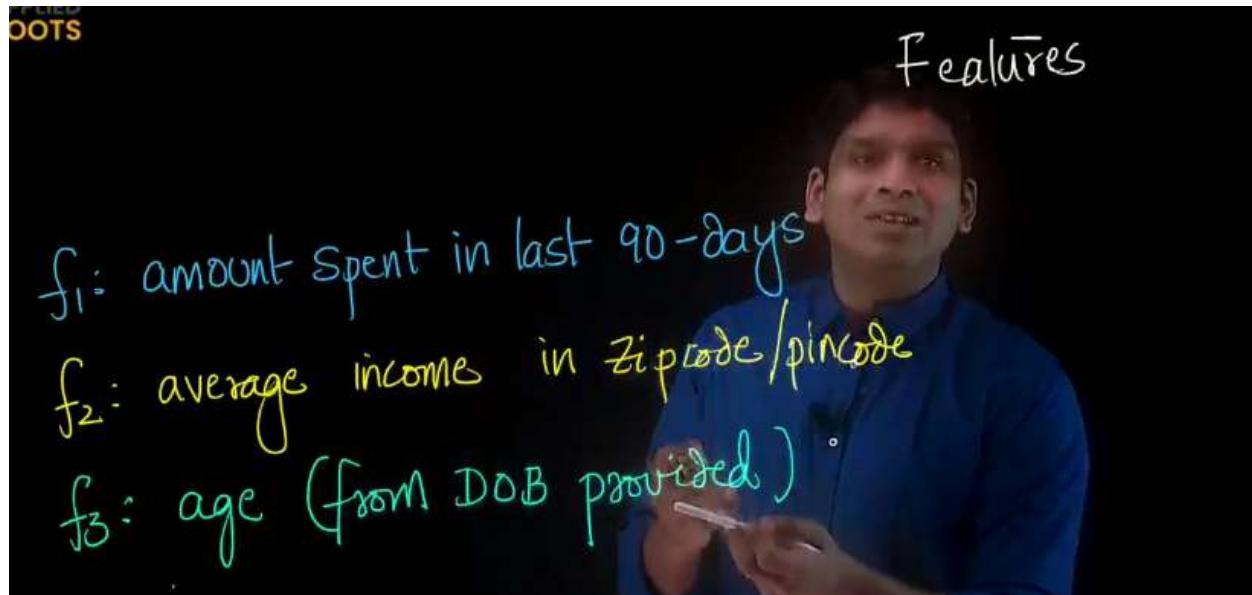
- Let's assume we have four features and we have to predict income based on these features. Each row corresponds to data of one customer.

At timestamp 6.24



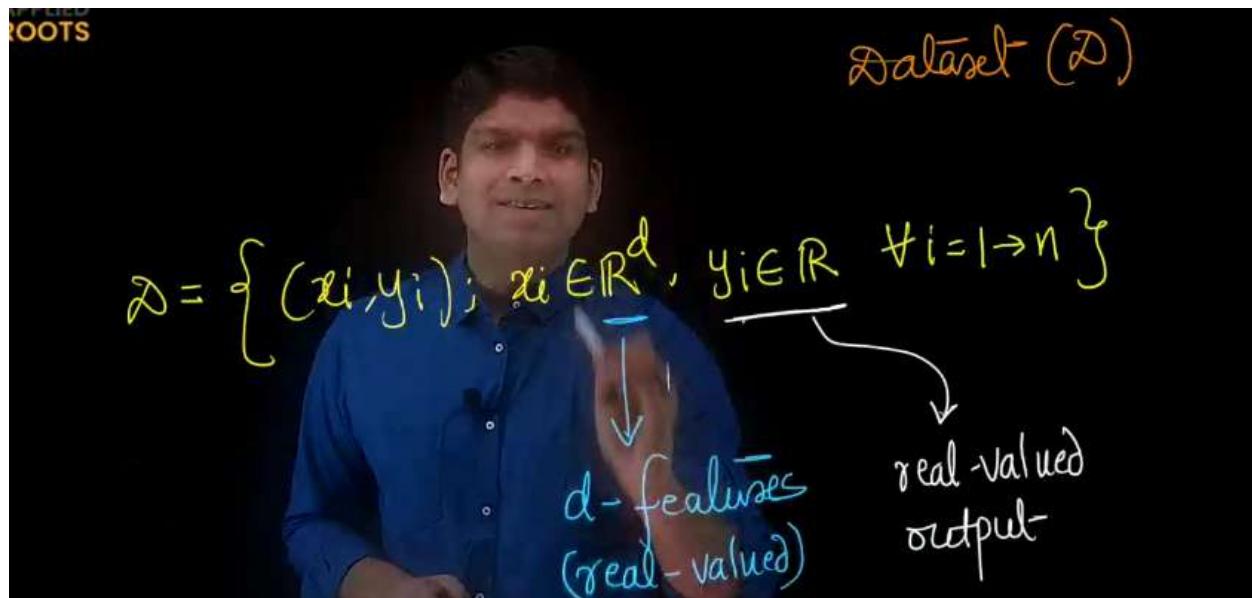
- We have four features we call them as input variables and based on these features we predict output or response variable.
- Once we build a mathematical model we give the model input features and it outputs us the response.

At timestamp 9.53



- The features can be any data that we have collected as shown above

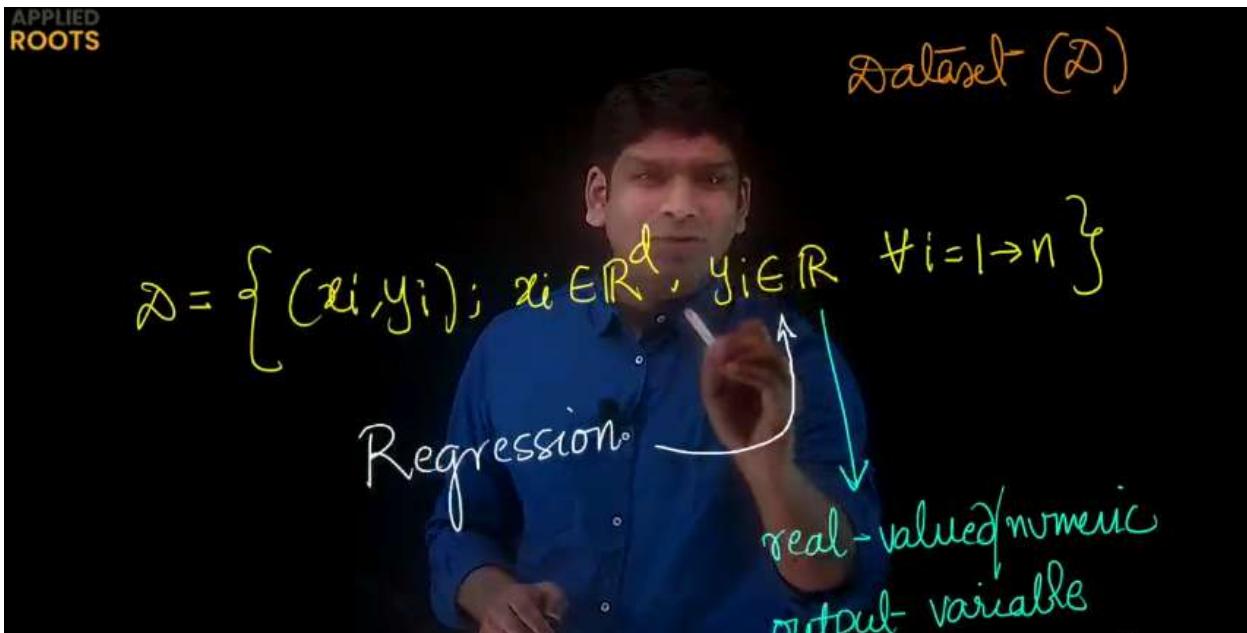
At timestamp 14.22



- In this chapter we consider all features to be numeric going further we discuss categorical features as well.

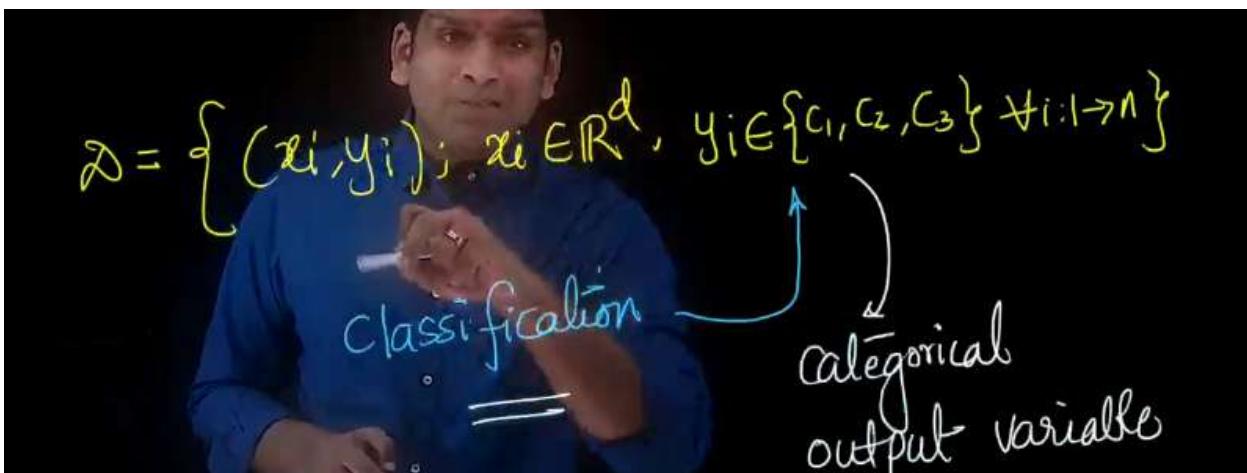
3.3 Visualize the Data

At timestamp 14.53



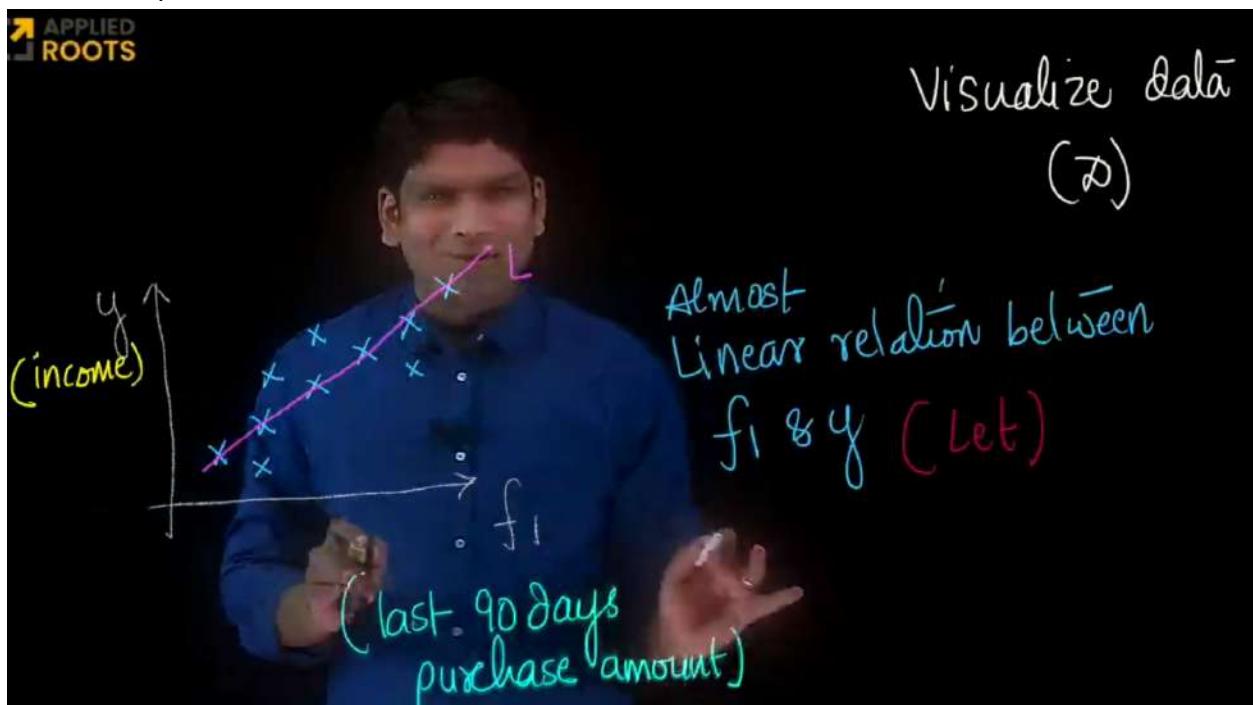
- If the output variable is real valued then the problem at hand is a regression problem .

At timestamp 15.40



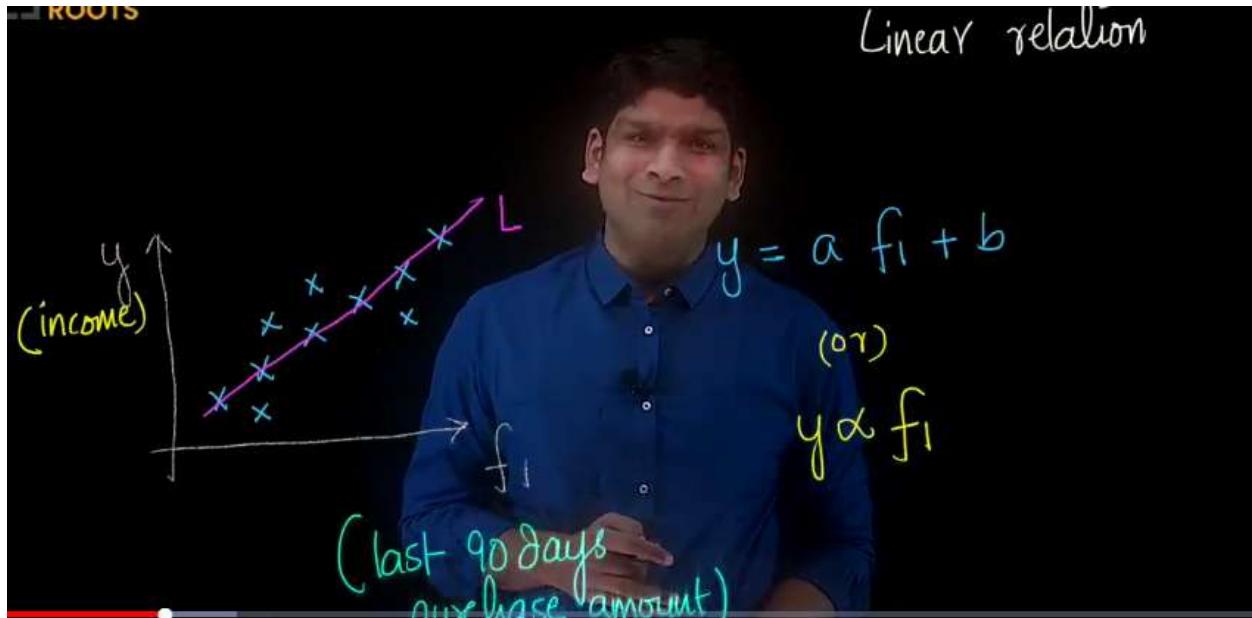
- If the response variable is categorical ,this means the problem at hand is classification problem.(like fish sorting problem)
- If y belongs to more than one class it is called a multiclass classification problem.

At timestamp 2.54 in video

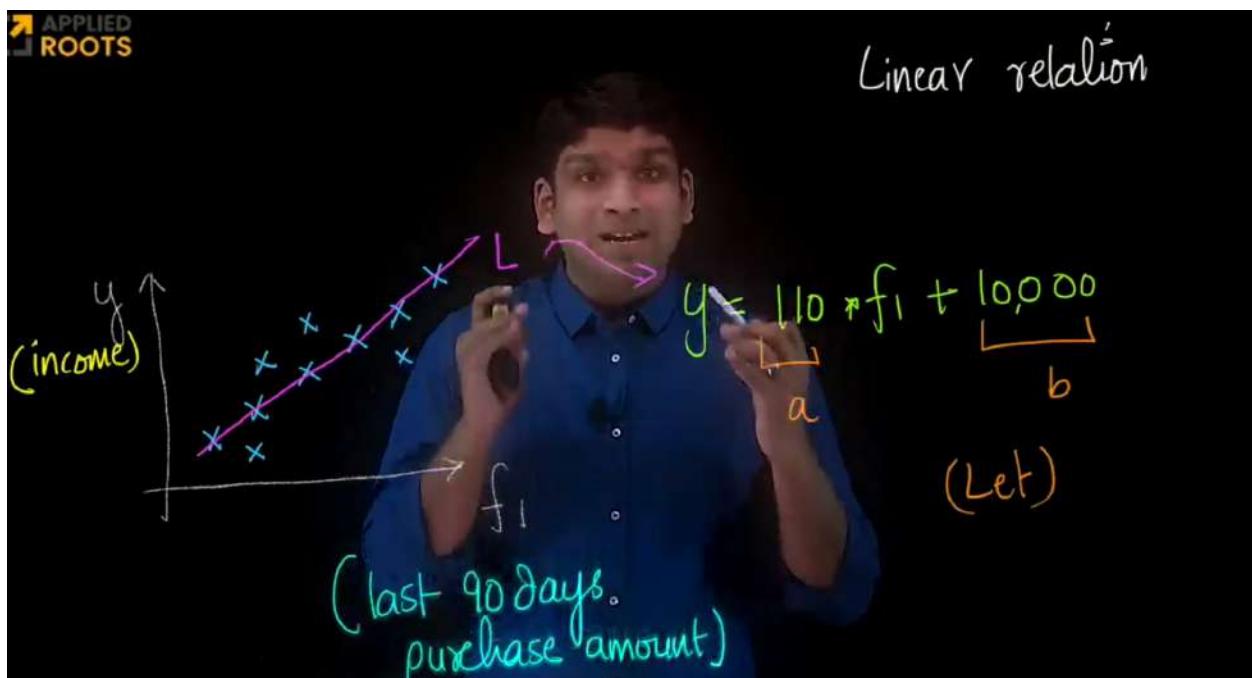


- As we are going to solve the regression problem ,let's try to visualize our data in 2 D,So we have only 1 feature as shown above along the x axis.
- We can clearly see that the feature f and response variable y are having almost a linear relationship.

At timestamp 4.02 in video

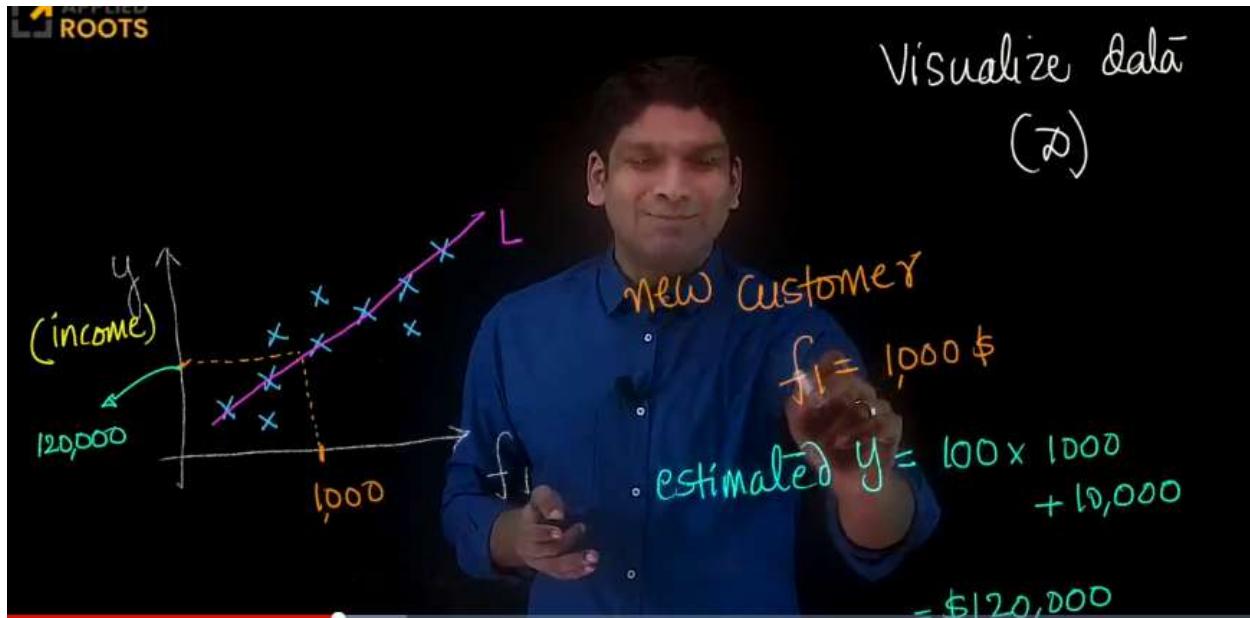


- So we can represent the relation as $y=a.f+b$ (as a line).



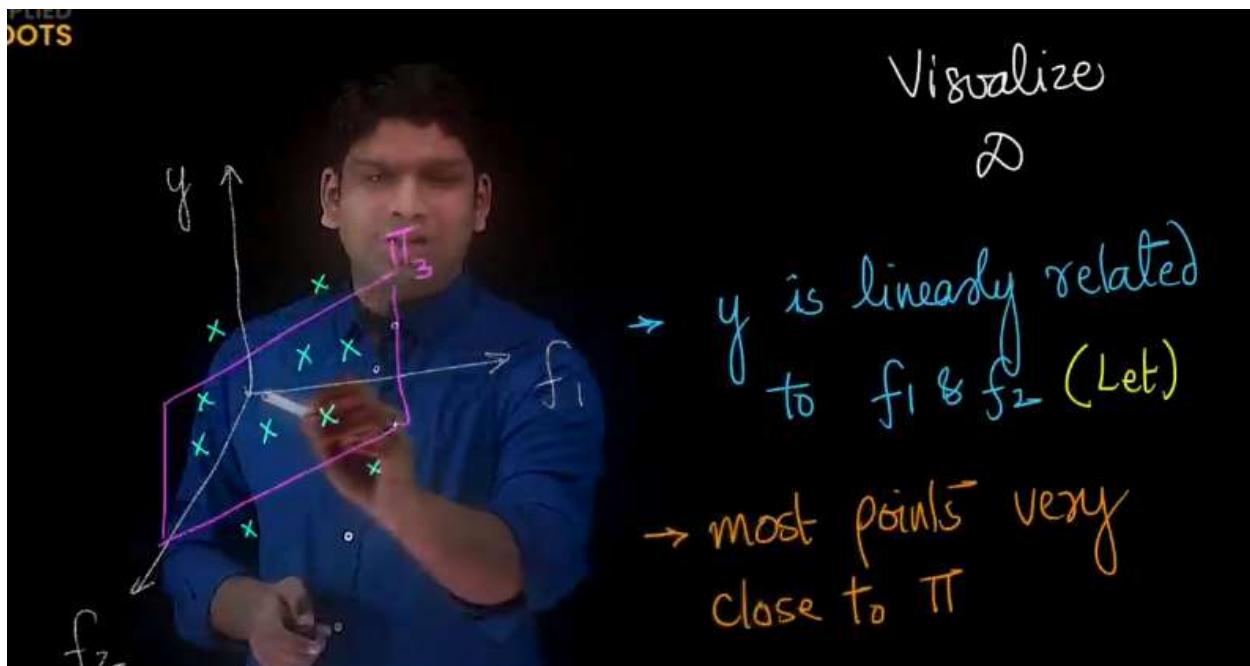
- Let's say we trained and fit the model and we got the above line as our model.
- $y=110.f_1+10000$ is our mathematical model

At timestamp 7.21 in video

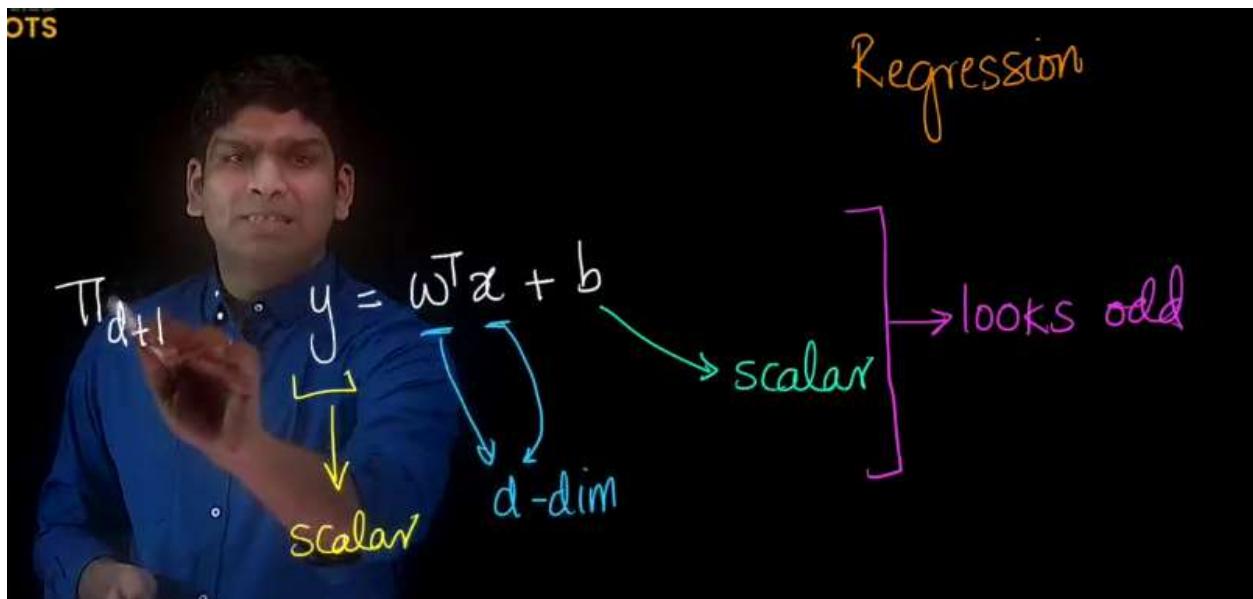
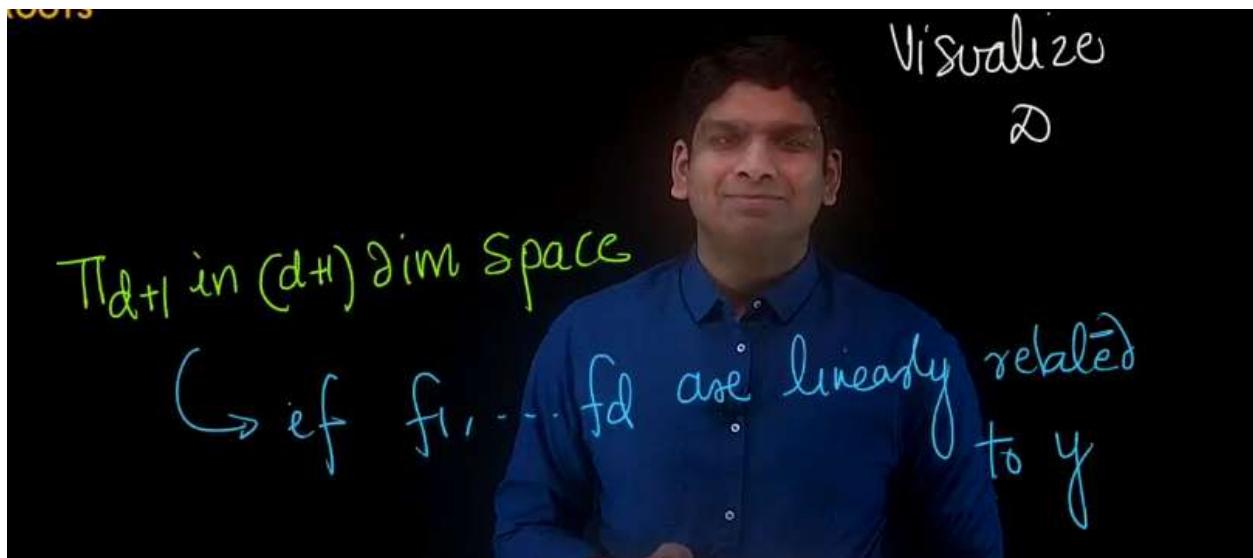


- If we consider the above line as our model then when we have data about a new customer we can just substitute it and we get the income of the new customer.

At timestamp 7.31 in video



- Let's visualize data in 3D ,we have two features f_1, f_2 and our response variable is y .Assume that y is linearly related to f_1 and f_2 then we can fit a plane to our data.We try to find a plane such that most points lie close to the plane or on the plane itself.
- If we have 1 feature we use a line as our model,if we have two features we use a plane.



- Similarly if we have d features which are linearly related to y we can have a hyperplane in $d+1$ dimensional space.
- We cannot use lines and planes if features are not linearly related to y

At timestamp 20.19 in video

Big Question

Given \mathcal{D} , how to find π_{d+1}

w, b ↗ scalar

$d\text{-dim}$ ↗

- The challenge is how do we find the plane . Lets find out.

3.4 Formulation of regression

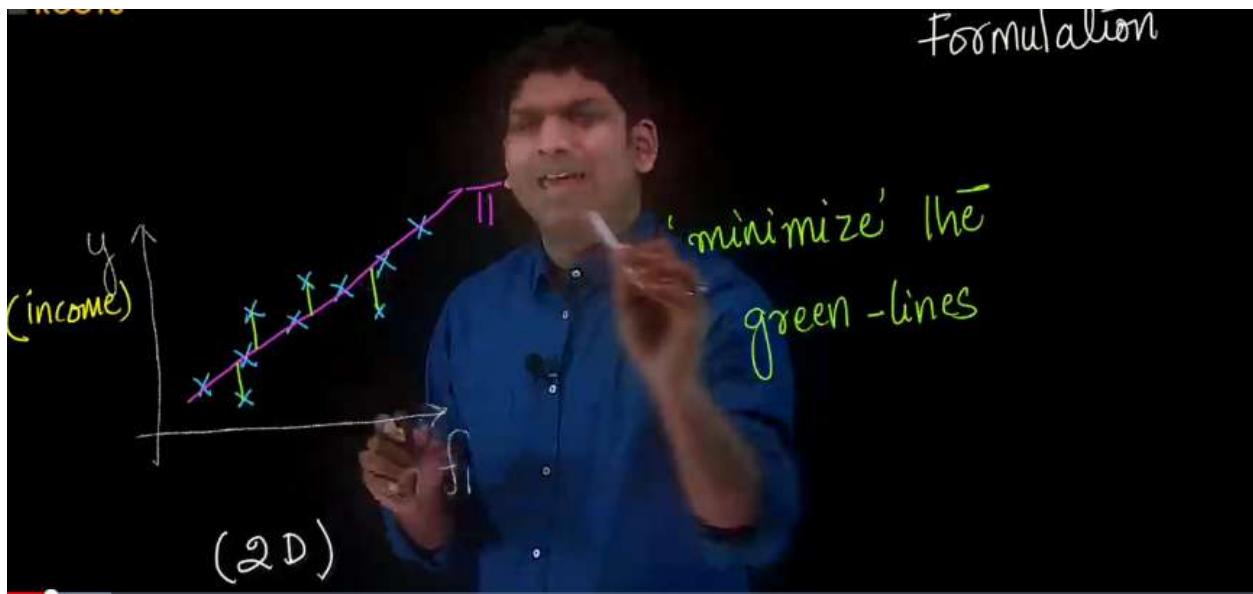
At timestamp 0.11 in video

Formulation of
Regression

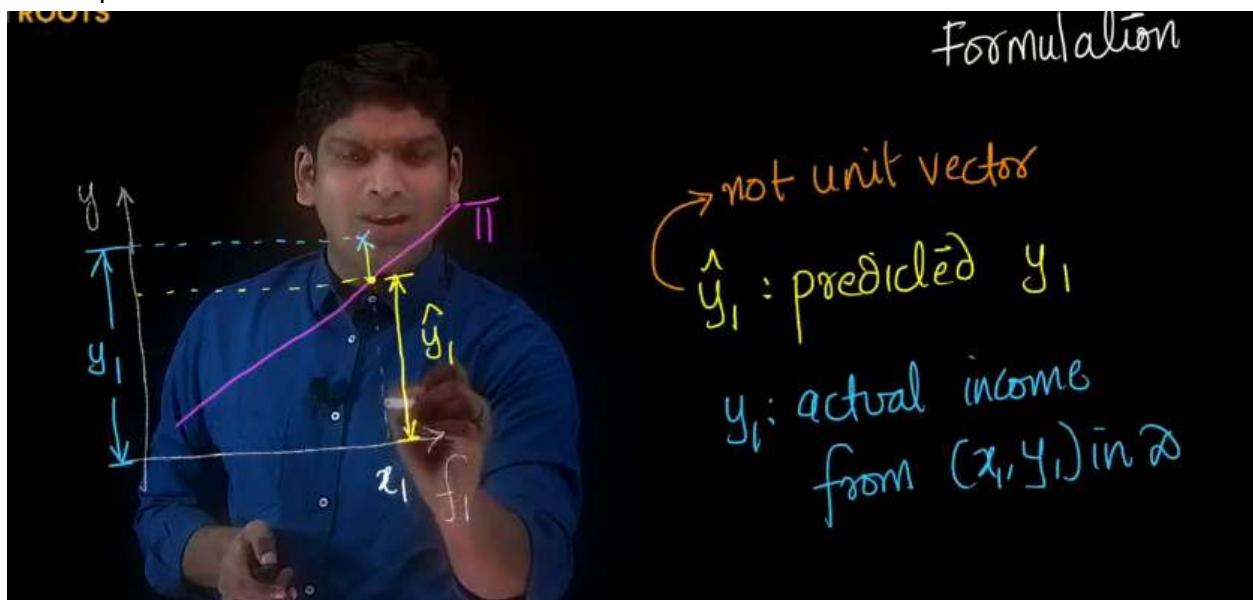
Given 'n' pairs of (x_i, y_i) in \mathcal{D}

find the "best" π_{d+1}

- Given (x, y) n pairs in dataset d, we have to find the best plane. We will try to formulate a regression problem from a mathematical point of view.

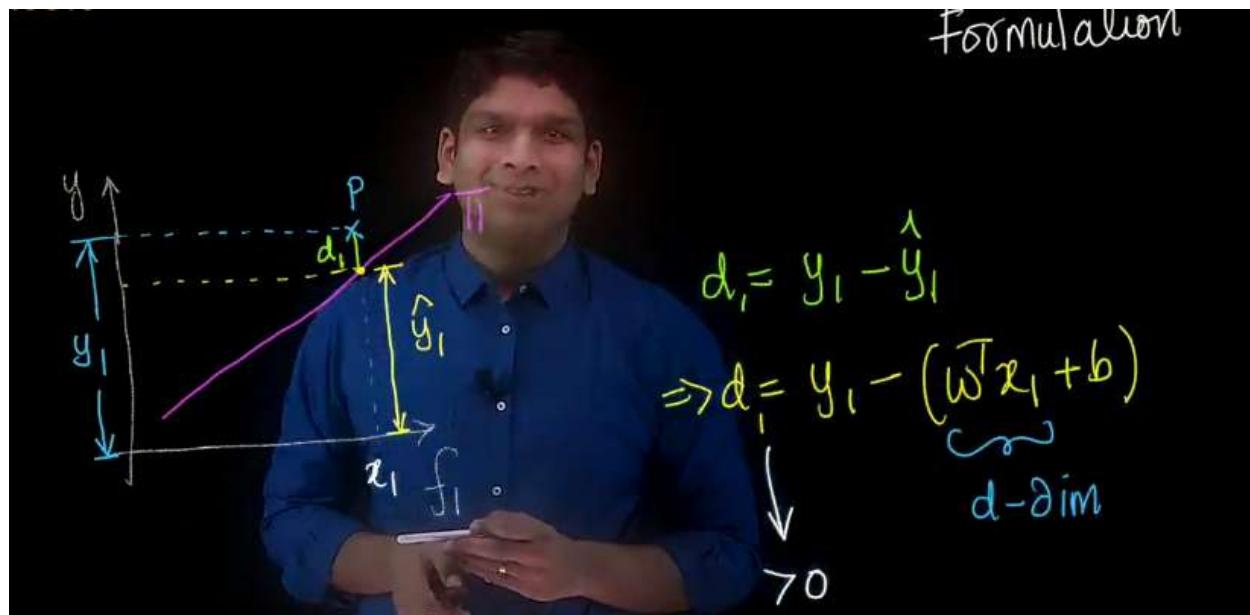


- In order to find the best plane we want to minimize the distance from the point to the plane .



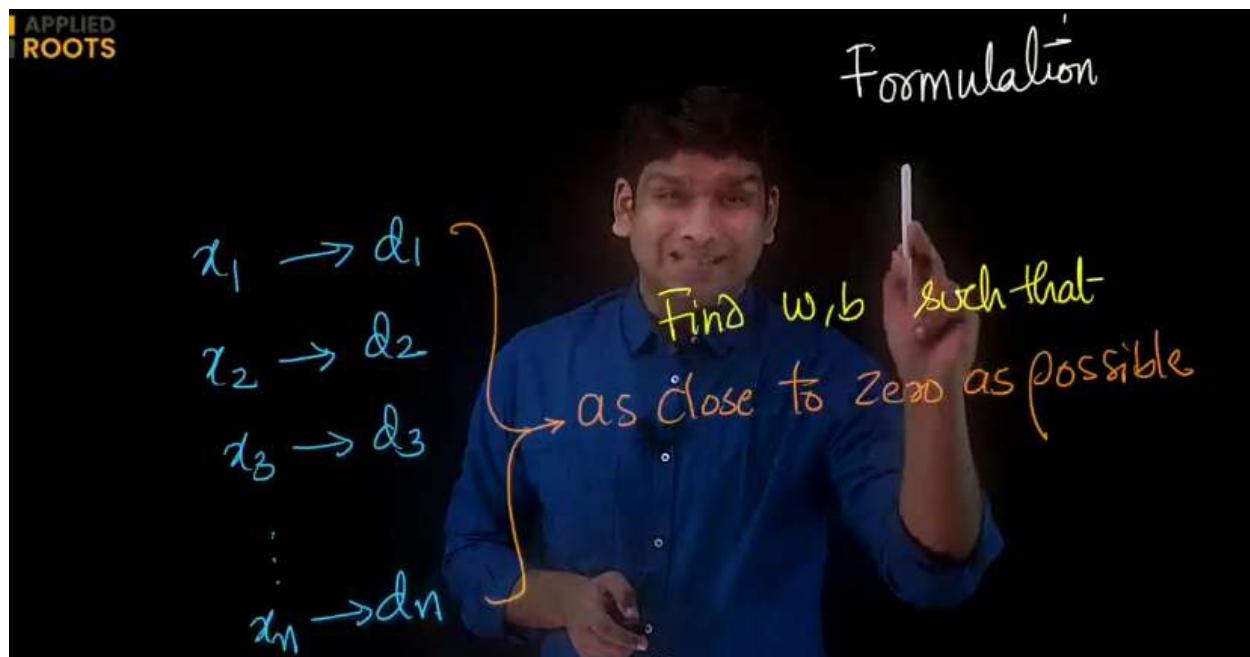
- The predicted value for y is the value which line gives us ,the actual y value is given in the dataset.Our predicted and actual y values may differ based on the line we have as our model

Formulation



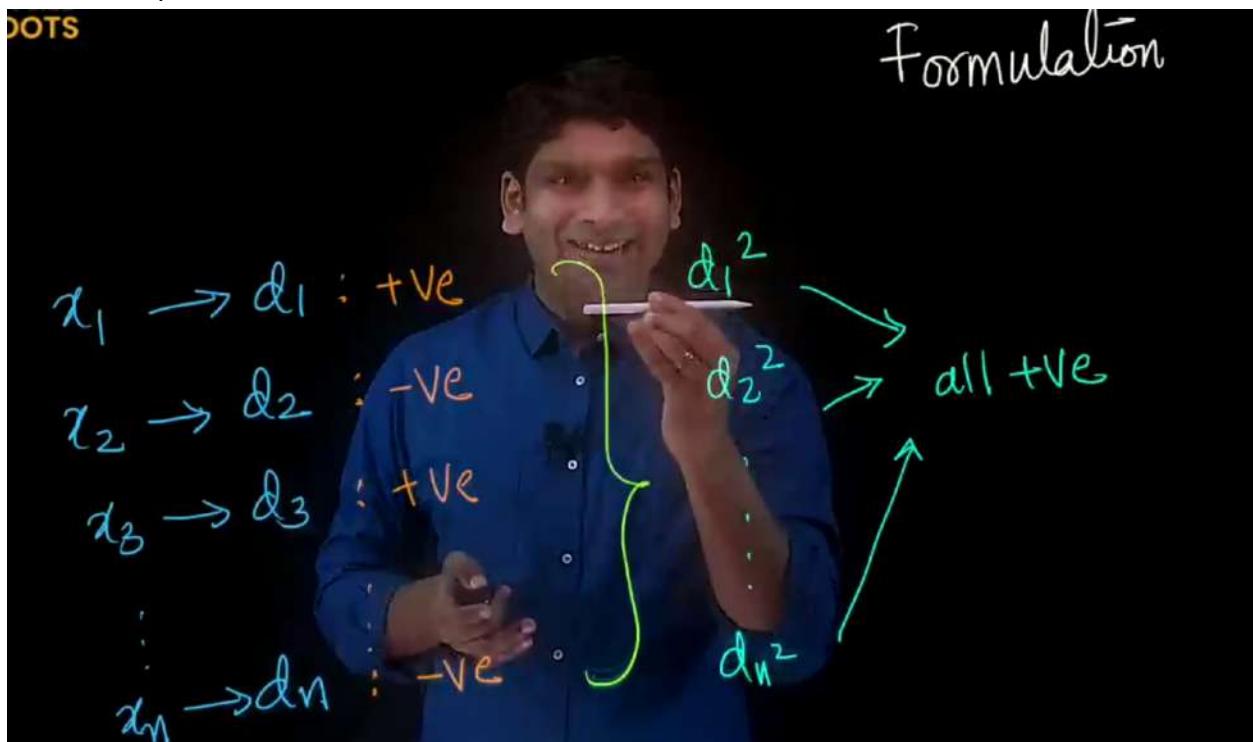
- When we say minimize the distance from point to the line we are actually trying to minimize the value d as shown .(we are minimizing the difference between actual predicted y)

At timestamp 12.36 in video



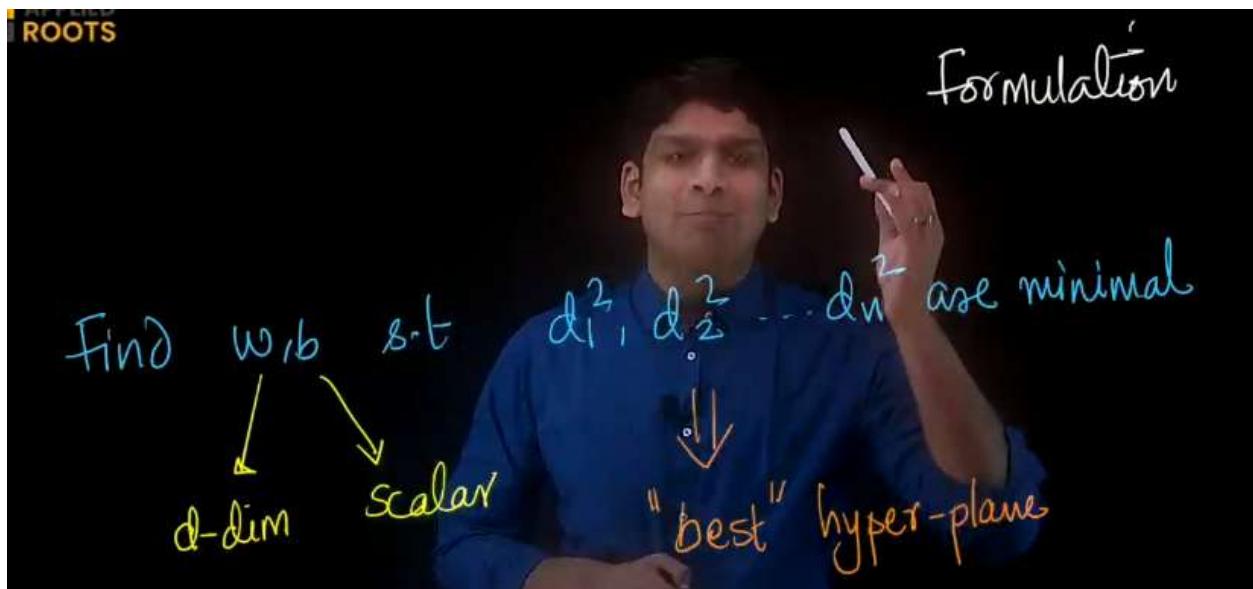
- Our objective here is to find w, b such that the distance (d) from point(x) to the plane should be close to zero.
- We want the sum of distances as close to zero as possible.If distances are closer to zero the points are closer to hyperplane

At timestamp 13.57



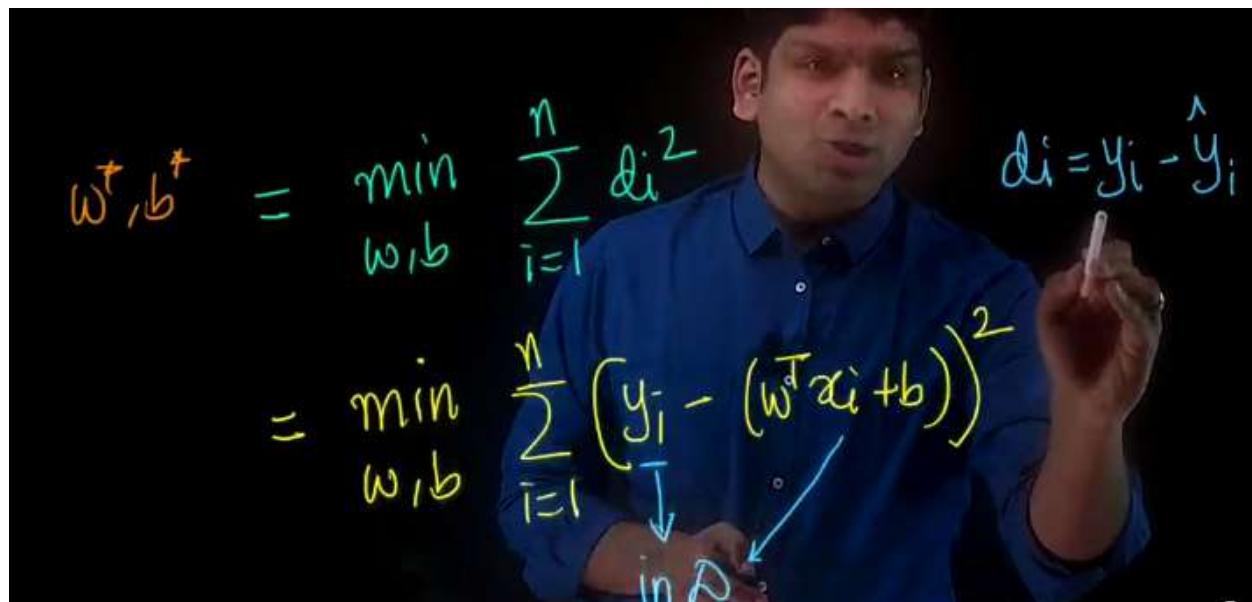
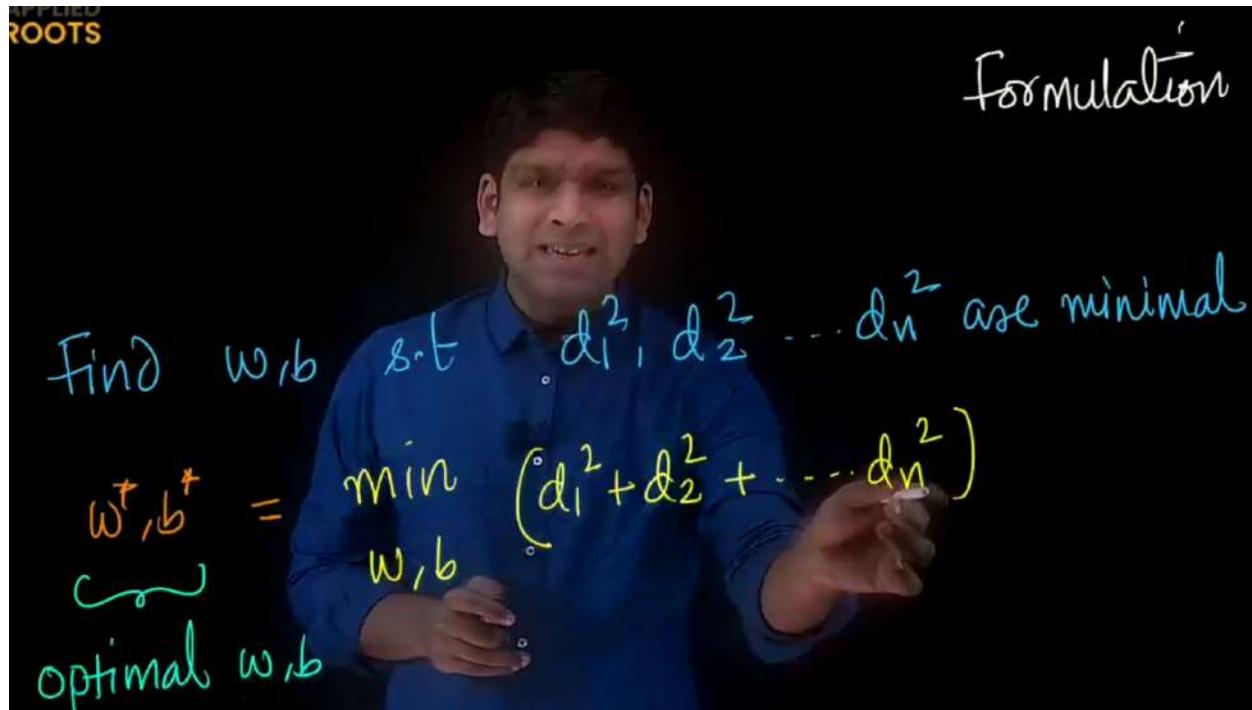
- For points which are either side of the plane the distances will be both positive and negative ,if we sum up the distances it will not be appropriate so we consider the sum of squares of distances.
- We want these sum of squares of distances as close to zero as possible

At timestamp 15.42



- Our whole problem boils down to finding w, b such that we can minimise the squared distances as much as possible.
- By minimising each if d_i^2 we arrive at best hyperplane.

At timestamp 18.30 in video



- We have to find optimal w, b such that the sum of squared distances will be minimised.
- Here we are minimise loss so that we can obtain w, b

At timestamp 22.35 in video

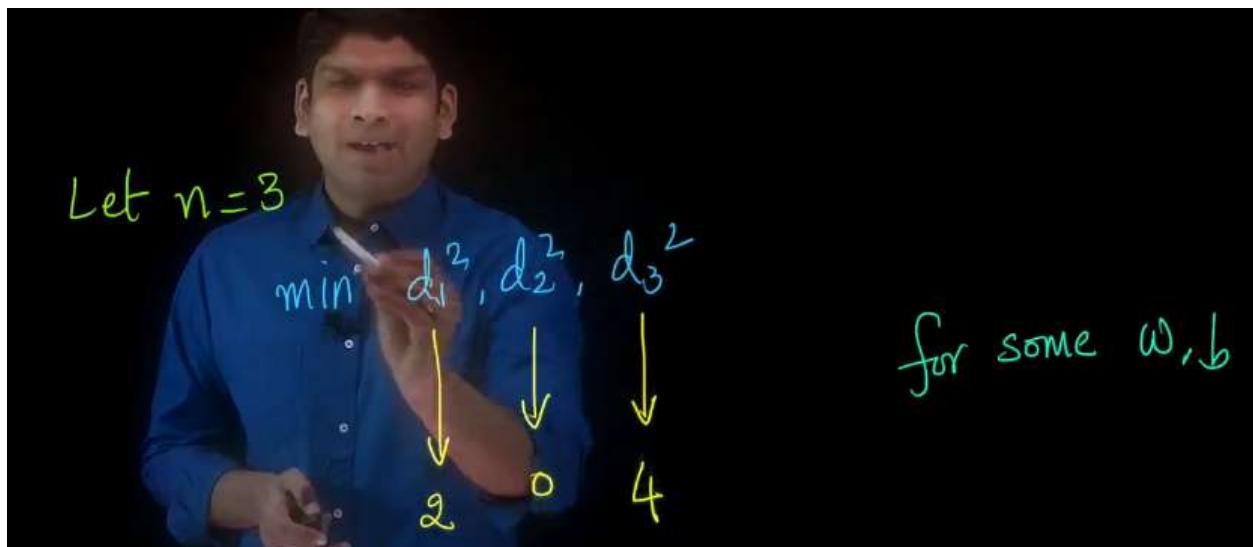


- To solve this regression problem we use maxima minima from calculus.

At timestamp in video

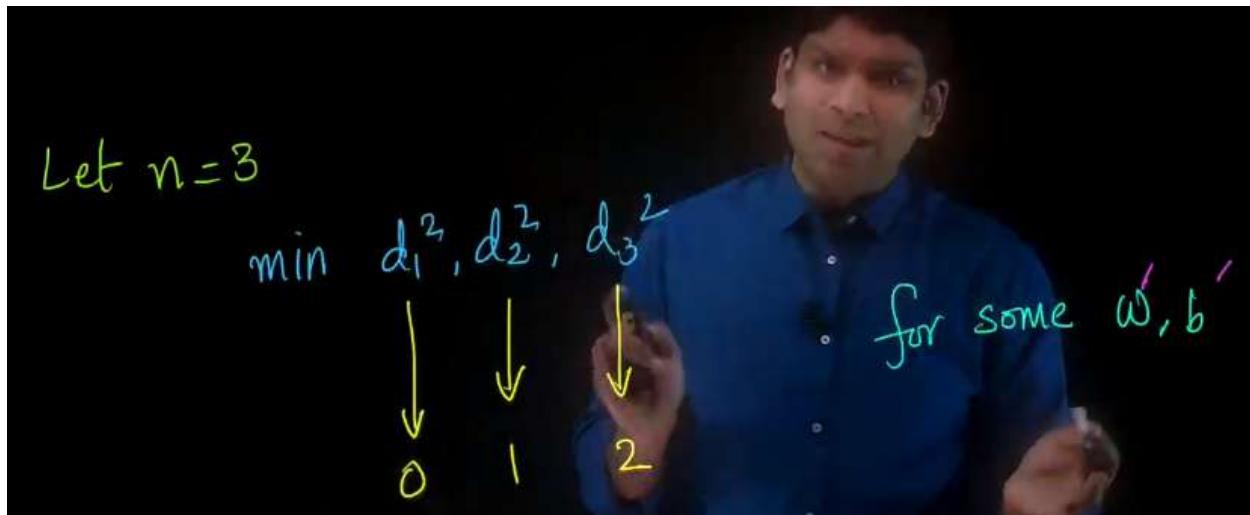
3.5 Minimisation of multiple values

At timestamp 2.08 in video



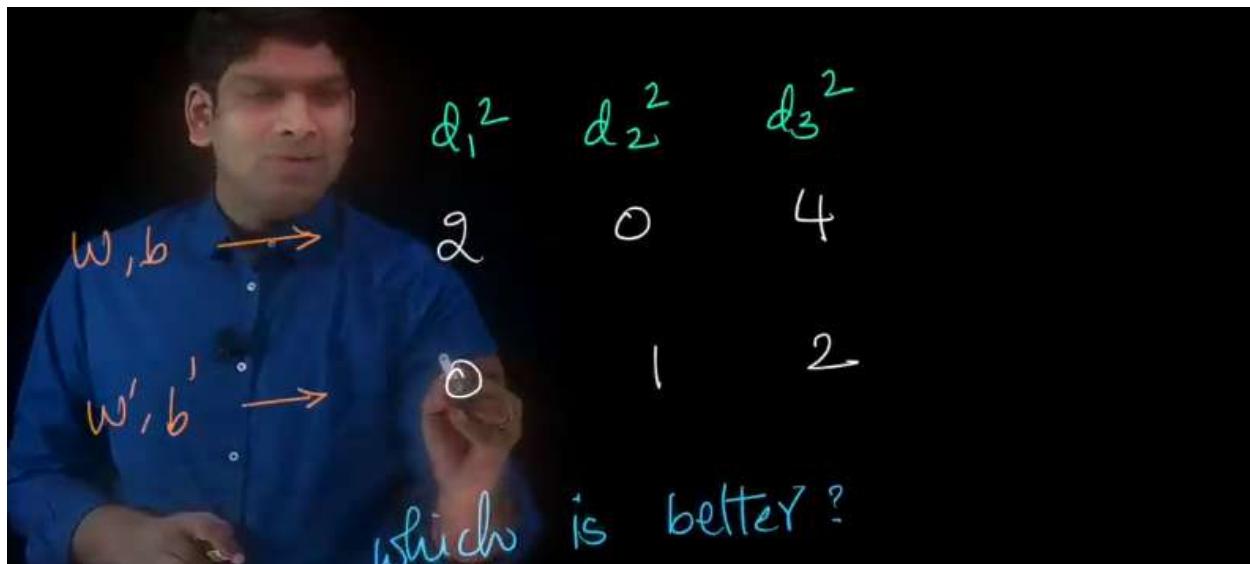
- Imagine we have 3 points and corresponding to each point we have squared distances as shown above. Lets assume a plane with some values of w,b as our model.(remember we can determine a plane if we know and b)

At timestamp 2.55 in video



- For some other plane with w', b' the squared distances are as shown above
- As the plane changes the distances change.

At timestamp 3.40 in video



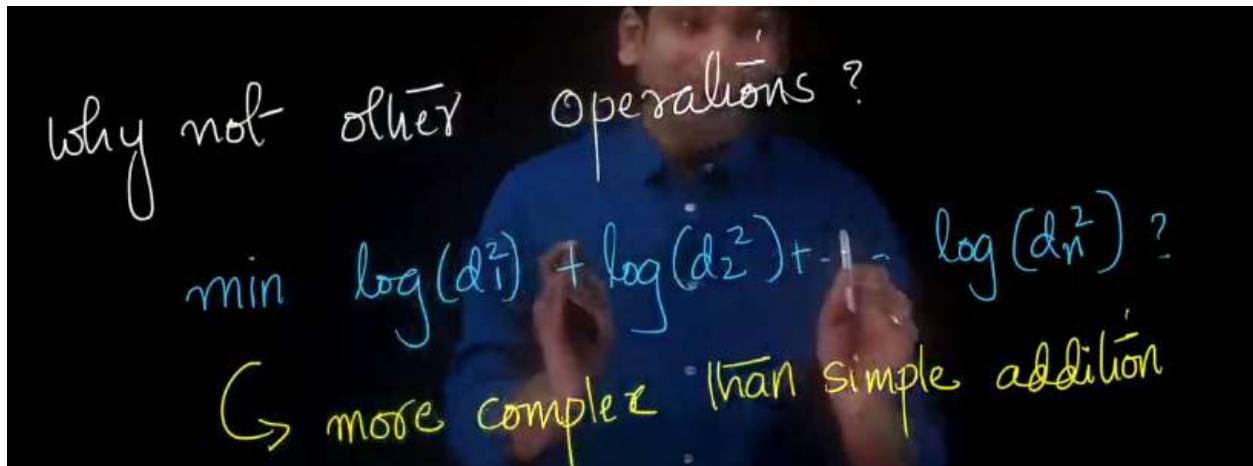
- We choose the plane which minimises total sum of squared distances.

$\min d_1^2, d_2^2, \dots, d_n^2$
 $\min d_1^2 + d_2^2 + \dots + d_n^2 \quad (\text{why not?})$
 $= \min \prod_{i=1}^n d_i^2$

$n, b \rightarrow d_1^2 \quad d_2^2 \quad d_3^2$
 $w', b' \rightarrow 2 \quad 0 \quad 4$
 $1 \quad 2$
 $\sum d_i^2 \quad \prod d_i^2$
 $6 \quad 0$
 $3 \quad 0$

- Minimising each squared distance is equivalent to minimising the sum of squared distances and we cannot use the product of the squared distances even if one distance becomes zero(actual and predicted y are same) total product becomes 0.

At timestamp 19.47 in video

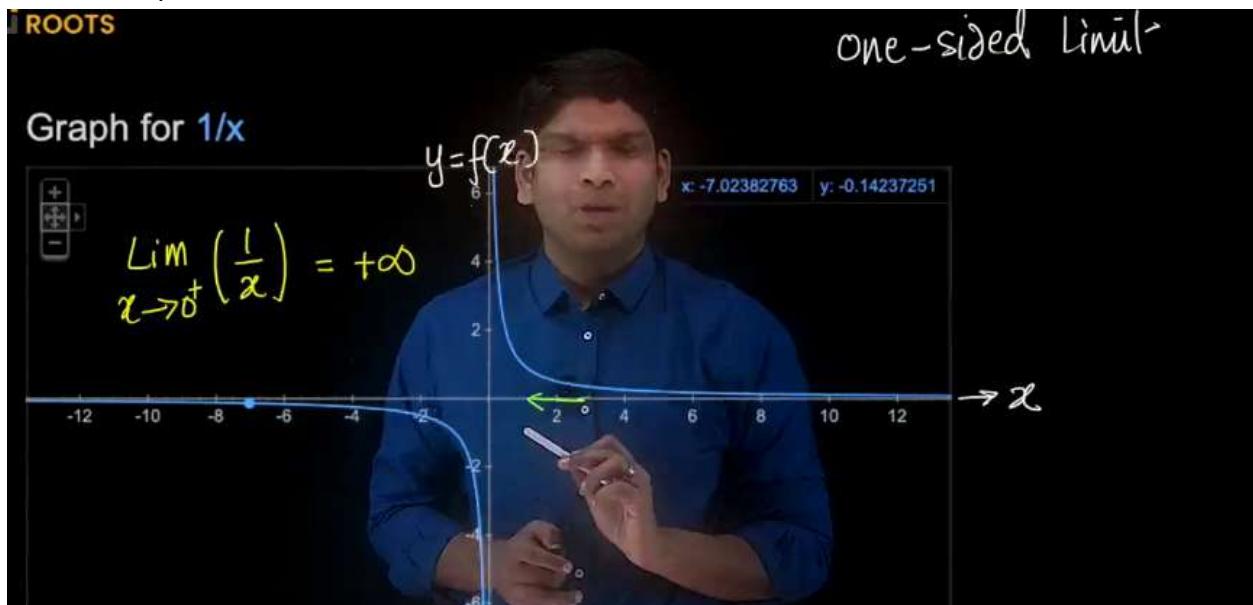


- We can use other operations than addition such as log. We don't want to increase the complexity so we use addition.

3.6 Limits, Range and Domain of a function

In this chapter we learn calculus ,limits and range for most popularly used functions in AI/ML

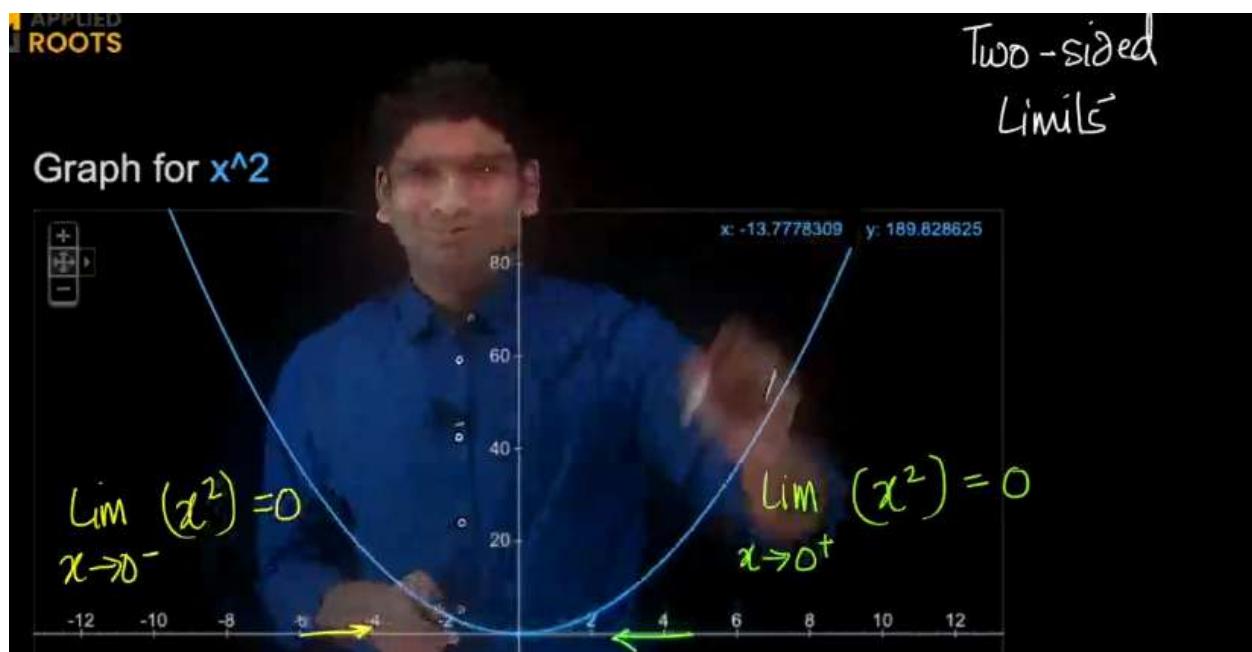
At timestamp 2.15 in video



- Consider function $f(x) = 1/x$ and lets understand the concept of one sided limits

- As you are coming closer and closer towards 0 from the positive side of the x axis, the curve is moving towards infinity. The same fact can be represented mathematically using limits as shown above and it is often referred to as the right/positive sided limit.
- As you are moving closer from the negative side of the x axis towards 0, the curve is tending towards minus infinity. The same fact can be represented mathematically using limits as shown above and it is often referred to as the left/negative sided limit.

At timestamp 5.49 in video



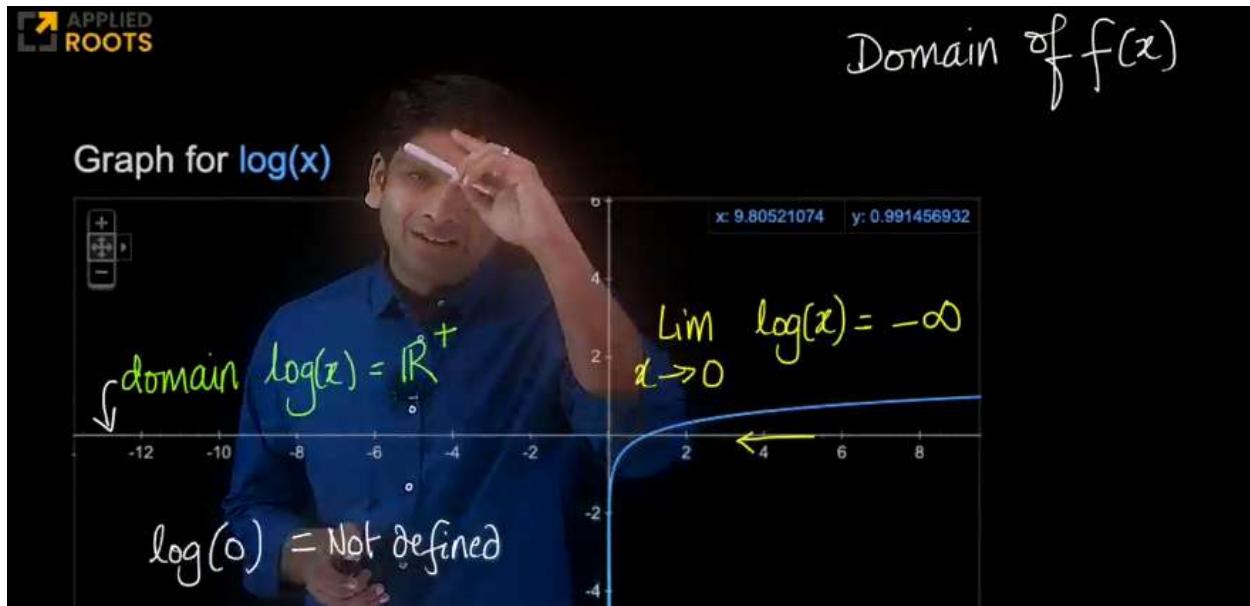
- The above curve is a parabola. There is a similar concept called two sided limit.
- As you are moving closer and closer towards 0 from the positive side of the x axis, the curve is moving 0 so the right hand limit is 0 similarly the left hand limit is also 0.
- We represent the above behaviour mathematically as shown below.

The teacher is holding a whiteboard marker and pointing towards the equation on the board. The equation is written in green ink:

$$\lim_{x \rightarrow 0^+} (x^2) = \lim_{x \rightarrow 0^-} (x^2) = 0 = \lim_{x \rightarrow 0} (x^2)$$

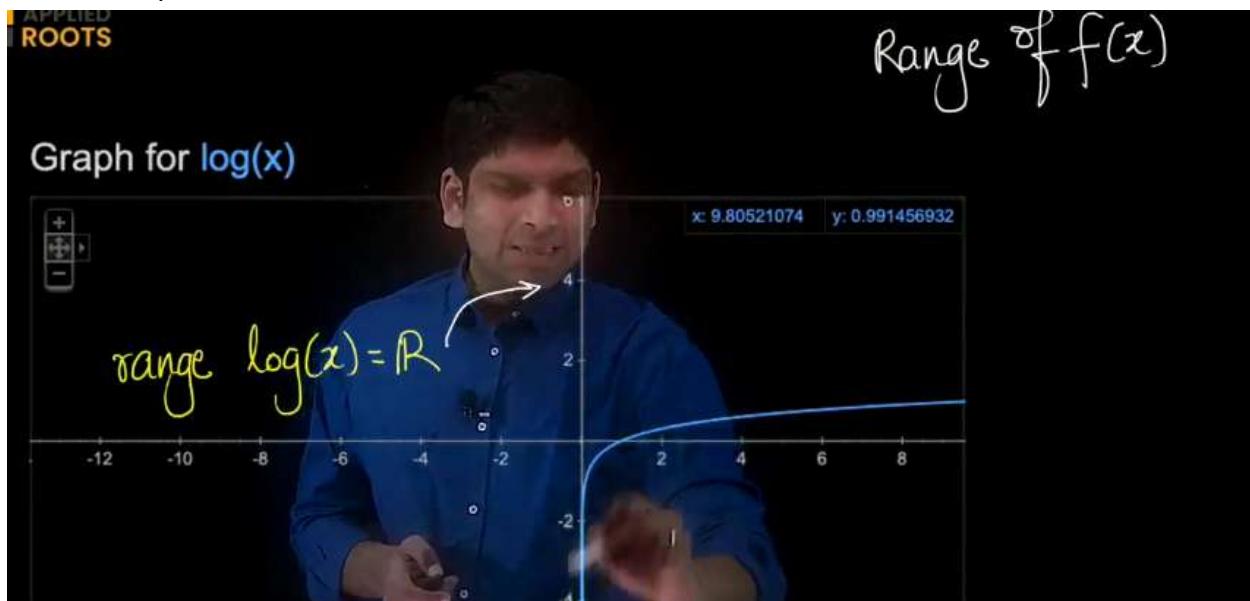
A callout box in the top right corner contains the text "Two-sided limits".

At timestamp 9.05 in video



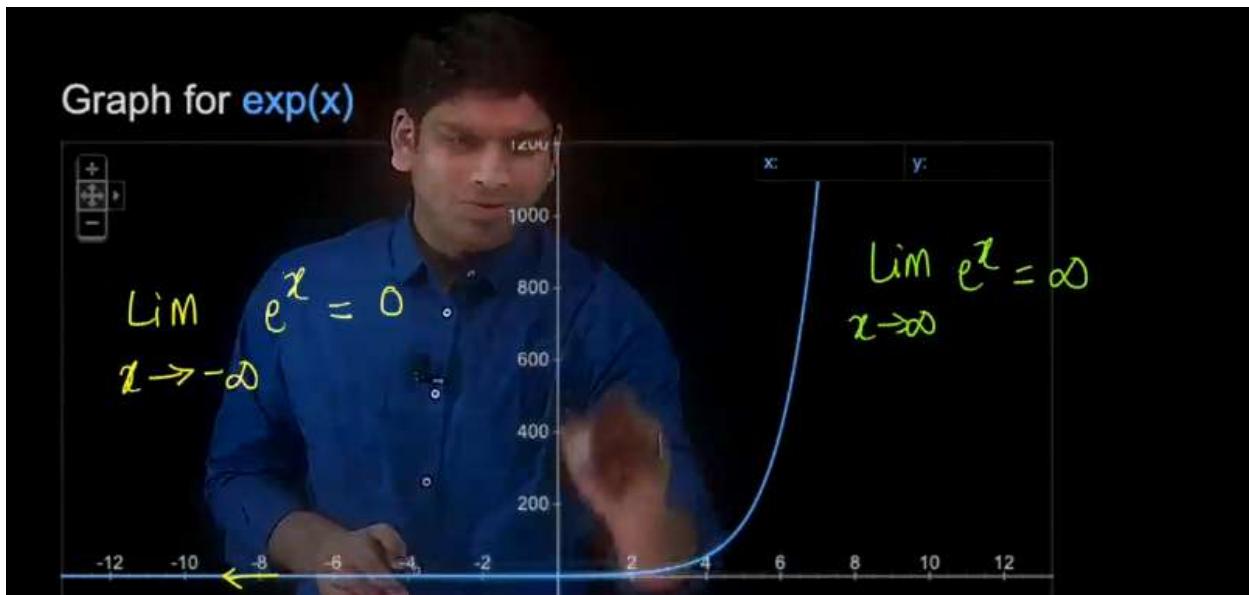
- The above is $\log(x)$, we use this function a lot in machine learning. We know that \log of 0 and -ve numbers is not defined, \log is defined only for positive numbers. The set of possible values which a function can take is called as domain and the domain for $\log(x)$ is positive real numbers.
- From the plot we can see that the right handed limit when we move closer to 0 from the positive x axis the function is tending towards - infinity.

At timestamp 11.41 in video

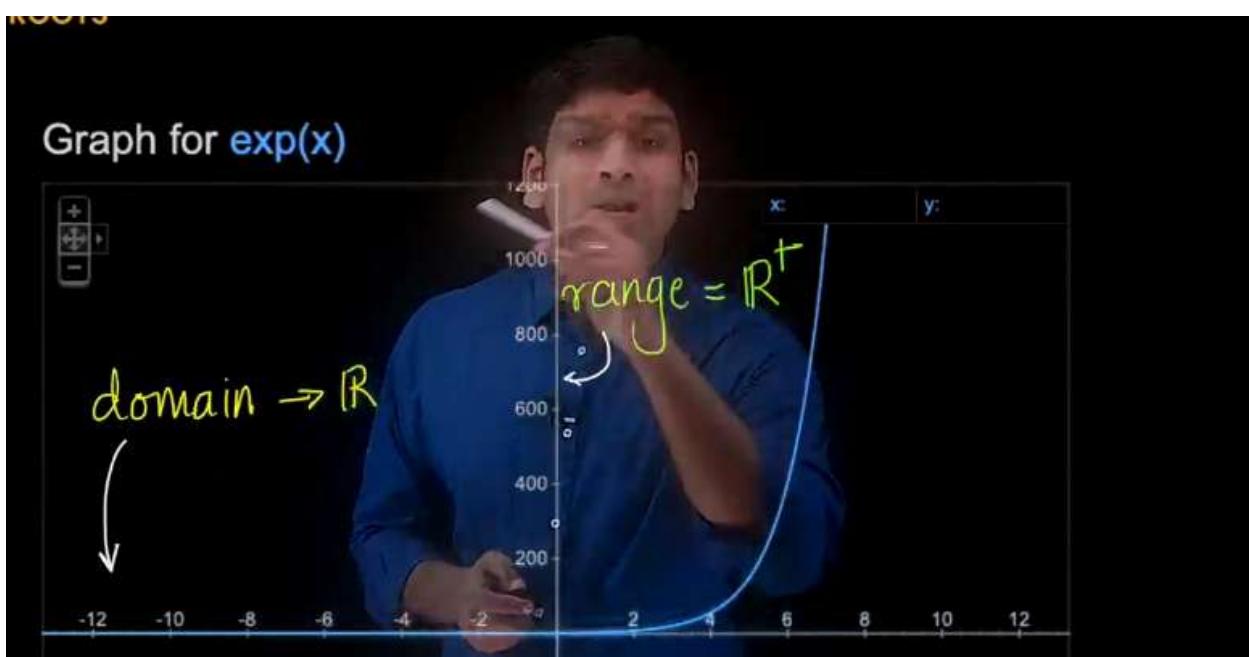


- The values taken by the function on y axis is called range, $\log(x)$ can be positive, 0 and negative. So the range of function $y = \log(x)$ is all real numbers.

At timestamp 12.40 in video

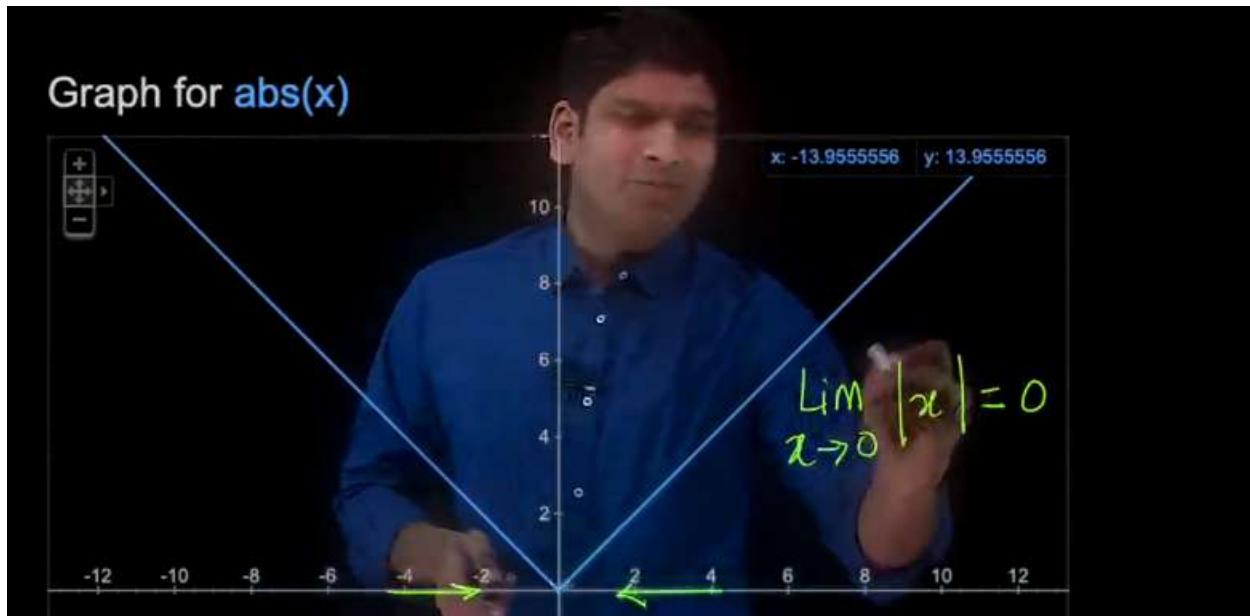


- Let's consider the function e^x , as we move closer and closer to $-\infty$ the function tends towards 0. It can be mathematically represented using limits as shown above.
- Similarly as we move towards $+\infty$ the function also moves towards $+\infty$.

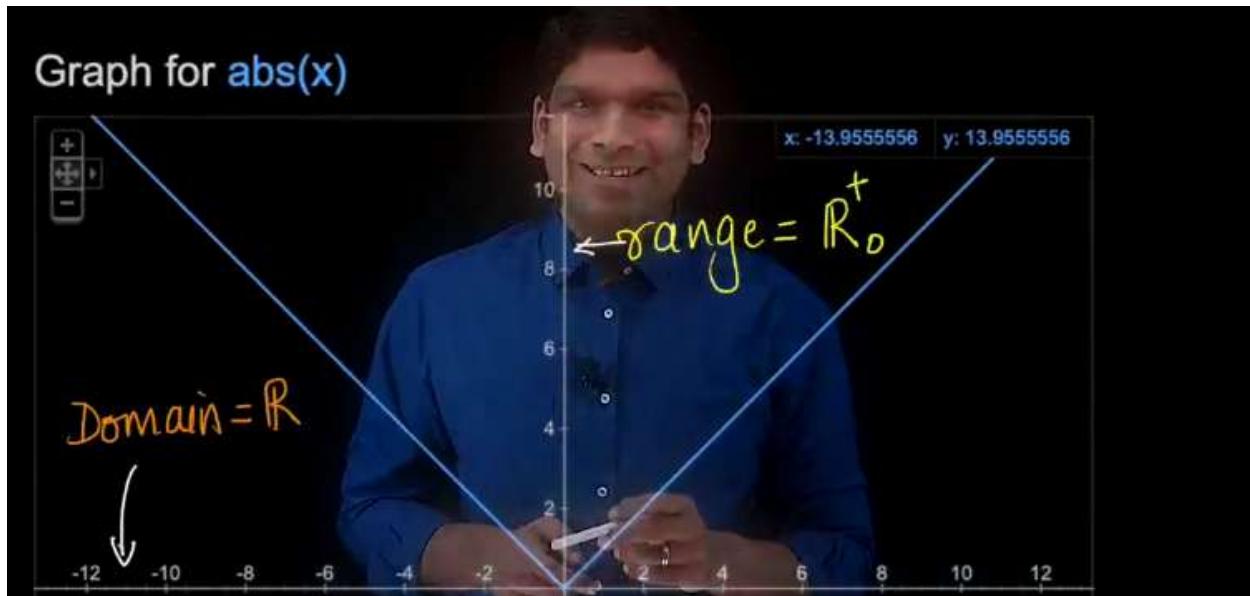


- The domain of the function is set of all real numbers and the range is all positive real numbers. (\mathbb{R}^+ means all positive real numbers excluding 0)

At timestamp 16.01 in video

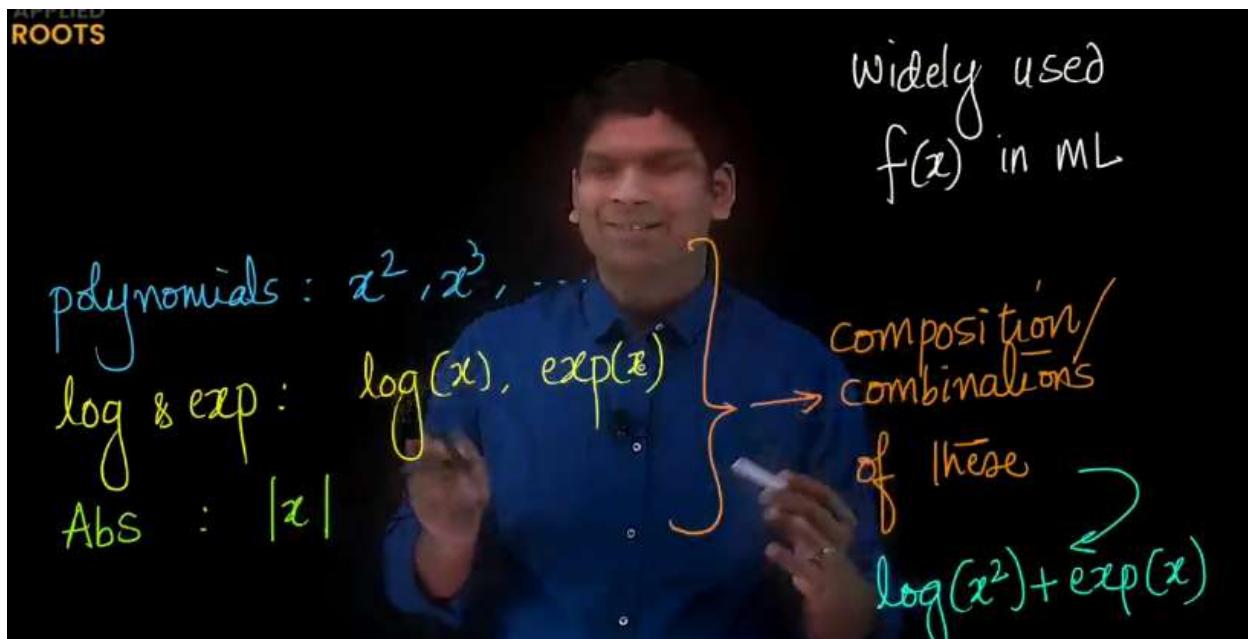


- The above function is absolute value function, the both sided limits of the function are 0 as shown above.



- The domain of the function is all positive real numbers and the range of the function is all positive real numbers including 0 because at $x=0$ the function gives 0 .

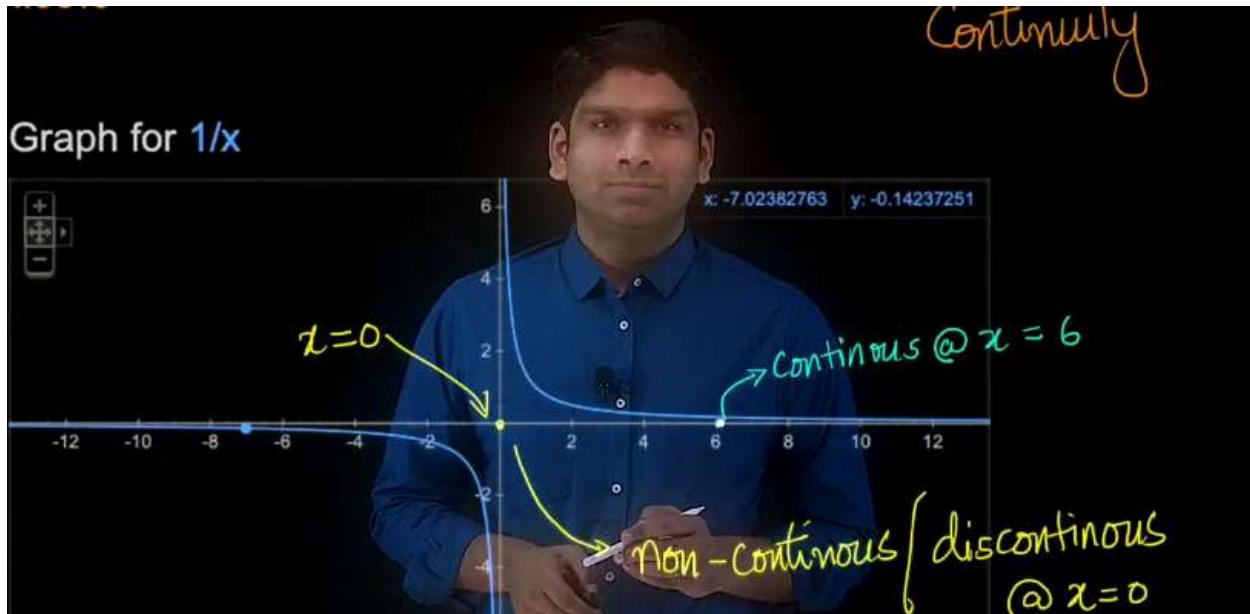
At timestamp 17.59 in video



- We use the above shown functions extensively in ML.

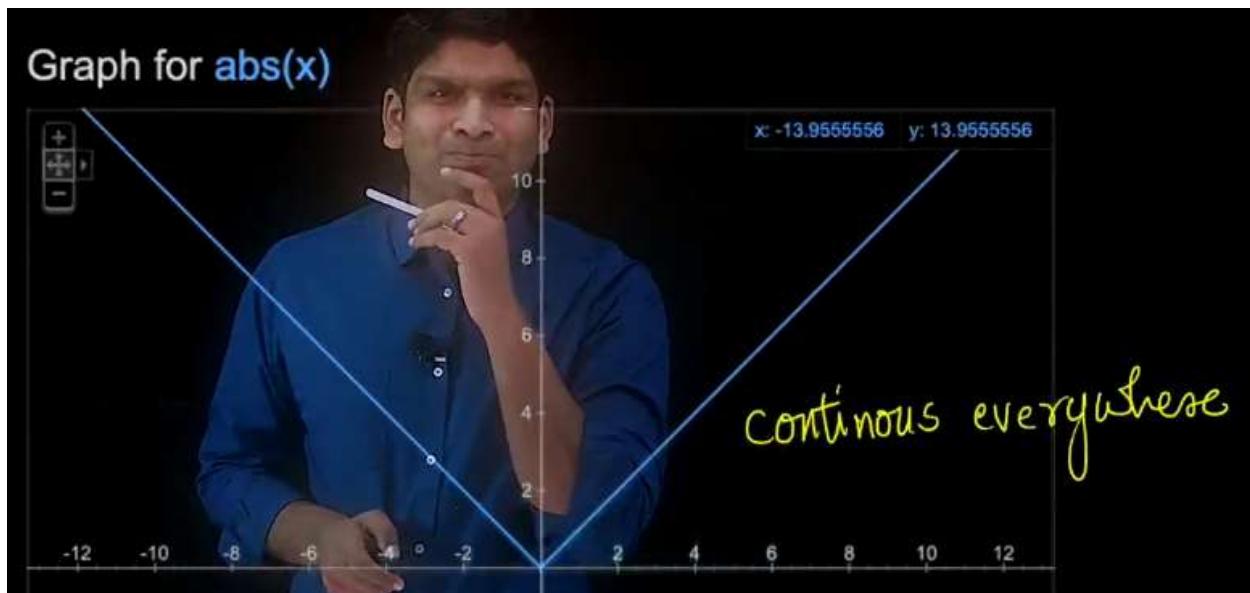
3.7 Continuity of functions

At timestamp 0.19 in video



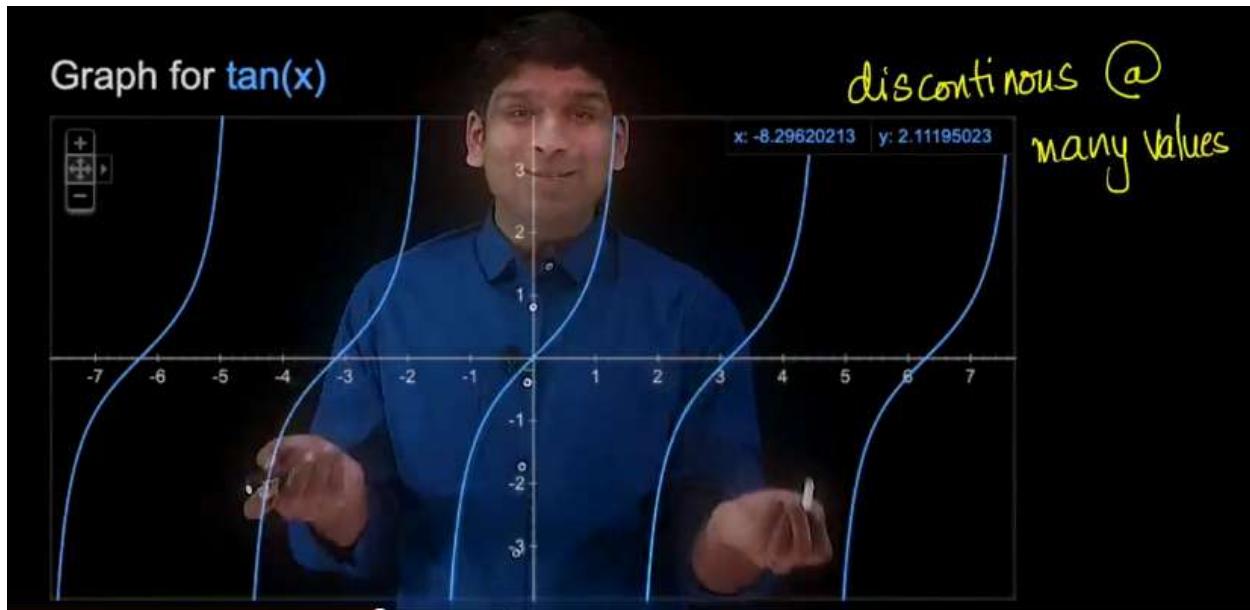
- Let's understand the concept of continuity of functions.
- Consider the function $1/x$ and from the plot you can see that at $x=6$ the function is continuous ,but at point $x=0$ the curve is discontinuous.

At timestamp 1.20 in video



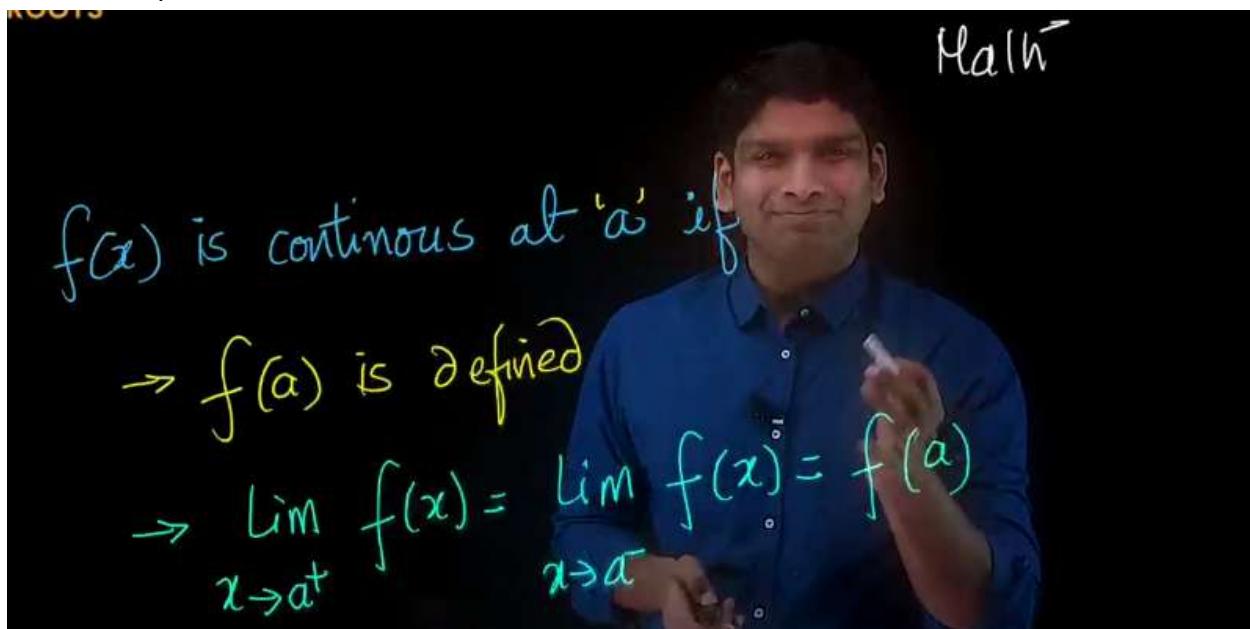
- If we consider the above function it is continuous everywhere even at $x=0$ th function is continuous,there is no gap or break or discontinuity.

At timestamp 2.08 in video



- For the function $\tan(x)$ you can clearly see from the above plot that there is a lot of discontinuity.

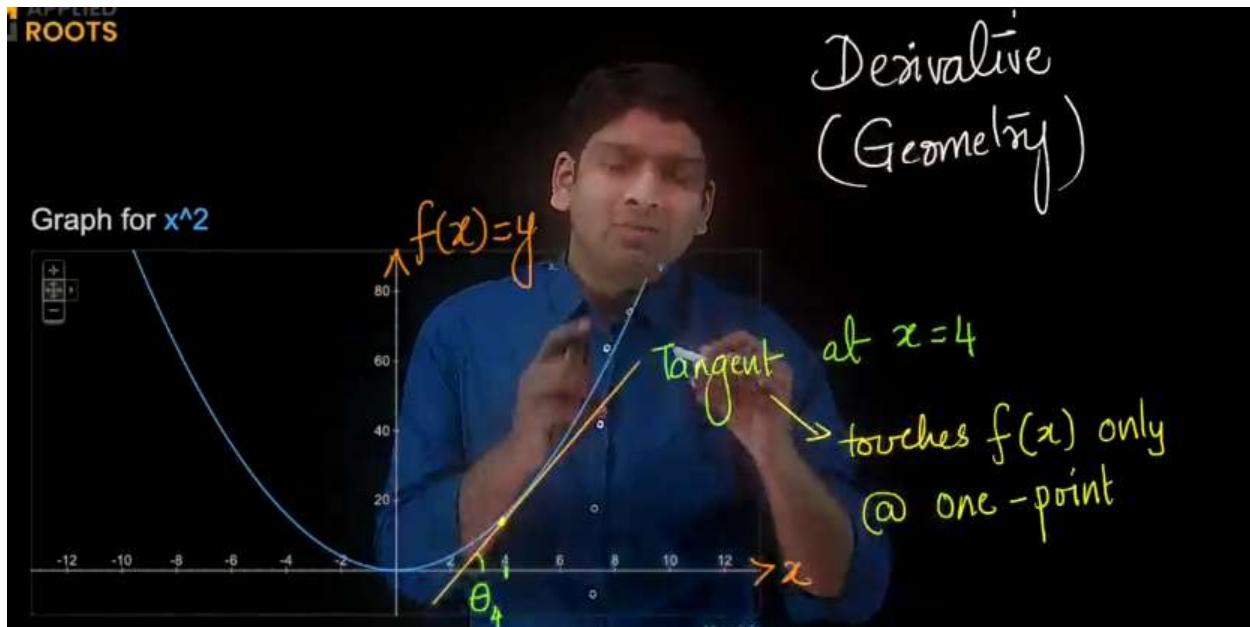
At timestamp 3.32



- Now that we have seen enough examples ,we define what continuity is mathematically as shown above.
- A function $f(x)$ is said to be continuous at a point $x = a$, in its domain if the following three conditions are satisfied:
 - $f(a)$ exists (i.e. the value of $f(a)$ is finite)
 - $\lim_{x \rightarrow a} f(x)$ exists (i.e. the right-hand limit = left-hand limit, and both are finite)
 - $\lim_{x \rightarrow a} f(x) = f(a)$

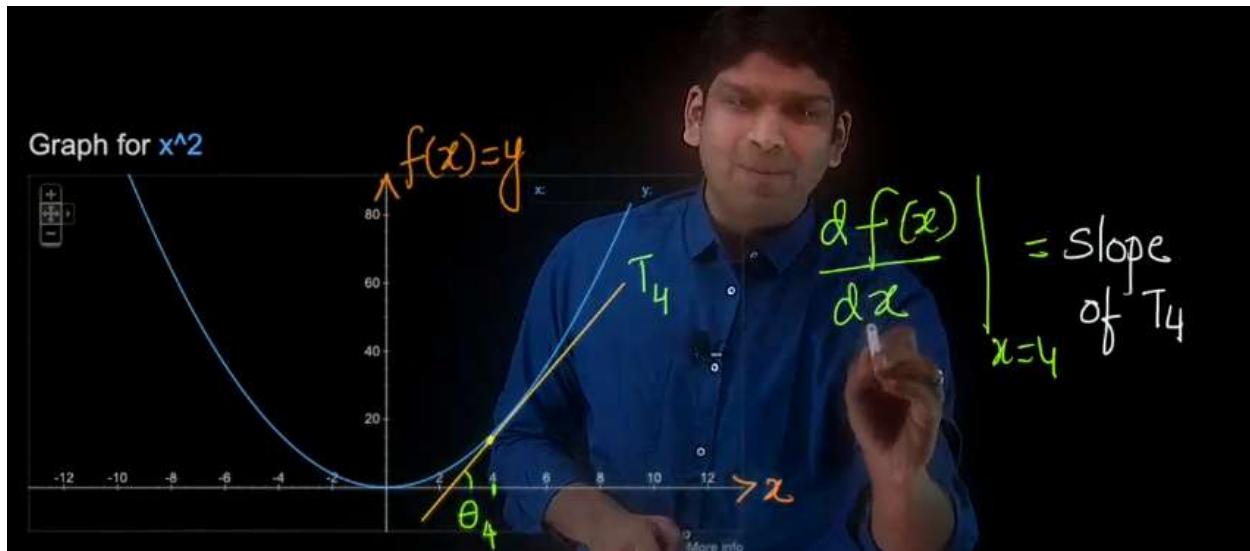
3.8 Derivatives: geometric intuition

At timestamp 0.24 in video



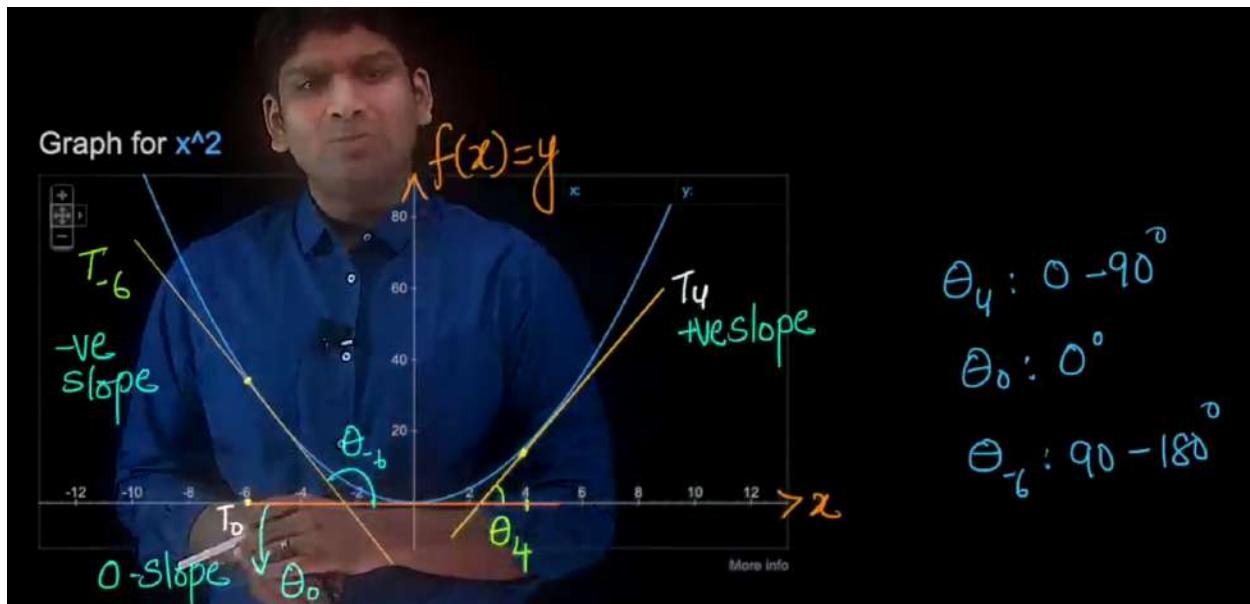
- Let's understand the concept of derivatives, consider the function $f(x) = x^2$ which is a parabola as shown above.
- A tangent is a straight line that touches the curve exactly at one point.
- If we consider point $x = 4$ corresponding y will be 16 , if we draw a tangent to the curve at this point $x = 4$ and we call it as T_4 . This tangent makes some angle with x axis we call it θ_4 .

At timestamp 3.35 in video



- The derivative of $f(x)$ with respect to x at $x=4$ as the slope of tangent T_4 and it is denoted mathematically as shown above.
- The slope of T_4 is nothing but $\tan\theta_4$.
- The slope of tangents at different points on the curve could be different.

At timestamp 6.19 in video

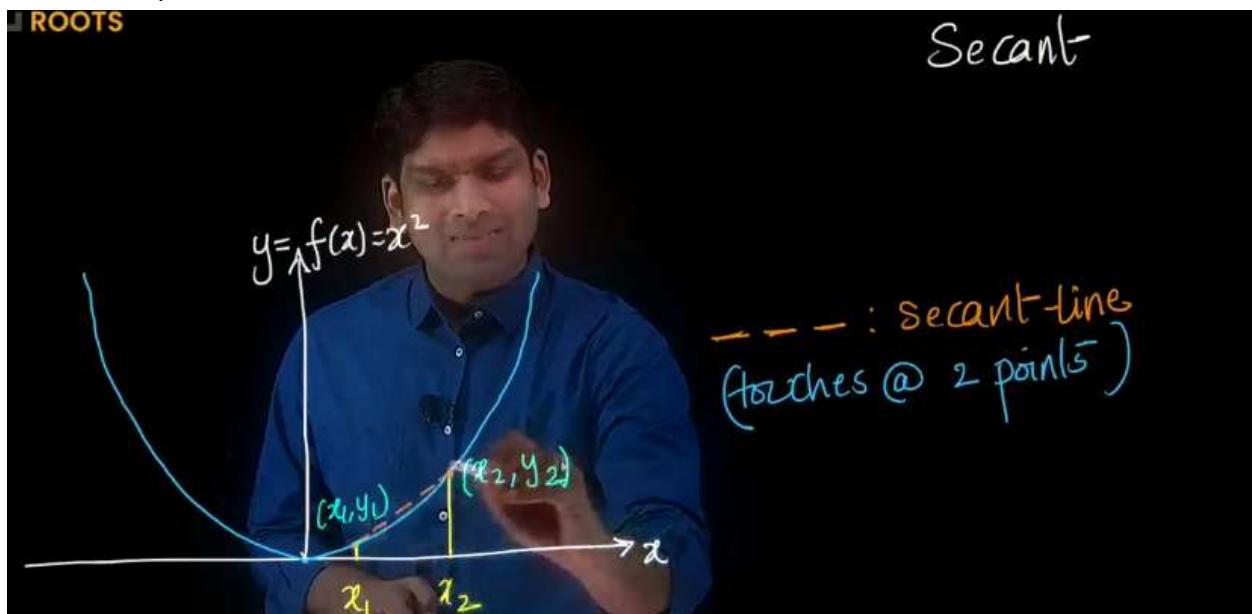


- Consider three tangents and angles made by the tangents with x axis as shown above. We can clearly see that T_4 is having positive slope, T_{-6} is having negative slope and T_0 has zero slope.
- If slope at a point is positive it means that the underlying curve is increasing on the other hand if the slope is negative at a point the curve will be decreasing.

- If the slope of a tangent at a point is zero at a point it's slightly complicated and we will discuss it.

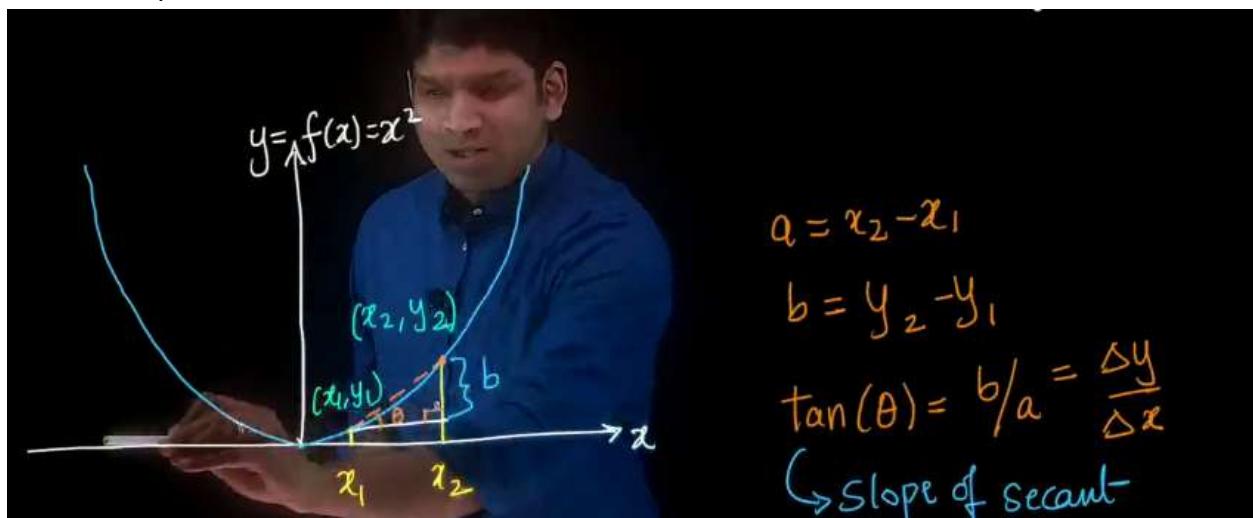
3.9 Derivatives: rate of change + Math

At timestamp 0.55 in video

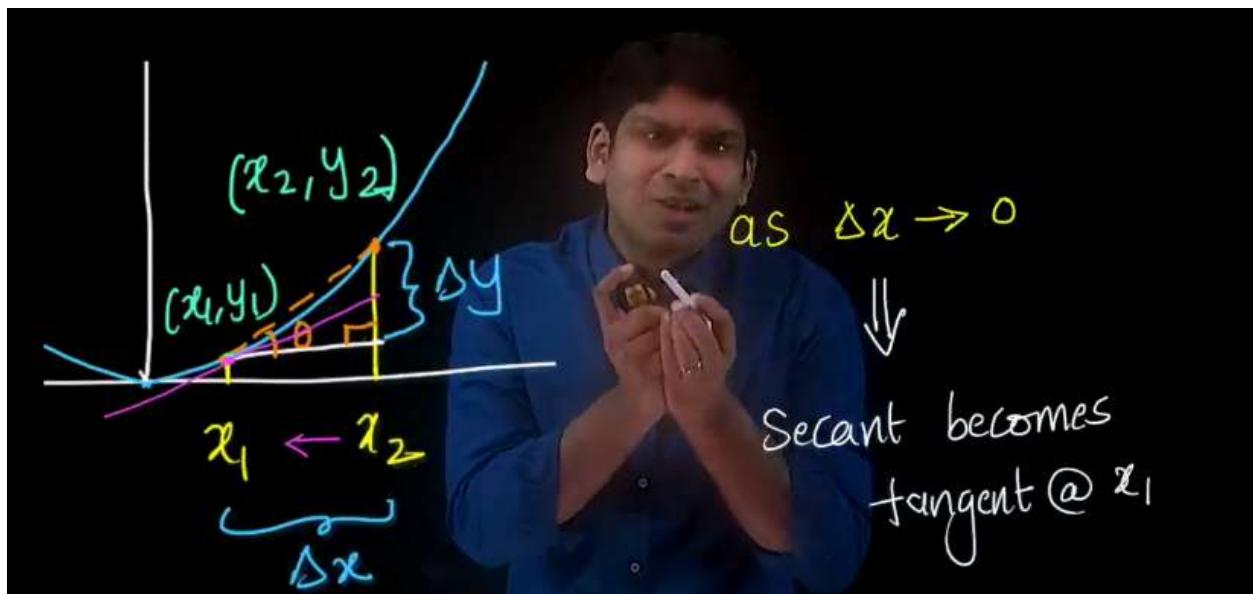


- Let's understand derivatives from a rate of change perspective .Consider two points on the curve x_1, x_2 and their corresponding coordinates as shown above .
- The line between two points is called as secant line

At timestamp 1.32 in video



- Slope of the secant line can be calculated as shown above.



- When $(x_2 - x_1) \Delta x \rightarrow 0$ secant becomes tangent x_1 .

$$\frac{dy}{dx} \Big|_{x=x_1} = \lim_{(x_2-x_1) \rightarrow 0} \frac{y_2 - y_1}{x_2 - x_1}$$

$$= \lim_{\Delta x \rightarrow 0} \frac{\Delta y}{\Delta x}$$

- The derivative of function $y=f(x)$ at $x=x_1$ can be written using limits as shown above
- So the derivative is the rate of change of y around the point $x=x_1$. Instead of computing the derivative at a particular point we can compute it at all points.

At time stamp 9.56

$$\frac{d(x^2)}{dx} = \lim_{\Delta x \rightarrow 0} \frac{(x+\Delta x)^2 - x^2}{(x+\Delta x) - x}$$

$$\frac{d(x^2)}{dx} = \lim_{\Delta x \rightarrow 0} \frac{(x + (\Delta x)^2 + 2x\Delta x) - x^2}{\Delta x}$$
$$= \lim_{\Delta x \rightarrow 0} (\Delta x + 2x) = 2x + 0 = 2x$$

$$\frac{d(x^2)}{dx} = 2x$$
$$\left. \frac{d(x^2)}{dx} \right|_{x=4} = 2 \times 4 = 8$$

- The derivative of the function can be calculated as shown above .
- For computing the derivative at a specific point we can substitute the corresponding x value.

3.10 Derivative of Common functions

At timestamp 0.10 in video

The image shows a man in a blue shirt writing mathematical equations on a whiteboard. He is writing the derivative of x^n with respect to x , which is nx^{n-1} , noting that $n \neq 0$ and n is a rational number. Below this, he is writing the derivative of $x^{2.5}$ with respect to x , which is $(2.5)x^{1.5}$. There is handwritten text "Proof in references (under the video)" pointing to the derivation. The word "Derivatives" is written in orange at the top right of the whiteboard.

$$\frac{d x^n}{d x} = nx^{n-1}; n \neq 0, n: \text{rational number}$$

e.g:

$$\frac{d x^{2.5}}{d x} = (2.5)x^{1.5}$$

Derivatives

Proof in references
(under the video)

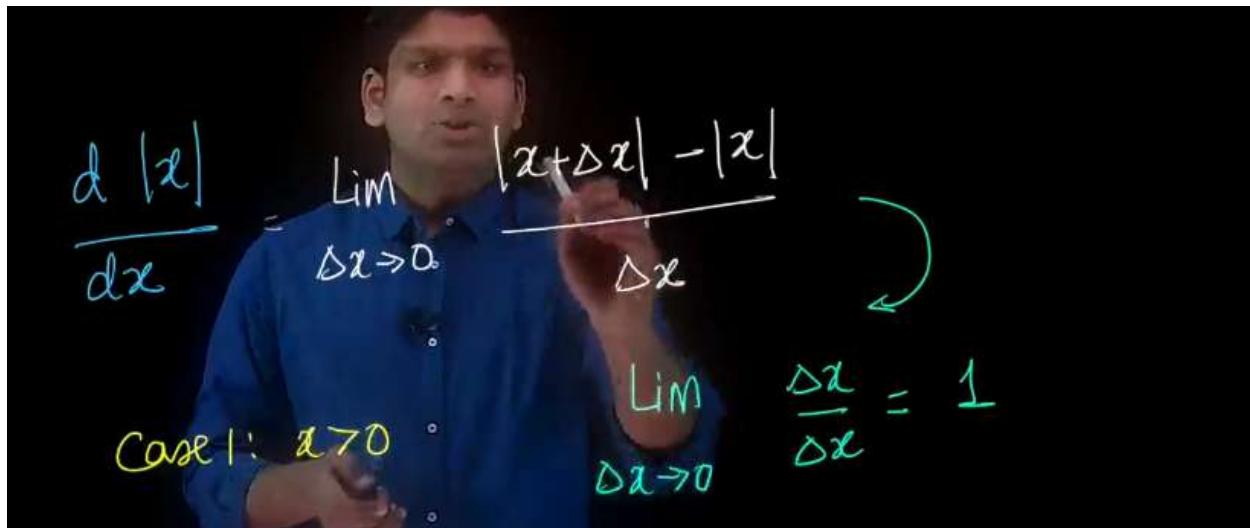
- Let's see the derivatives of most commonly used functions in Machine learning. The function shown above is a polynomial function and the derivative can be obtained as shown.

The image shows a man in a blue shirt writing derivatives on a whiteboard. He is writing the derivative of $\log(x)$ with respect to x , which is $\frac{1}{x}$. Next to it, he is writing the derivative of e^x with respect to x , which is e^x . There is handwritten text "Proofs in references under video" pointing to the derivations.

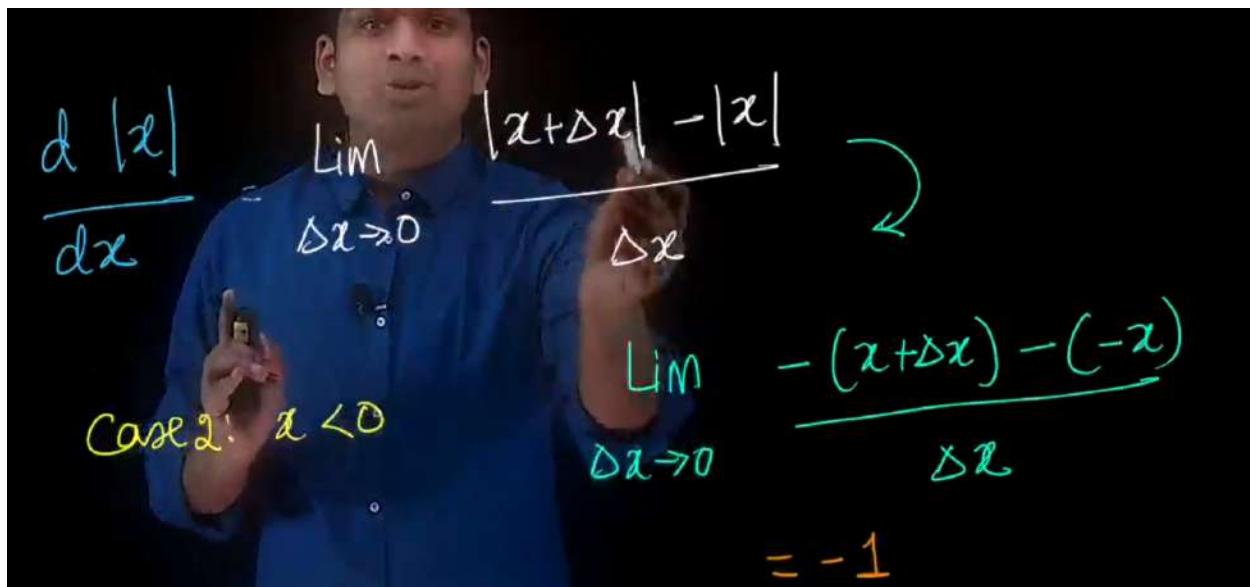
$$\frac{d \log(x)}{d x} = \frac{1}{x}$$
$$\frac{d (e^x)}{d x} = e^x$$

Proofs in references under video

- Derivatives of $\log x$ and e^x are as shown above, using limits you can solve these easily.



- The derivative of $\text{abs}(x)$ can be calculated as shown using limits.
- In case of absolute value function if $x>0$ the derivative is +1



- When we have negative x value $x<0$ then the derivative is -1.

$$\frac{d|x|}{dx} = \lim_{\Delta x \rightarrow 0} \frac{|x + \Delta x| - |x|}{\Delta x}$$

Case 3: $x=0$

$$\lim_{\Delta x \rightarrow 0} \frac{|\Delta x|}{\Delta x}$$

$$\text{Case 3: } x=0$$

$$\lim_{\Delta x \rightarrow 0^+} \frac{|\Delta x|}{\Delta x} = \frac{\Delta x}{\Delta x} = 1$$

$$\lim_{\Delta x \rightarrow 0^-} \frac{|\Delta x|}{\Delta x} = \frac{-\Delta x}{\Delta x} = -1$$

- When x value $x=0$ then the derivative is not defined because the left handed limit is not equal to right handed limit.

3.11 Differentiability of functions

At time stamp 0.10

Differentiability

$|x|$ is not differentiable at $x=0$

$$\lim_{\Delta x \rightarrow 0} \frac{|x+\Delta x| - |x|}{\Delta x} \Big|_{x=0}$$

↓
is not defined

- Absolute value function is not differentiable at $x=0$ but it might be differentiable at other points

At timestamp 1.59

Geometry

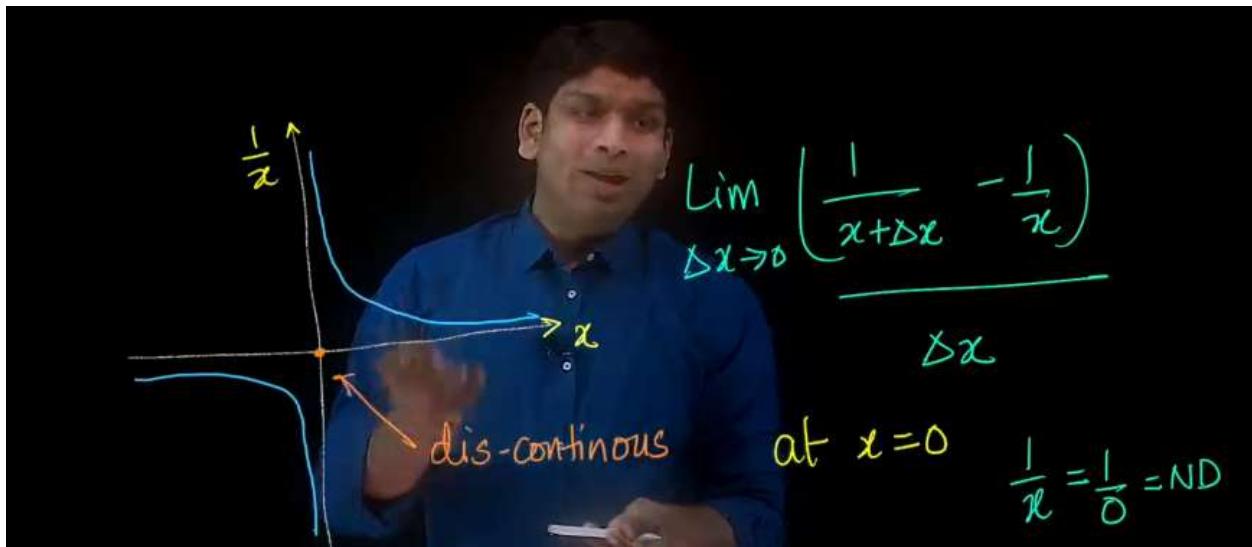
$|x|$

$f(x)$

non-differentiable
(non-smooth)

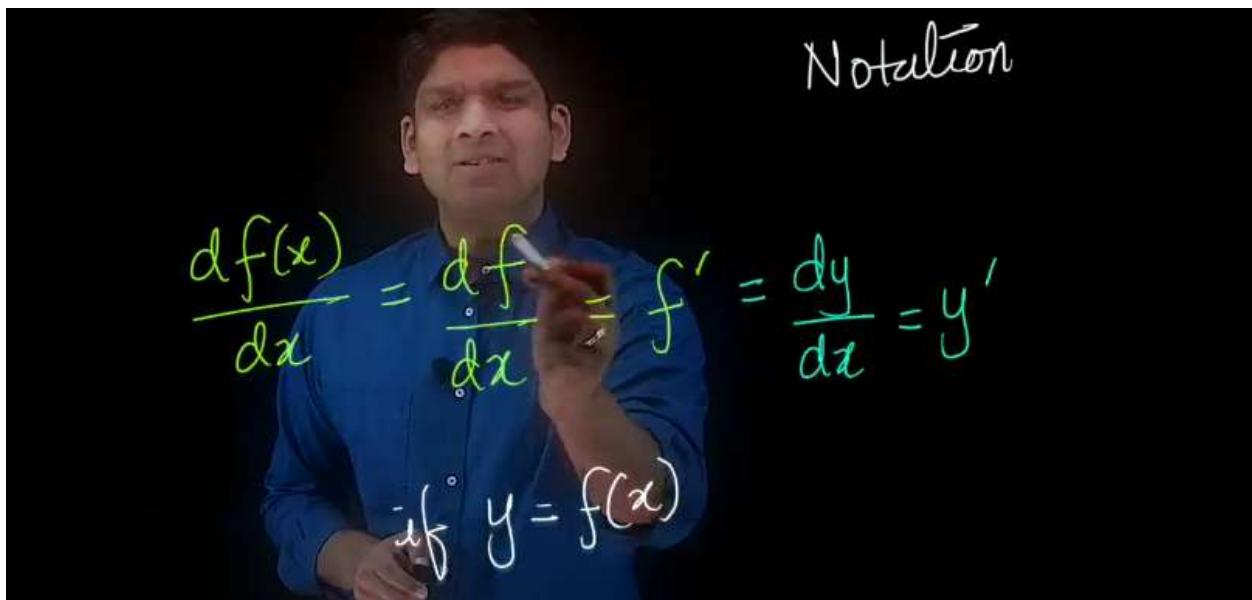
- You can see for above functions are differentiable at other parts of the curve but at $x=0$ they are not. Wherever the graph or curve is non-smooth we tend to face the problem of non-differentiability.

At timestamp 3.02



- We face the problem of non-differentiability when the function is discontinuous, because the positive and negative limits will not be the same .

3.12 Rules of differentiation



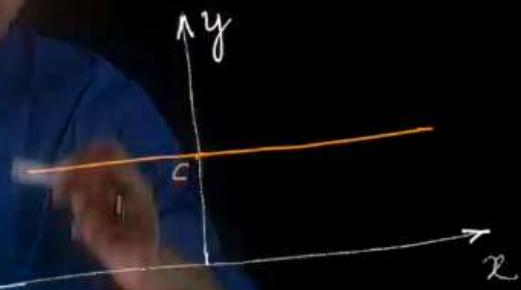
- The derivative of $y=f(x)$ with respect to x can be represented in any of the above ways.
- We have rules of differentiation which help us when we try to differentiate large equations. These rules are listed below.

Rules of differentiation

$$\frac{d}{dx} (f(x) + g(x)) = \frac{df}{dx} + \frac{dg}{dx}$$

Rules of differentiation

$$\frac{d c}{d x} = 0$$



- The derivative of constant is with respect to x is 0.

Chain rule

$$\frac{d f(g(x))}{dx} = f'(g(x)) \cdot g'(x)$$

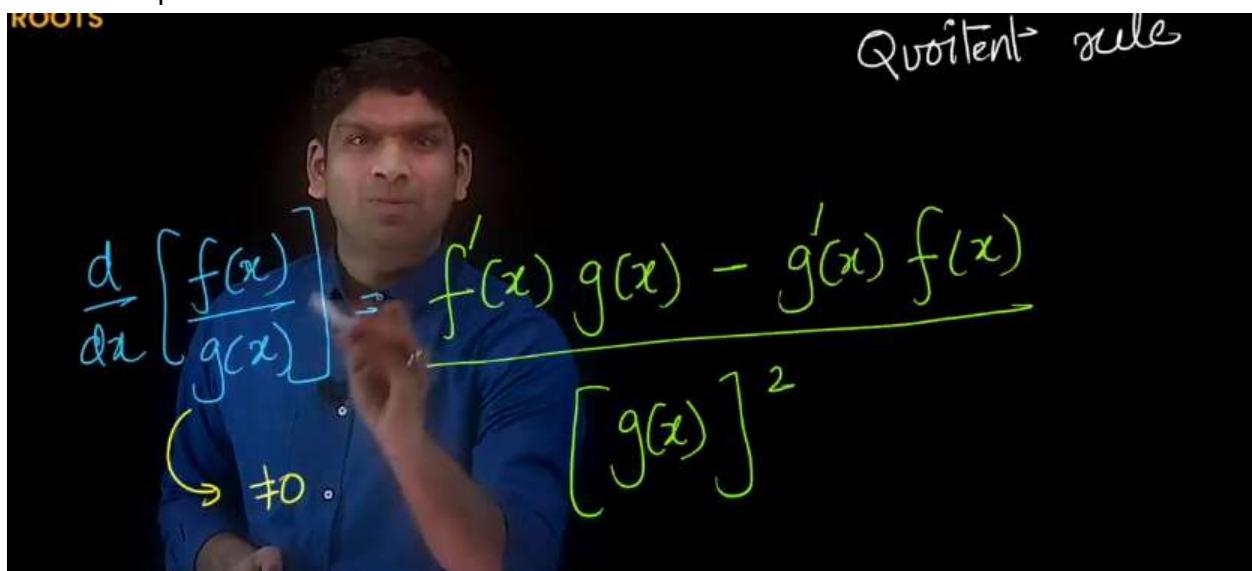
Rules of differentiation

$$\frac{d f(x) \cdot g(x)}{dx} = f(x) \frac{dg}{dx} + g(x) \frac{df}{dx}$$

↑ Product - rule

- When we have to find the derivative of a product of two functions we use product rule.

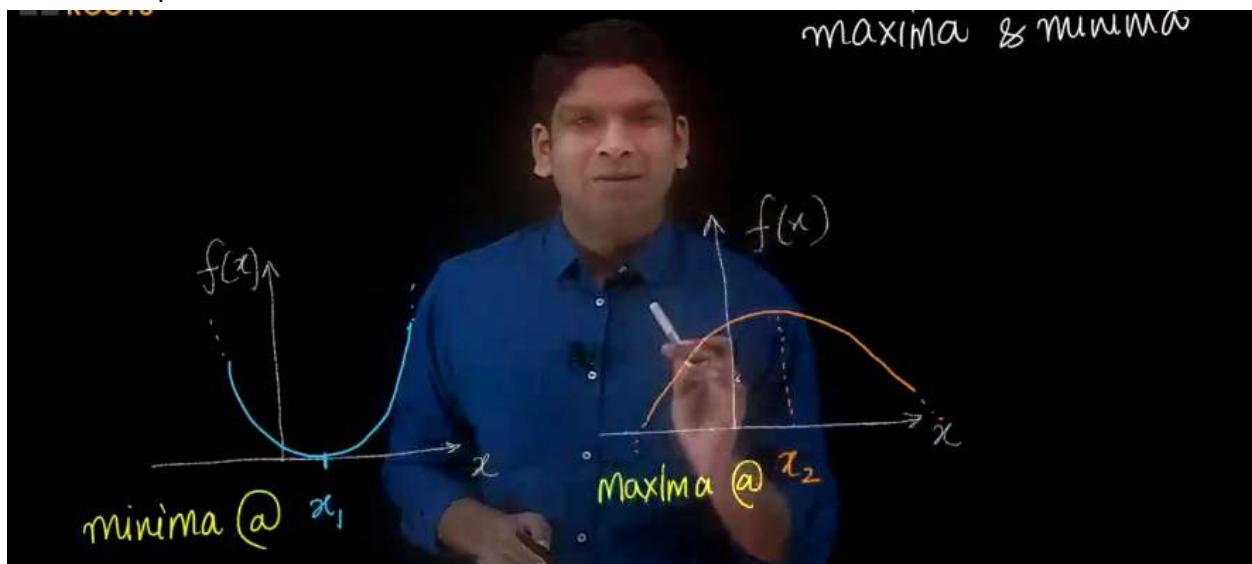
At timestamp 13.07



- Quotient rule is used when we have to find the derivative of two functions where $g(x)$ is dividing $f(x)$. Here $g(x)$ cannot be 0.

3.13 Maxima and Minima

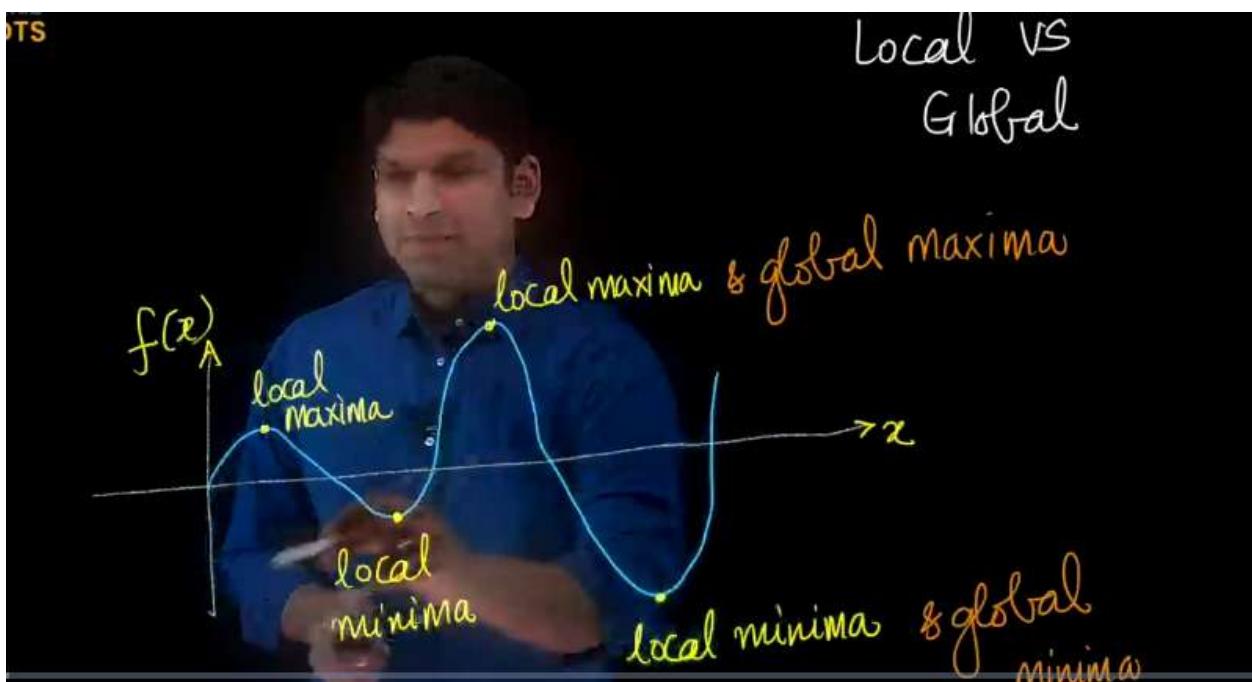
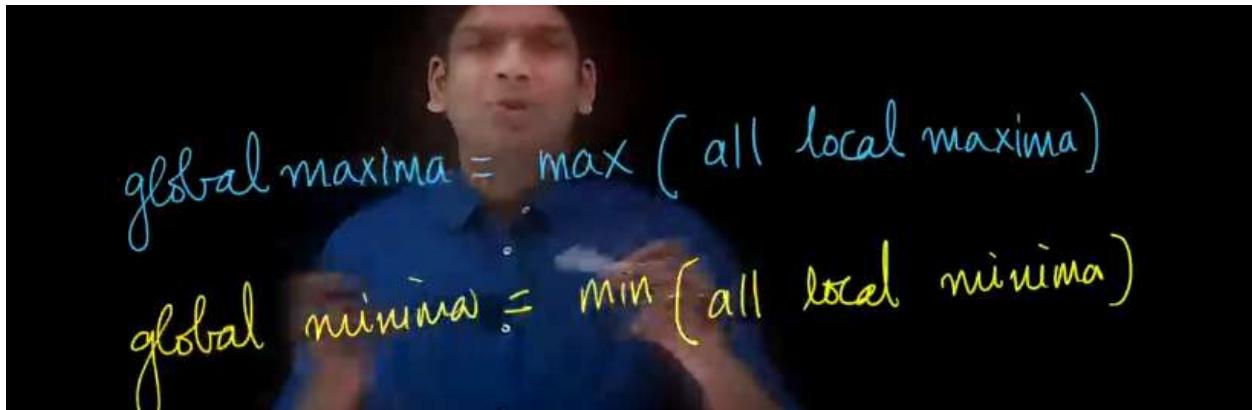
At timestamp 1.19



- All the calculus we have studied till now is primarily to study maxima and minima. In MI most of the time we try to minimize or maximize certain functions or mathematical expressions.
- In the functions shown above the first function doesn't have maxima but has minima at $x=x_1$ and the other doesn't have minima but has maxima at $x=x_2$.

- Function can have maxima and minima some functions can have multiple maxima and minima some functions may not have any minima or maxima.

At timestamp 2.19



- Whenever we have local maxima or minima we use a certain term called optima and it can be either maxima or minima.
- We can have local optima which can be either local maxima or minima. Similarly we have global optima which can be global maxima or global minima.

At timestamp 6.09

ROOTS

How to find them?

$f(x)$

T_1

T_2

θ_1

θ_2

x_1

x_2

$0 \longleftrightarrow +$

$0^\circ < \theta_2 < \theta_1 < 90^\circ$

$0 < \tan(\theta_2) < \tan(\theta_1) < \infty$

Slope @ ($x=0$) = 0

- Consider a curve as shown above $y=x^2$ and two tangents T_1, T_2 at points x_1, x_2 on curve and making angels θ_1 and θ_2 with x axis respectively. Let's see how to find minima mathematically.
- We can clearly see that the slope of tangent T_2 is less than slope of T_1 . Slope is nothing but the derivative of the function at $x=x_1, x_2$
- The derivative of x_1 and x_2 are positive but derivative of $x_1 >$ derivative of x_2 .
- At $x=0$ the tangent is nothing but x-axis, so the slope of tangent at $x=0$ is 0.

At timestamp 13.10

ROOTS

How to find them

$f(x)$

T_3

T_4

θ_3

θ_4

x_3

x_4

$0 \longleftrightarrow -ve$

$180^\circ > \theta_3 > \theta_4 > 90^\circ$

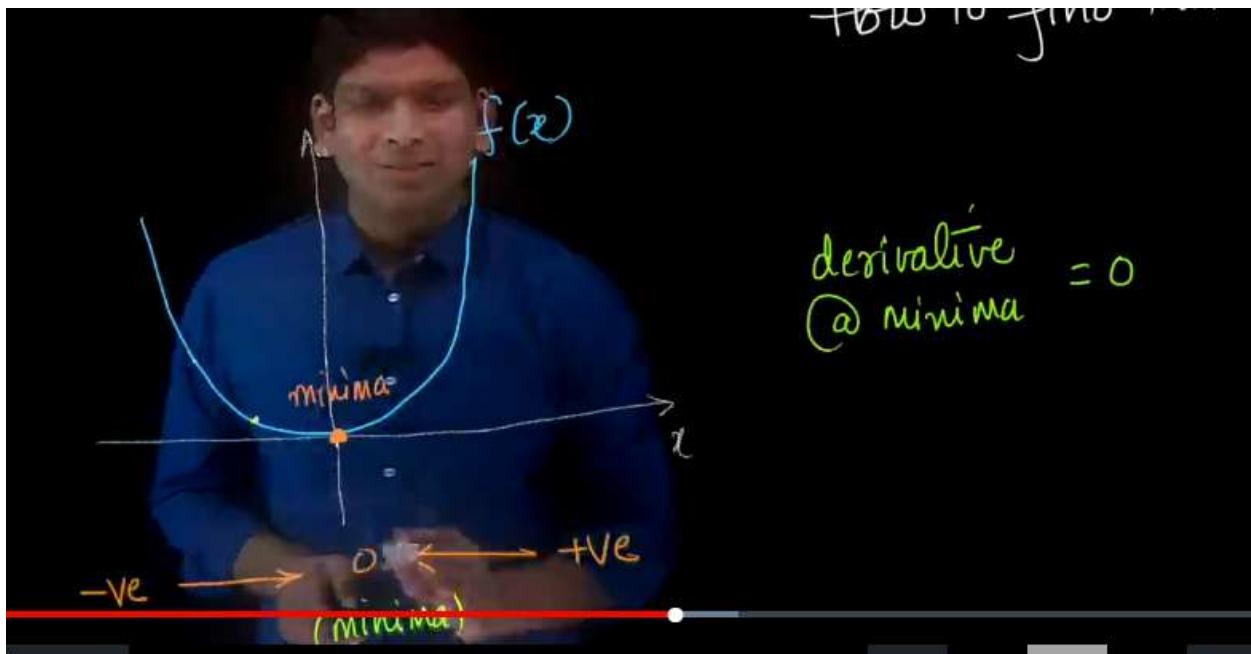
$\Rightarrow \tan(\theta_3) < 0$

$\tan(\theta_4) < 0$

$\tan(\theta_3) > \tan(\theta_4)$

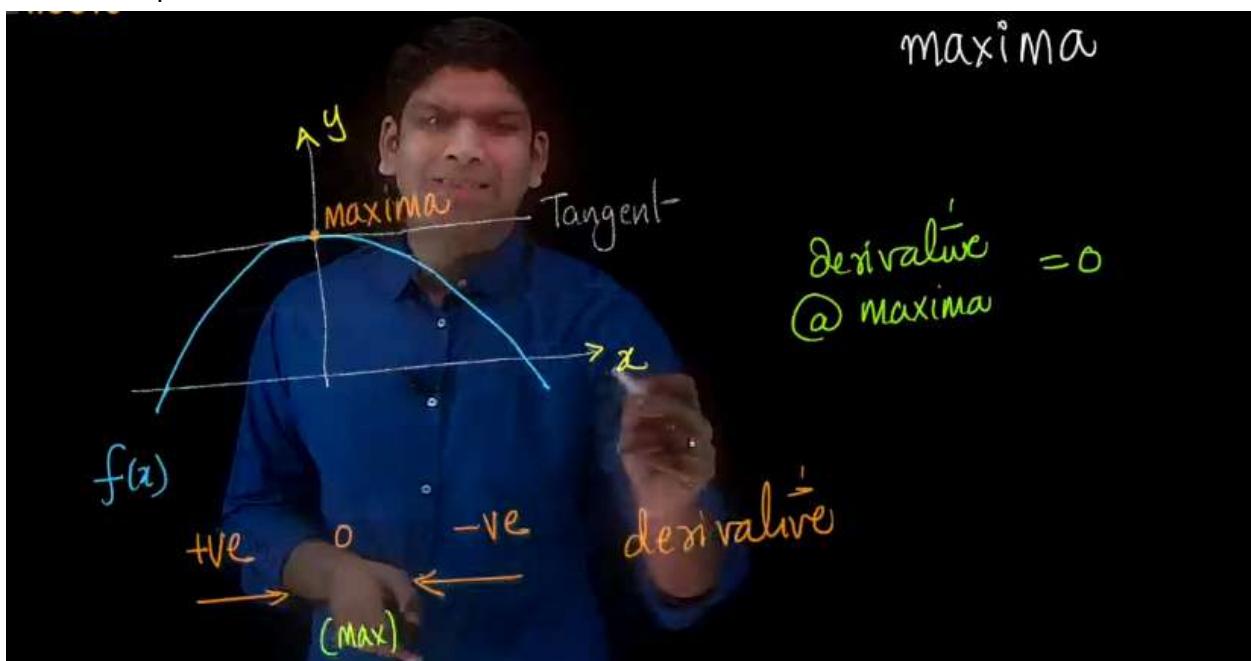
- You can observe a similar trend on the left side of the curve as well as shown above.

At timestamp 13.58



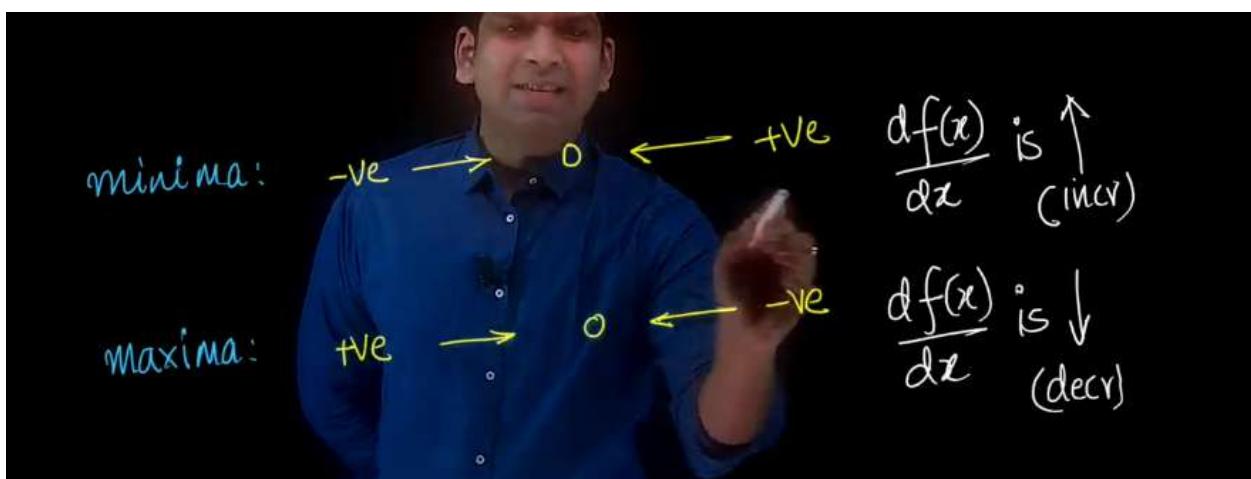
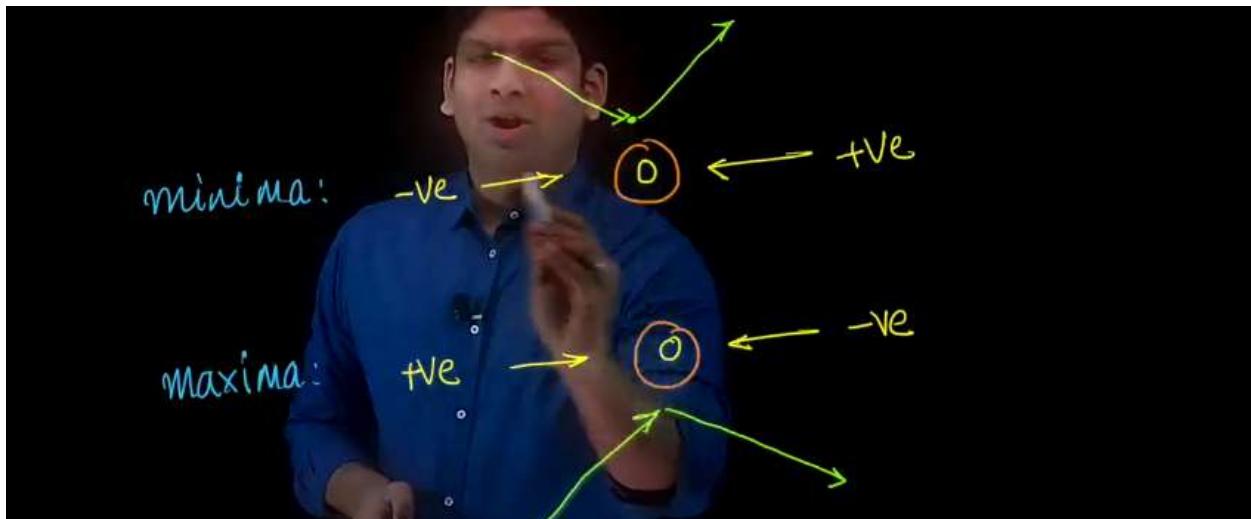
- We can observe that on one side of minima the derivatives are all positive and on other side derivatives are all negative at minima the derivative is zero
- We find the minima at a point where the derivative is zero and function should be increasing on right side of minima and decreasing on left side of the minima

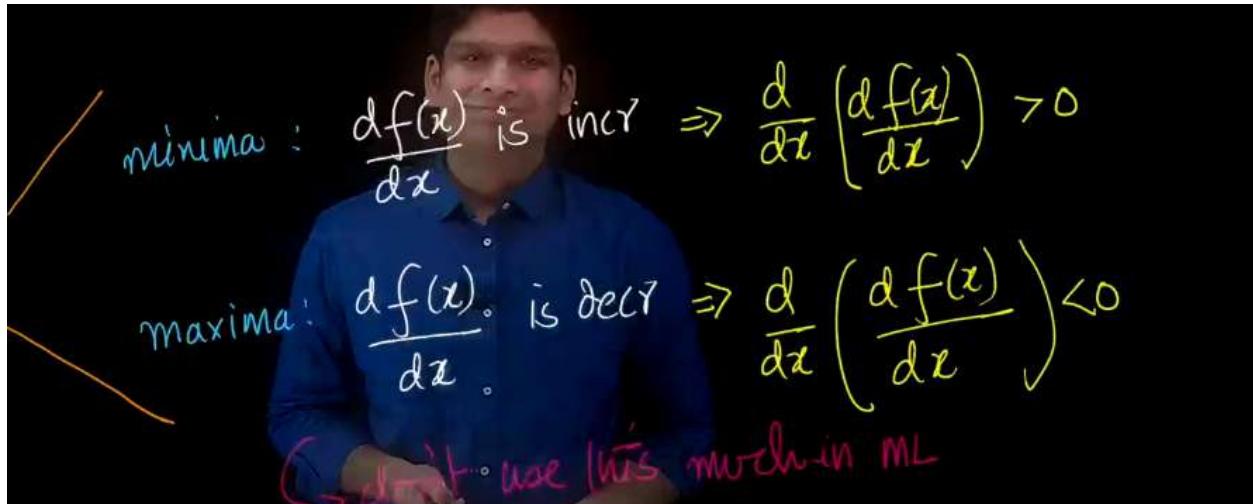
At timestamp 16.40



- Just like the derivative at minima is zero the derivative at maxima is also 0. You can see that the angle made by tangent with x-axis is 0 so the derivative or slope is 0.
- We find the maxima at a point where the derivative is zero and the function should be decreasing on the right side of the maxima and increasing on the left side of the maxima

At timestamp 18.54

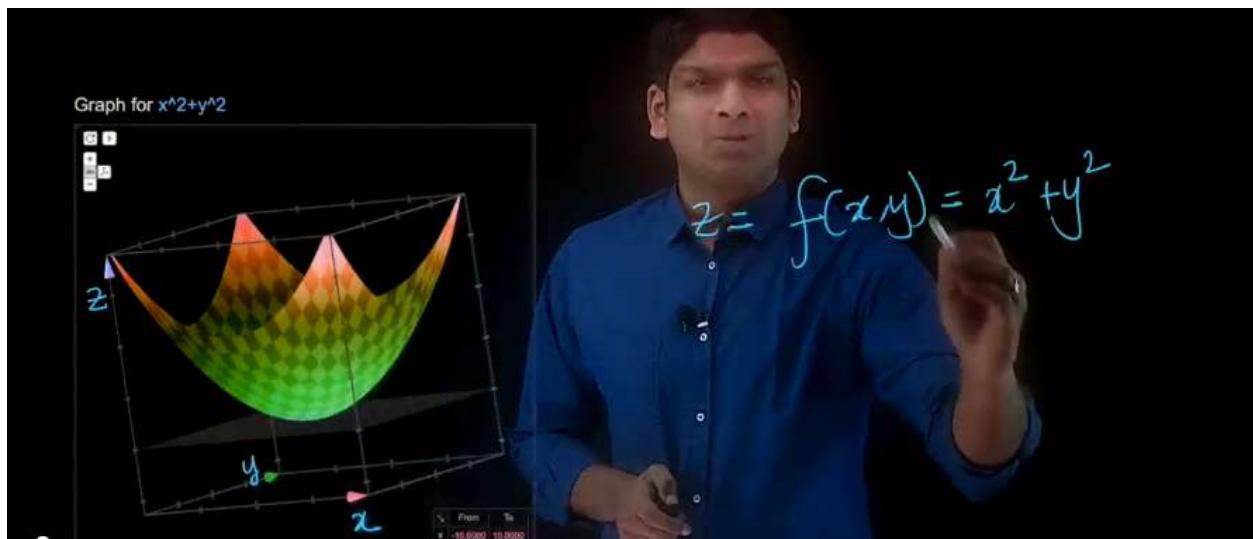




- When the derivative is zero we can analyse whether it is maxima or minima as shown above.

3.15 Partial derivatives & Del

At timestamp 2.4



- When we have multivariate functions as shown above we use partial derivatives .
- The above function is quadratic,we have three variables x,y,z so we can plot the function in 3D.

At timestamp 3.51

ROOTS

Multi-variable
derivative

derivative of $f(x, y) = z$?

$$\nabla z = \begin{bmatrix} \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial y} \end{bmatrix}$$

del

partial derivative

2-dim vector

doh

$$z = x^2 + y^2$$

$\frac{\partial z}{\partial x} : \frac{dz}{dx}$ with y as const

$$\frac{\partial z}{\partial x} = 2x + 0$$

$$\frac{\partial z}{\partial y} = 0 + 2y$$

$$\nabla z = \begin{bmatrix} \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

rate of change w.r.t x

rate of change w.r.t y

Let $z = x_1^2 + x_2^2$

$$\text{So } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\nabla z = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{bmatrix}$$

- We have seen how to find derivatives when we have one variable, here we have more than one variable so we use a concept called partial derivative as shown above.
- We can extend the same concept even for n dimensions.

3.16 Optima using Partial derivatives

At timestamp 0.53

$$z = f(x, y)$$

@ optima,

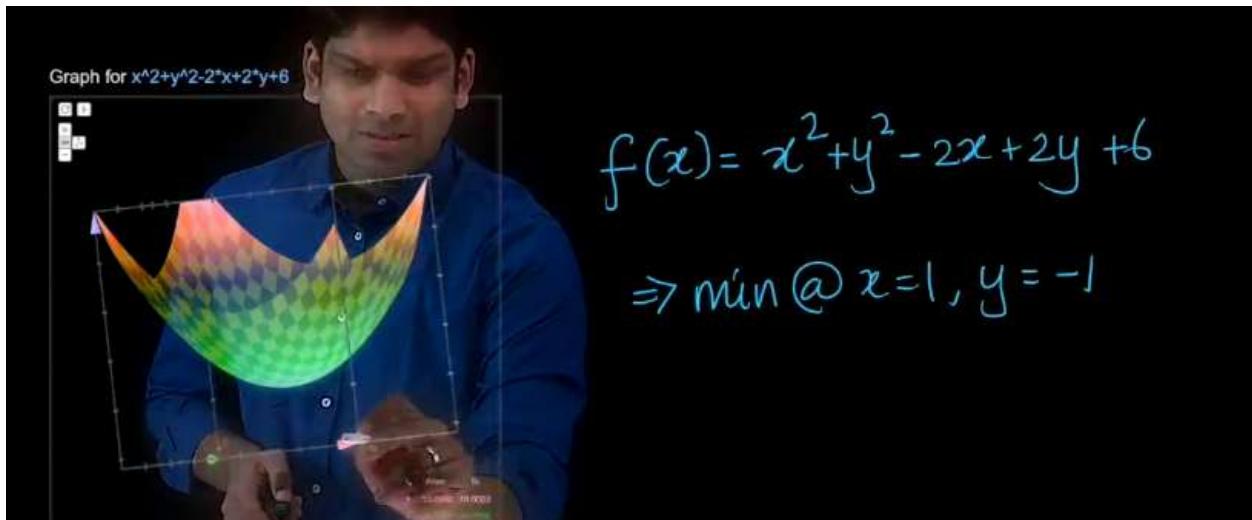
$$\begin{bmatrix} \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- At optima each of these partial derivatives will be zero (since we vector of partial derivatives each component must be equal to 0)

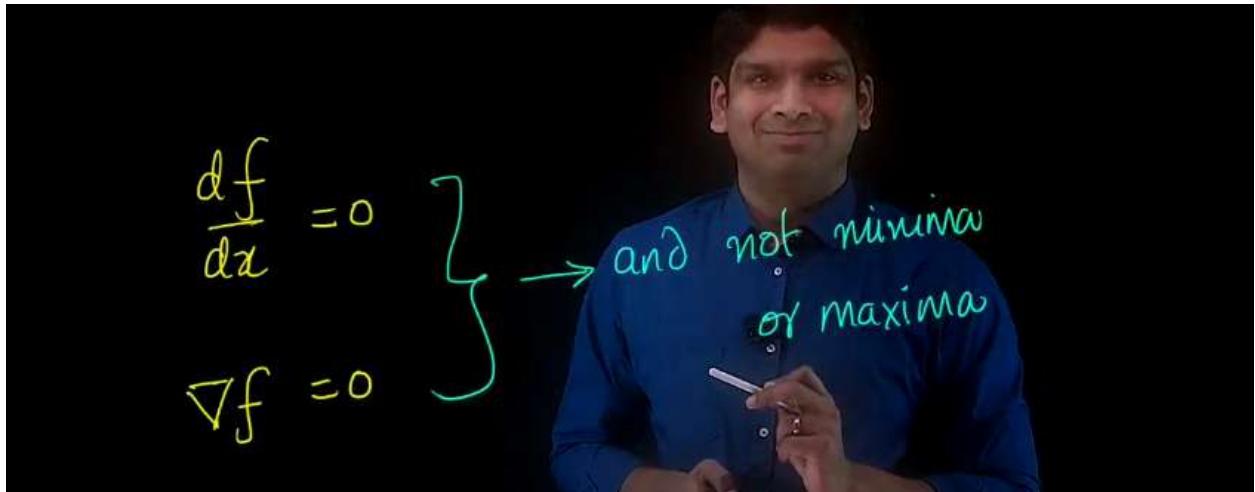
At timestamp 2.01

$$(e.g) \quad z = x^2 + y^2 - 2x + 2y + 6$$
$$\nabla z = \begin{bmatrix} \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x - 2 \\ 2y + 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow x = 1, y = -1$$

- Above is an example of calculating partial derivatives and by equating them to 0 we got optima at $x=1$ and $y=-1$. The optima can be either minima or maxima.



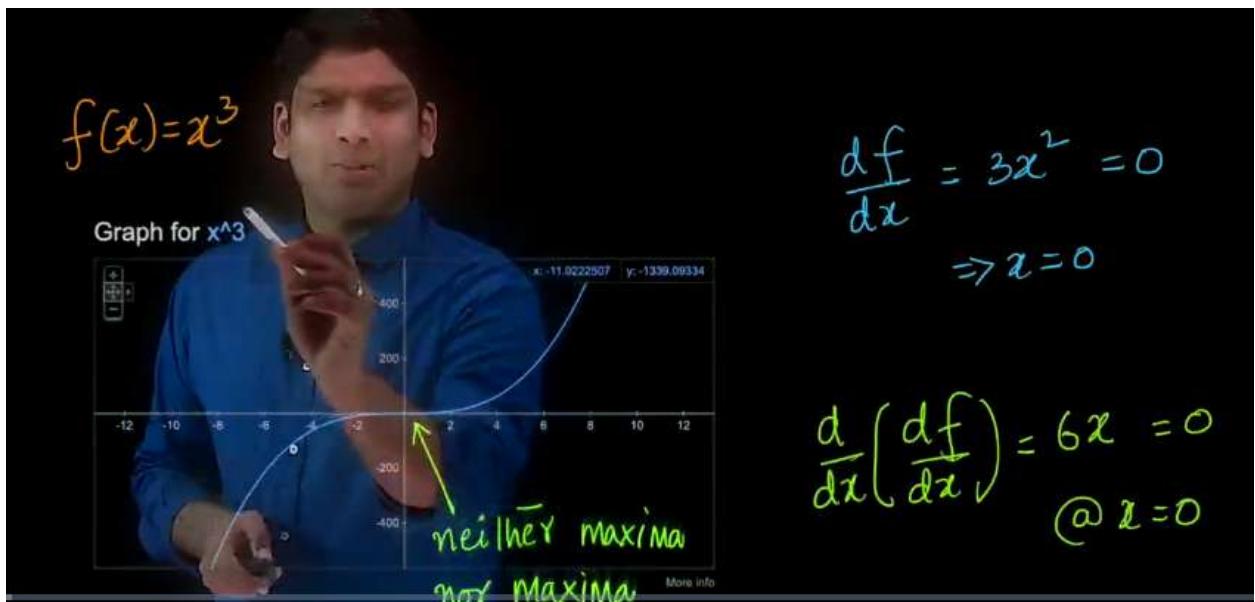
- If we plot the function we can clearly see that the optima that we arrived at using partial derivatives at $x,y=(1,-1)$ is minima.



- There is a special case where the derivative of the function (can be regular derivative or vector of partial derivatives) is 0 but it doesn't have either minima or maxima.

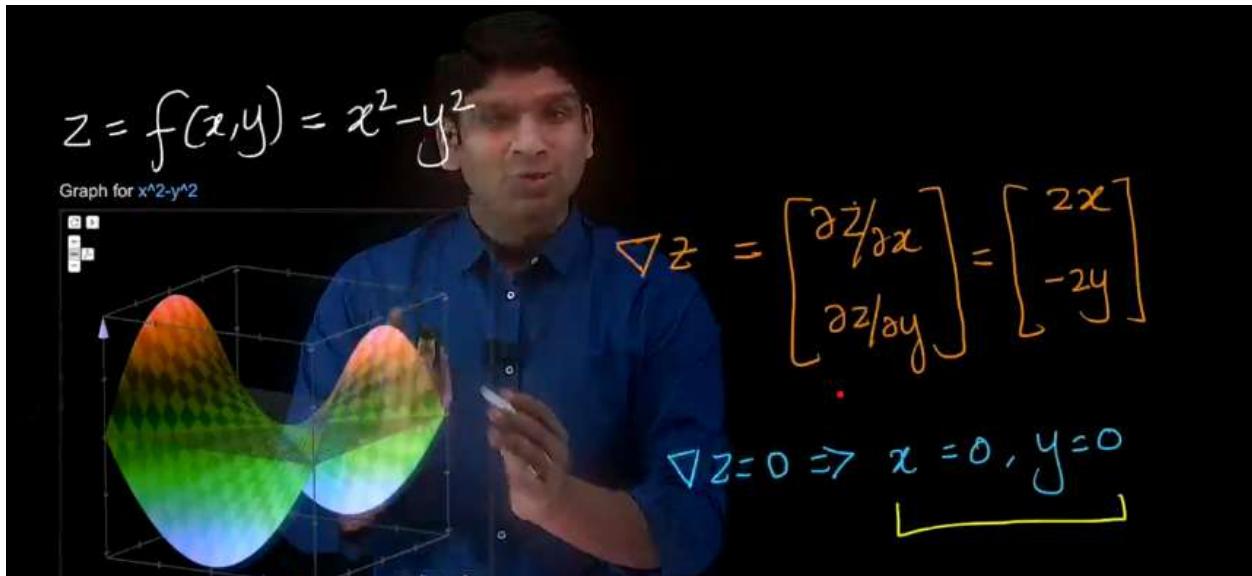
3.17 Saddle point

At timestamp 1.15

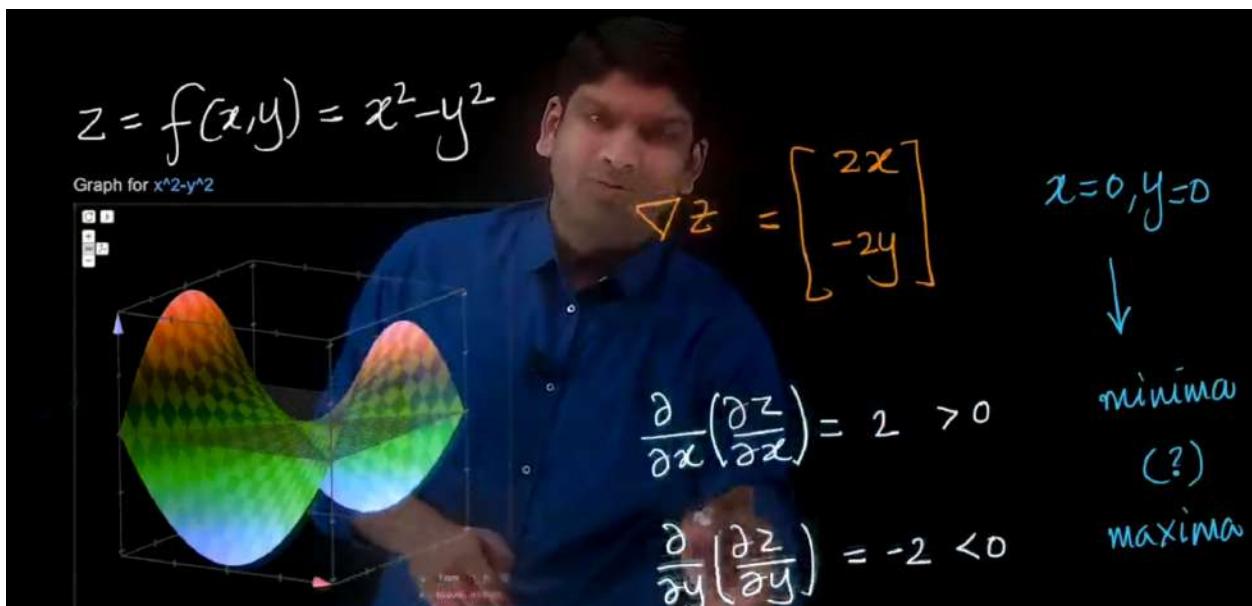


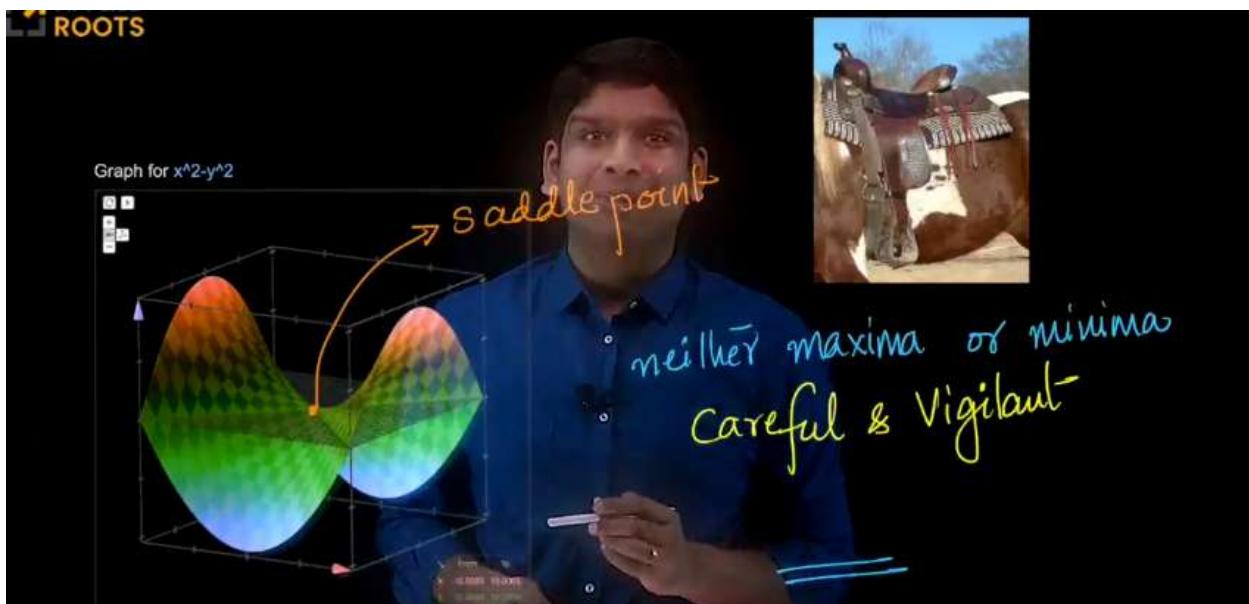
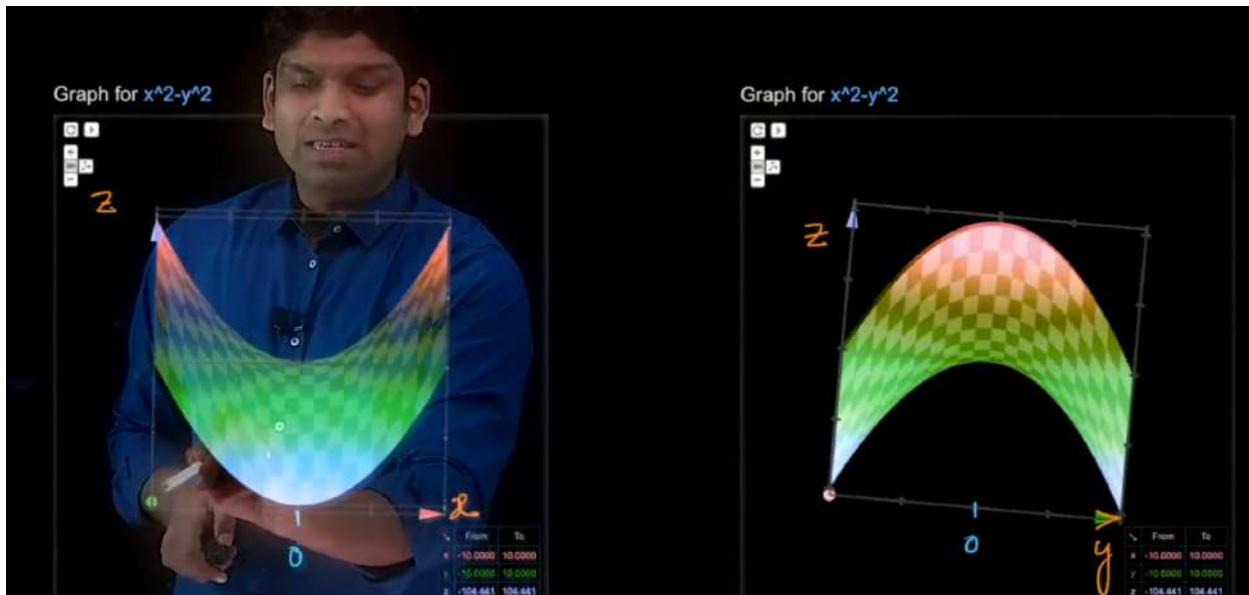
- For the above function you can clearly see that the derivative of the function at $x=0$ is zero but it's either minima or maxima .

At timestamp 3.45



- We we have more than one variable in our function as shown above it's slightly tricky ,you can find the partial derivatives and can say that at $x,y=(0,0)$ we have optima.But there is no optima

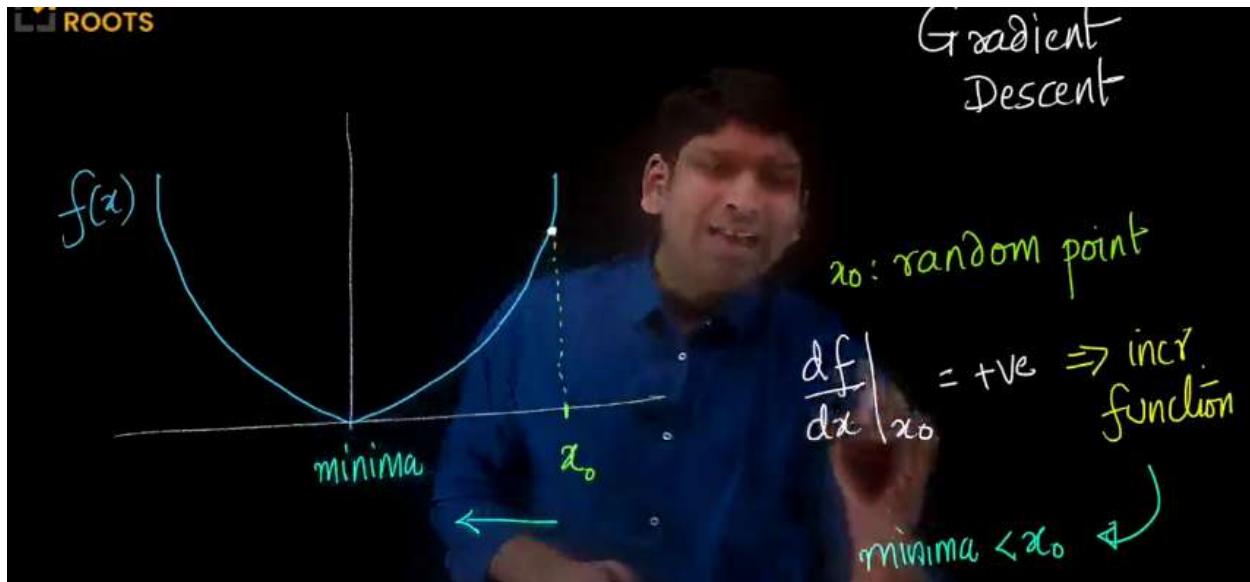




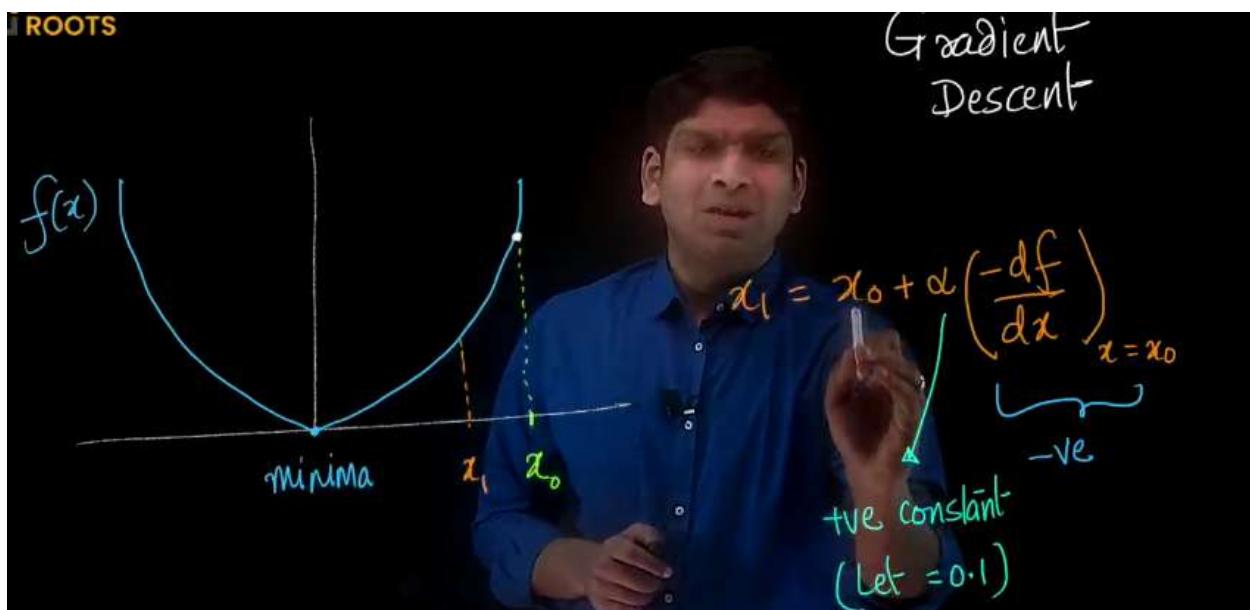
- When we calculate second order partial derivatives you can observe that x is increasing(positive) and y is decreasing(negative). From x's perspective the point is minima and from y's perspective it is maxima.
- Such points are called saddle points and we cannot say that it is either maxima or minima.

3.18 Gradient Descent

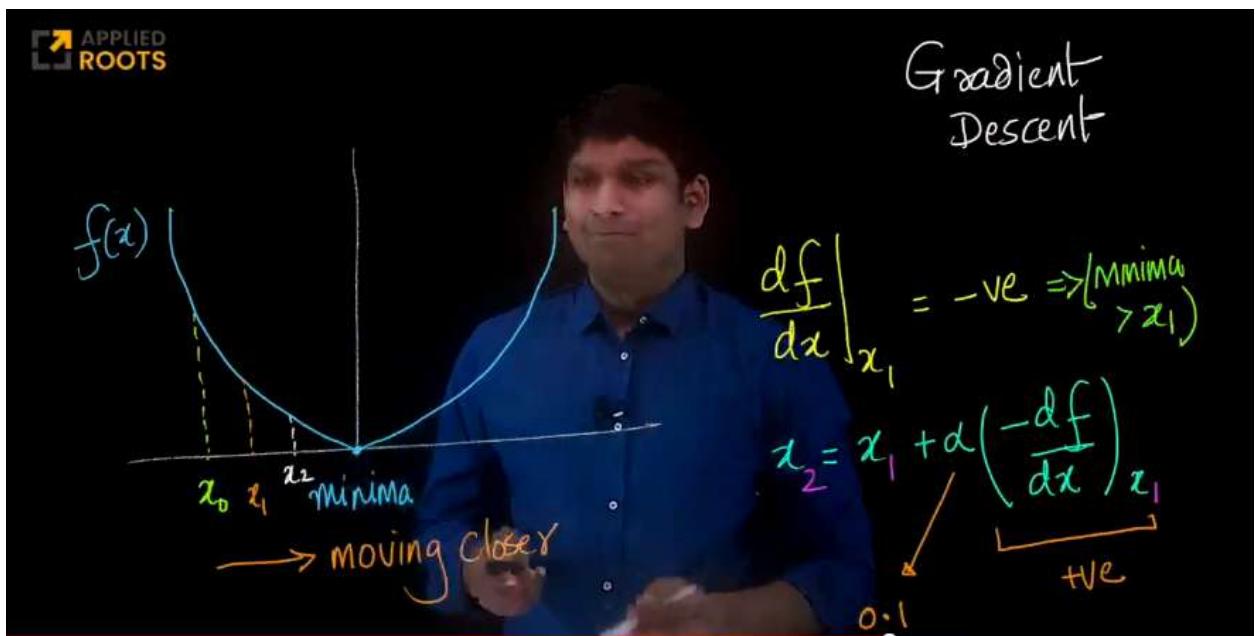
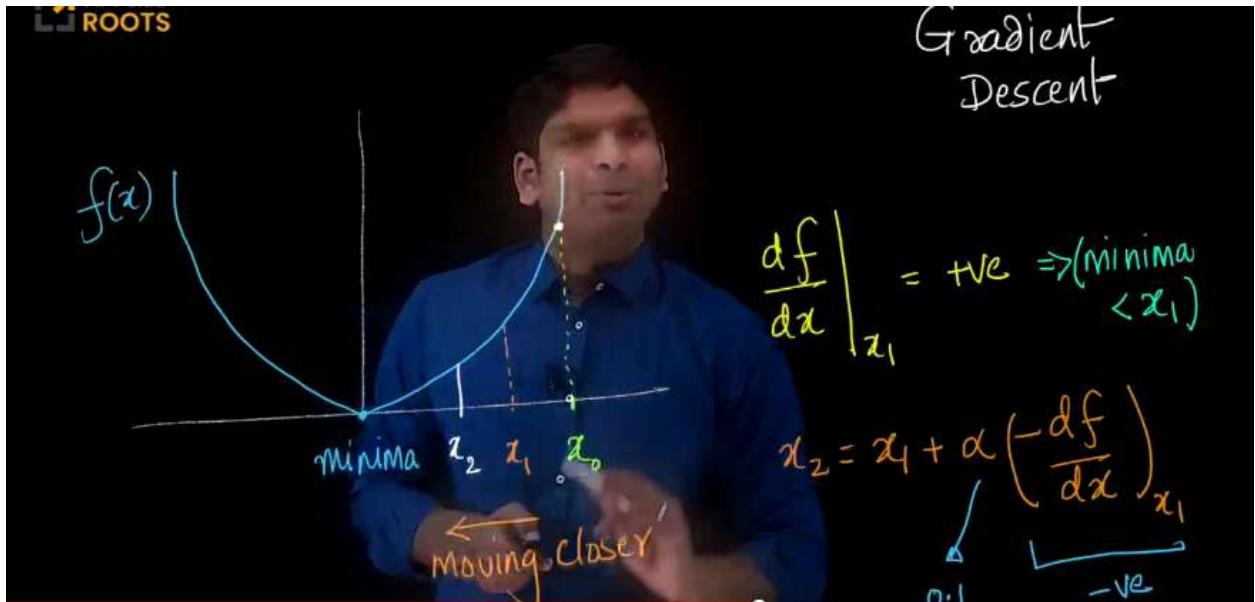
At timestamp 3.48



- Let's consider the above curve and we want to find the minima ,we consider a random point x_0 and find the derivative at that point .Since the derivative is positive we know that the function is increasing and $\text{minima} < x_0$.

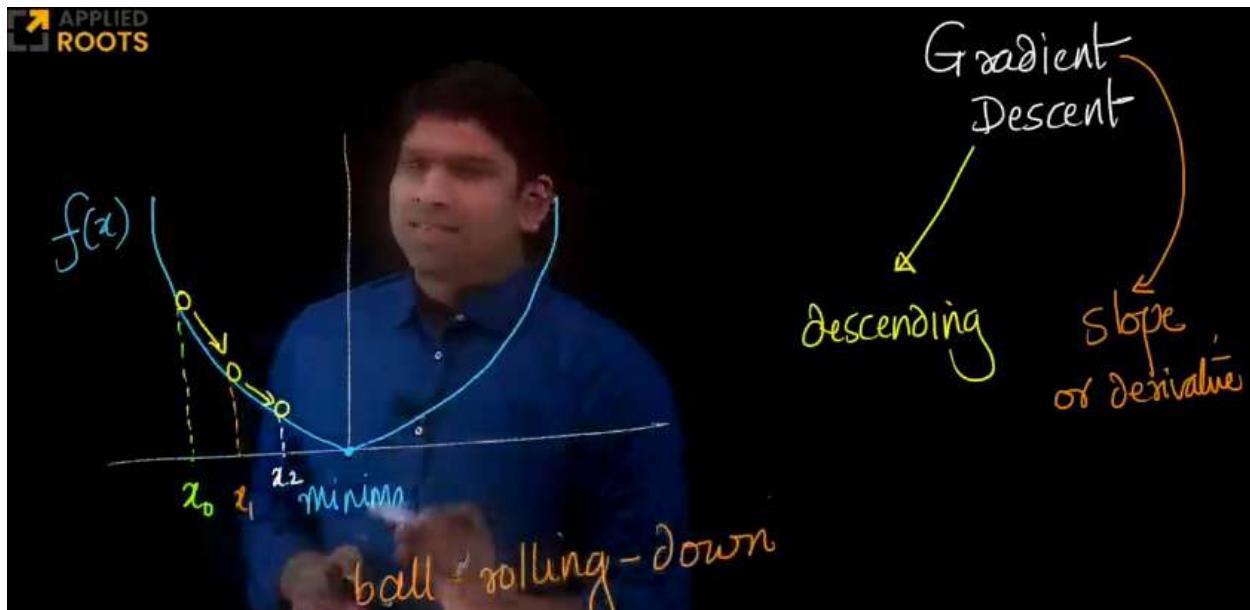


- We know that we have to move towards minima and we do this by using the above equation.



- We keep on updating x using the equation until we reach the minima as shown.

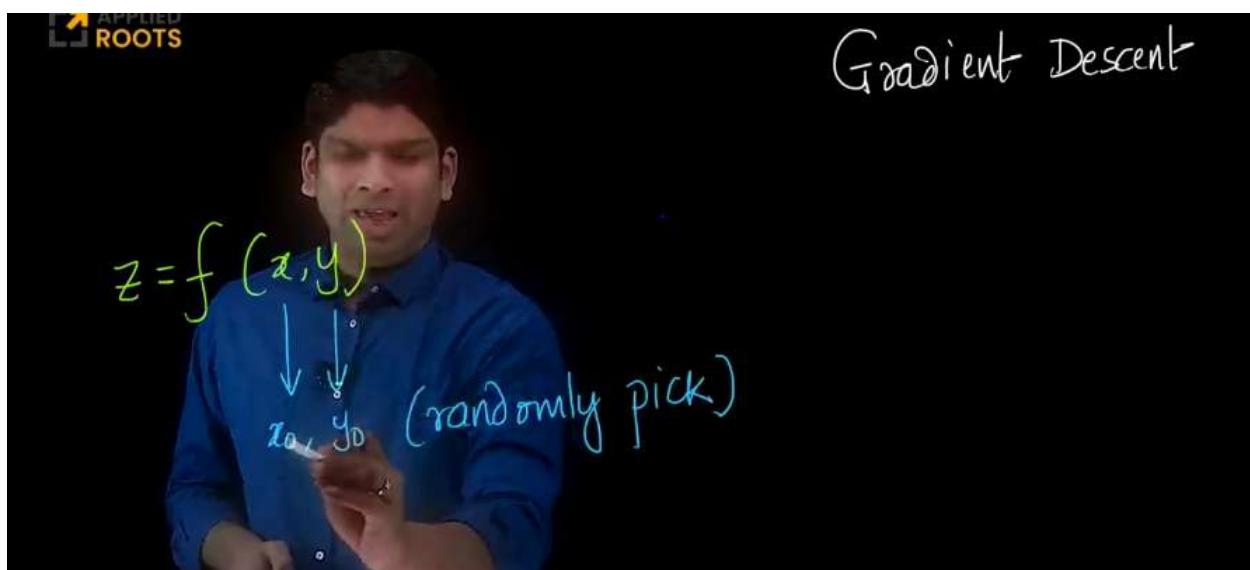
At timestamp 13.45

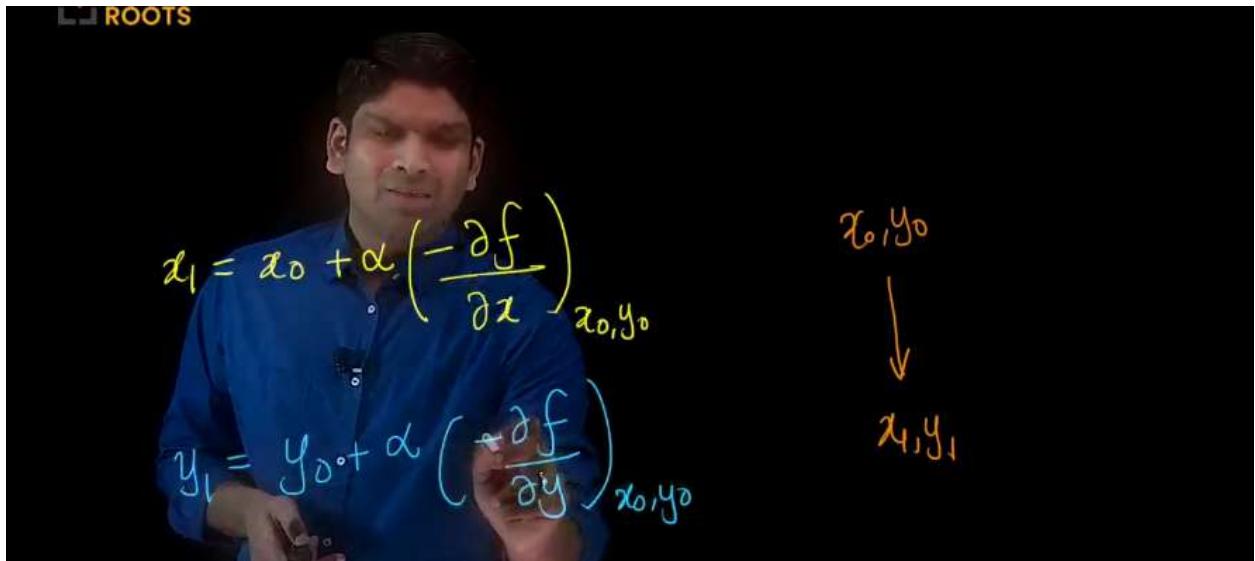


- The algorithm is called gradient descent because we are using gradient or slope or derivative and we are slowly descending towards minima by starting at a random point on the curve.

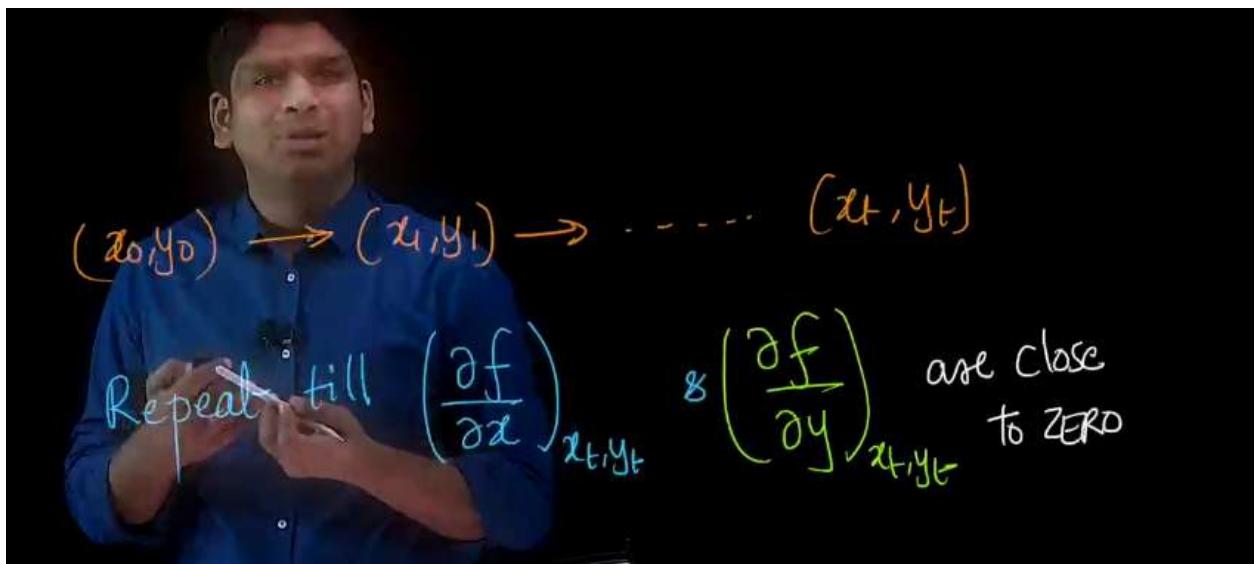
3.19 Gradient Descent with multiple variables

At timestamp 0.23





- We are trying to understand gradient descent with multiple variables as shown above



- We start with x_0, y_0 (we randomly choose them) and we keep on updating the values using the above equation until we get partial derivatives as close to zero as possible, then we consider that we have reached the minima.

3.20 Regression using Gradient Descent

At timestamp 0.23

Regression problem

$$w^*, b^* = \min_{w,b} \sum_{i=1}^n (y_i - (w^T x_i + b))^2$$

w^* , b^* y_i x_i

d-dim scalar in D (scalar) in D (d-dim)

$$\mathcal{D} = \{(x_i, y_i)\}$$

- Let's solve our regression problem using gradient descent .We arrived at the above optimization problem where we have to find optimal w,b such that our sum of squared distances should be as minimum as possible.
- Given dataset $D(x_i,y_i)$,The problem at hand is minimizing the above function

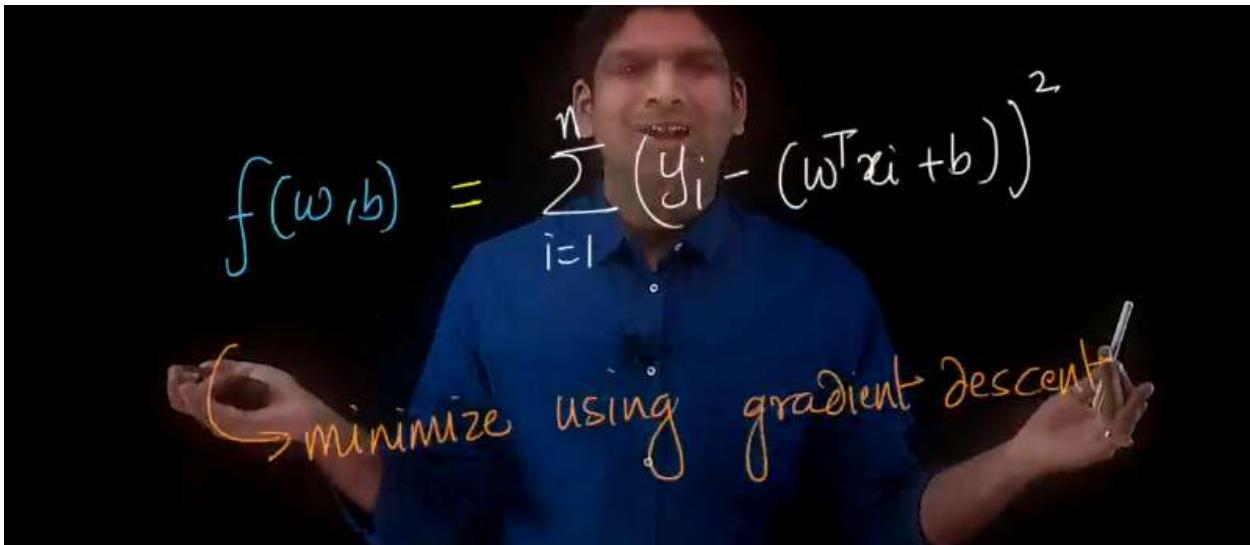
$$\sum_{i=1}^n (y_i - (w^T x_i + b))^2 = f(w, b) = f(w_1, w_2, \dots, w_d, b)$$

$\sum_{i=1}^n (y_i - (w^T x_i + b))^2$

function with $(d+1)$ variables

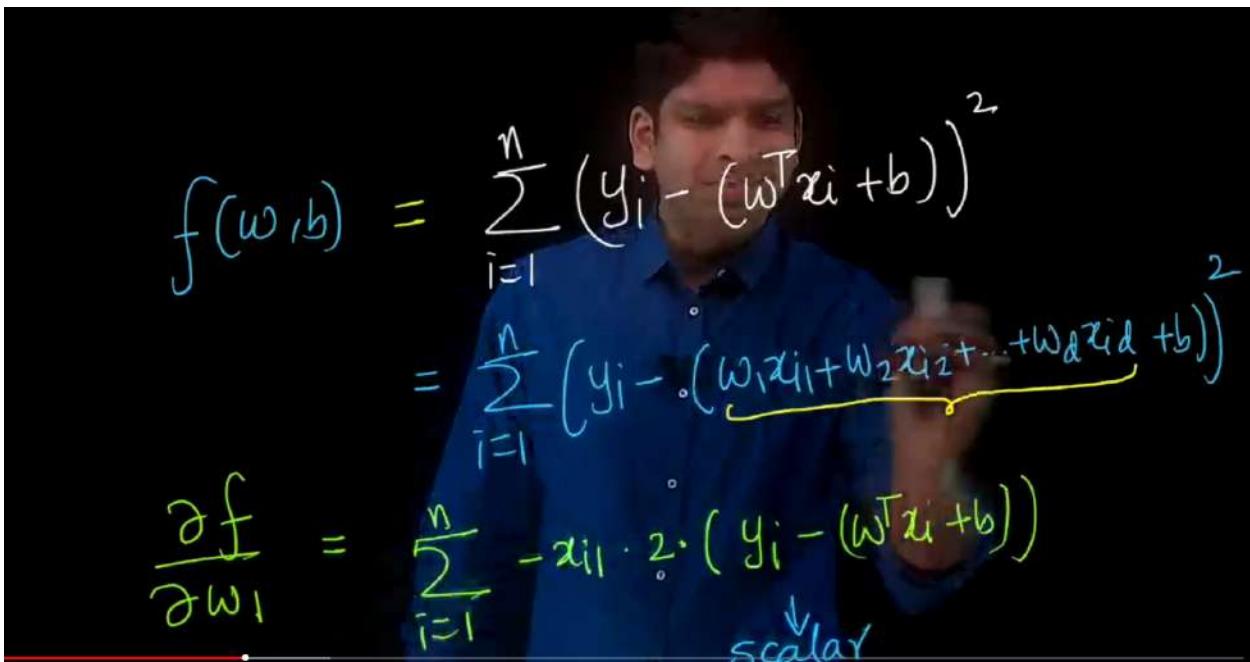
- We can write this as function of vector W and scalar b

At timestamp 3.40



- We can use gradient descent here since we have two variables w, b and we have to find the minima.

At timestamp 4.16



$$\begin{aligned}
 f(w, b) &= \sum_{i=1}^n (y_i - (w^T x_i + b))^2 \\
 &= \sum_{i=1}^n \left(y_i - \underbrace{(w_1 x_{i1} + w_2 x_{i2} + \dots + w_d x_{id} + b)}_2 \right)^2
 \end{aligned}$$

$$\frac{\partial f}{\partial b} = \sum_{i=1}^n (-1) \cdot 2 (y_i - (w^T x_i + b))$$

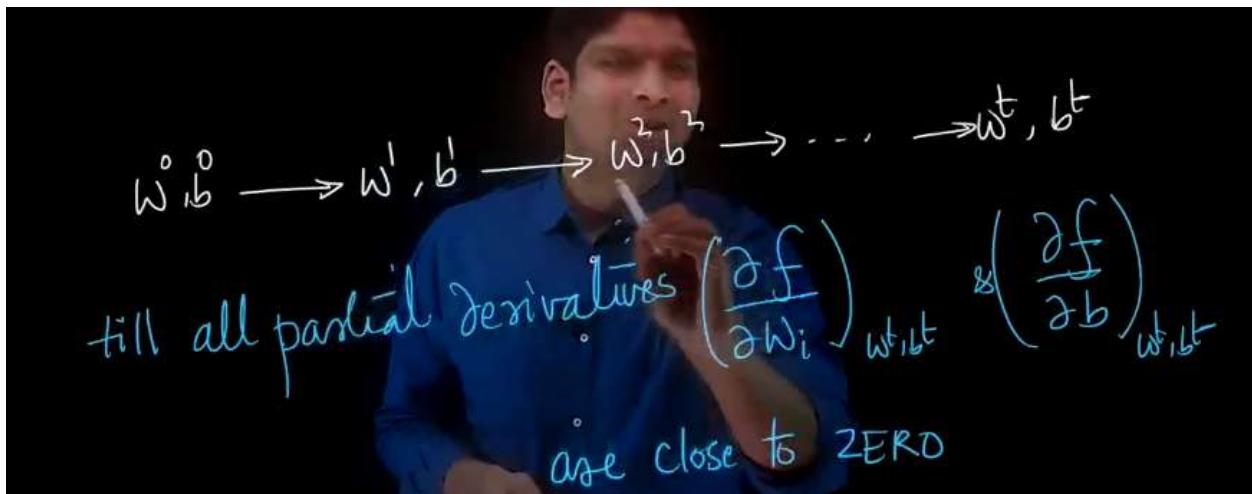
- We calculate the partial derivatives as shown above.

$\overbrace{w^0, w_1^0, w_2^0, \dots, w_d^0, b^0} \rightarrow \text{pick randomly}$

$$\begin{aligned}
 w_i^1 &= w_i^0 + \alpha \left(-\frac{\partial f}{\partial w} \right)_{w^0, b^0} \\
 b^1 &= b^0 + \alpha \left(-\frac{\partial f}{\partial b} \right)_{w^0, b^0}
 \end{aligned}$$

$$w = [w_1^0, w_2^0, \dots, w_d^0]$$

At timestamp 12.42



- Initially we randomly pick w vector and b , then we use gradient descent and keep on update the values and move towards the minima as shown above.
- We use gradient descent even when we have more than two variables. So given a regression problem we can find the best line/plane using gradient descent while minimising the loss.

4.4 Keywords and Identifiers

Keywords

- Keywords are the reserved words in Python and cannot be used as the names of variables, functions, classes and other identifiers.
- There are 35 keywords in Python and all of them are case sensitive. (Till Python 3.6 version, there were 33 keywords, but two more got added from 3.7 version onwards)
- Keywords are always available and we need not import them explicitly. We can use them directly in our code whereas the in-built functions, modules have to be imported explicitly before use.
- The statement `import keyword` is used to import the **keyword** class. The statement `print(keyword.kwlist)` prints the list of the available keywords in Python.

```
import keyword

#print(keyword.kwlist)
"Total number of keywords ", keyword.kwlist

: ('Total number of keywords ',
 ['and',
 'as',
 'assert',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'exec',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
 'is',
 'lambda',
 'not',
 'or',
 'pass',
 'print',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield'])
```

Note: We are importing the **keyword** module, just to check the list of available keywords, which is possible only by accessing the **kwlist()** method of **keyword** class.

- If we want to access any method/variable of a class, we first have to import the class.
- If you want to use any one of these 35 keywords in your code, you need not import the **keyword** module. You can directly use them.

Identifiers

- Identifier is the name given to the entities like classes, functions, variables, etc. in Python. It helps in differentiating one entity from the other.

Rules for writing an Identifier

- 1) Identifiers can be a combination of lowercase letters (a-z) or uppercase letters (A-Z) or digits (0-9) or an underscore (_).
- 2) An identifier cannot begin with a digit, but can contain digits in the middle. (ie., the first character in the identifier should never be a digit and we can use digits anywhere else in the identifier except the first position.)
- 3) Keywords cannot be used as the identifiers. Below is an example shown at the timestamp 3:43 in the video.

```
: global = 1
  File "<ipython-input-3-d0026cf49b71>", line 1
    global = 1
    ^
SyntaxError: invalid syntax
```

- 4) Apart from underscore (_), no special character should be used in an identifier. Below is an example shown at the timestamp 5:22 in the video.

We cannot use special symbols like !, @, #, \$, % etc. in our identifier.

```
: a@ = 10          #can't use special symbols as an identifier
  File "<ipython-input-5-b512271f00c8>", line 1
    a@ = 10          #can't use special symbols as an identifier
    ^
SyntaxError: invalid syntax
```

4.5 Comments, Indentation and Statements

Comments

- Comments are the lines written in the code and are not executed by the compiler and the interpreter.
- Comments are in general used to describe which part of code is doing what job.
- Comments make the code look readable and understandable by the humans and it is always a good practice to write comments in the code because when we write the code and return back after sometime, we may

not be able to understand and we may not remember the thought process followed while writing the code. Hence writing the comments make the code much more understandable.

- The comments should always begin with the symbol '#' and once if the interpreter/compiler encounters the '#' symbol in the code, the statement from that position, till the end of that line will be ignored and will not be executed.

Multi-line Comments

In case, if we come across a situation where we have to write too lengthy comments, you can split it into more than one line. We have two approaches.

For example, the comment is "**Below given example is the most appropriate one for recursion**".

So the 2 ways to write the comment in multiple lines are

Approach 1

```
# Below given example  
# is the most appropriate one  
# for recursion
```

Approach 2

```
"""Below given example  
is the most appropriate one  
for recursion"""
```

Out of both these ways, the first one is mostly used by the programmers.

DocString

- DocString is the shortest description of a function that has to be written as the first line inside the function block. The docstring should always be the first line inside the function.
- The docstring has to be enclosed within triple quotes.

Example

```
def add(a,b):
```

```
    "This function performs the addition of two numbers." # This is the doc-string.
```

```
return a+b
```

Syntax to access the doc string of above function

```
print(add.__doc__)
```

Indentation

- Most of the programming languages like C++, Java use braces to define a block of code. But Python doesn't support the concept of using braces to define code blocks. In place of these braces, it supports Indentation (ie., whitespaces).
- If we find any line of code beginning with whitespaces, then it means a new block of code has begun and also this new block ends with the first unindented line.
- Generally one tab space or four white spaces are considered for indentation.

Example

```
a = 15
b = 8
if a>b:
        print('a is greater')
else:
        print('b is greater')
```

In the above example, as the **print()** begins with whitespaces, it means from there onwards a new block begins and the first unintended line is the '**else**' statement.

Example

```
for i in range(10,100):
        if i%10==0:
                print('{} is divisible by 10'.format(i))
        print('Done with looping')
```

In the above example, we could see a block within another block and the first unindented line is the line with the 2nd print statement. If we miss any of these indentations, it throws an error.

Statements

- Python Statements are the instructions passed to the Python interpreter for execution.
- Comments do not come under the category of statements as they are not executed by the Python interpreter.

Example

a = 25

b = 10

These two above initializations are the statements and when we pass these instructions to the Python interpreter, then the two variables 'a' and 'b' get created in the memory.

Example

c = a + b

d = a - b

The above 2 statements perform addition and subtraction of two numbers respectively.

Multi-line Statements

We can write a single statement in multiple lines.

Example

**a = 1 + 2 + 3 + **

**4 + 5 + 6 + **

7 + 8 + 9

Here it is one statement, but written in 3 lines. Whenever we use this kind of syntax, we have to use the '\' symbol to let the interpreter understand that the given statement is written in multiple lines.

b = (1 + 2 + 3 +

4 + 5 + 6 +

7 + 8 + 9)

Here the one statement is written in 2 lines. The other way to write the same code without using the ‘\’ symbol is to enclose all the numbers in small braces ‘()’.

4.6 Variables and Datatypes in Python

Variables

- A Variable is a location in the memory and is used to store some value.
- Each variable has a unique name so that it could be differentiated from the other memory locations. The rules for naming a variable are same as that of an identifier.
- While initializing a variable, we just have to assign a value. Unlike in programming languages like C/C++/Java, we need not mention any data type for the variable. Deciding the data type and allocation of memory is done internally, after we assign a value.

Variable Assignment

We use the ‘=’ operator to assign values to a variable. The below example was discussed at the timestamp 0:55 in the video.

```
: #We use the assignment operator (=) to assign values to a variable  
a = 10  
b = 5.5  
c = "ML"
```

Example in Java/C

```
int a = 10  
float b = 5.5
```

In the first statement, we are declaring the datatype of the variable ‘a’ as ‘integer’ and assigning the integer 10 to it.

Similarly in the second statement, we are declaring the datatype of the variable ‘b’ as ‘float’ and assigning the decimal value 5.5 to it.

But when we take the same example in Python, the statement would be

```
a = 10  
b = 5.5  
c = 'ML'
```

In the first statement, after assigning the value 10 to the variable ‘a’, then it comes to know that ‘a’ is an integer variable.

Similarly after assigning the value 5.5 to the variable ‘b’, it comes to know that the variable ‘b’ is a float variable.

In the third statement, after assigning the value ‘ML’ to the variable ‘c’, it knows that the given variable is of string type and then allocates the memory accordingly.

Multiple Variable Assignments

Below is an example that has been discussed at the timestamp 2:40 in the video.

```
: a, b, c = 10, 5.5, "ML"  
:  
: a = b = c = "AI" #assign the same value to multiple variables at once
```

The same previous example can be written as an example for multiple variable assignments as

a, b, c = 10, 5.5, 'ML'

The functionality is the same in both these types of variable assignments, but the main advantage with this multiple variable assignment approach is the **reduction in the length of the code**.

If we want to assign same value to multiple variables, then the syntax would be

a = b = c = 'AI'

Storage Locations

- The storage locations are the locations where the variables are stored. Each variable has a different and a unique storage location.
- Once after we assign a value to a variable, then immediately Python allocates a storage location to that variable. The storage location address is printed using the **id()** function in Python. Below is an example that has been discussed at the timestamp 3:30 in the video

```
|: x = 3  
print(id(x))          #print address of variable x  
140372891159288
```

```
|: y = 3  
print(id(y))          #print address of variable y  
140372891159288
```

Observation:

x and y points to same memory location

```
|: y = 2  
print(id(y))          #print address of variable y  
140372891159312
```

Example

x = 3

y = 3

print(id(x)) # prints the address of the variable 'x'

o/p: 140372891159288

print(id(y)) # prints the address of the variable 'y'

o/p: 140372891159288

Here both 'x' and 'y' are pointing to the same location. Hence we got the same value.

Note: For integer data, if you declare two or more variables and assign any one value in the interval [-5,256] to all the variables, then all those variables will point to the same location.

If any value outside the interval [-5,256] is chosen and is assigned, then they all point to different locations, as each of those variables will have unique locations.

Example

x = 3

y = 2

print(id(x)) # prints the address of the variable 'x'

o/p: 140372891159288

print(id(y)) # prints the address of the variable 'y'

o/p: 140372891159312

Here as both 'x' and 'y' are having two different values, they both point to two different locations.

Data Types

- Every value in Python has a data type.
- Since Python is an object oriented programming language, every data type is a predefined class in Python and every variable is an instance of these predefined classes.

Numbers

- Integers, floating point numbers and complex numbers fall under the Python numbers category.
- These numbers are classified into int, float and complex classes respectively in Python.
- We have to use type() function to know which class a variable/value belongs to.
- We have to use isinstance() method to check if a given object belongs to a particular class or not.

Below is an example that has been discussed at the timestamp 6:15 in the video.

```
: a = 5                                     #data type is implicitly set to integer
print(a, " is of type", type(a))

(5, ' is of type', <type 'int'>)

: a = 2.5                                    #data type is changed to float
print(a, " is of type", type(a))

(2.5, ' is of type', <type 'float'>)

: a = 1 + 2j                                 #data type is changed to complex number
print(a, " is complex number?")
print(isinstance(1+2j, complex))

((1+2j), ' is complex number?')
True
```

Boolean Data Type

- Boolean data type represents the truth values **True** and **False**.
- It doesn't take any value other than these two.

- Also **True** and **False** are not only the values of boolean data type, but also are the keywords in Python.

Example

```
a = True  
b = False  
print(type(a)) # prints the data type of the variable 'a'  
o/p: <type 'bool'>  
print(type(b)) # prints the data type of the variable 'b'  
o/p: <type 'bool'>
```

Strings

- String is a sequence of Unicode characters.
- A string has to be enclosed either between single quotes or double quotes.
- Multi-line strings have to be enclosed between triple quotes. You can either use "" (or) """.
- A string in Python consists of a series or a sequence of characters. It can have characters, numbers and special characters.
- Strings in Python also support indexing and we can access each character using an index. The first index would be 0.

Below is an example that has been discussed at the timestamp 9:30 in the video.

```
s = "This is Online AI course"  
print(s)
```

```
This is Online AI course
```

```
print(s[0])  
#last char s[len(s)-1] or s[-1]
```

```
T
```

```
#slicing  
s[5:]
```

```
'is Online AI course'
```

Example

```
s = "This is Online AI course"  
print(s[0]) # Prints the character present in 0th index  
print(s[5:]) # Prints the substring starting from the 5th index till the end.  
print(s[5:10]) # Prints the substring starting from the 5th index till the 9th index.
```

Python List

- List is an ordered sequence of items.
- It is one of the most used data types in Python and is very flexible.
- All the items in a list do not need to be of the same data type.
- Declaring a list is nothing but the items separated by commas are enclosed within brackets []
- We can access each of the elements in the list using their indexes.
- Lists are mutable. It means the elements in the list can be changed as per our requirement whenever needed.

Below are the examples that are discussed in the video at the timestamp 15:45

```
a = [10, 20.5, "Hello"]  
print(a[1])           #print 1st index element  
  
20.5
```

Lists are mutable, meaning, value of elements of a list can be altered.

```
a[1] = 30.7  
print(a)  
  
[10, 30.7, 'Hello']
```

In the first example, we can see how we are able to access the elements of the list using their indexes.

In the second example, we can see how we can manipulate/modify the values in the list.

Python Tuple

- Tuple is an ordered sequence of elements, the same as a List.
- The only difference between a List and a Tuple is the List is mutable whereas a Tuple is immutable. It means once if a tuple is created, then the elements in it could not be modified.
- Also the elements in a List are enclosed within square brackets (ie., []) whereas the elements in a tuple should be enclosed between a pair of round brackets (ie., ())

Below is an example that has been discussed at the timestamp 18:15 in the video.

```
: t = (1, 1.5, "ML")
: print(t[1]) #extract particular element
1.5
: t[1] = 1.25
-----
TypeError                                     Traceback (most recent call last)
<ipython-input-18-0ddc671fae60> in <module>()
----> 1 t[1] = 1.25

TypeError: 'tuple' object does not support item assignment
```

Python Set

- Set is an unordered collection of unique items.
- A set is defined by the values separated by command inside the curly braces (ie., {})
- Items in a set are unordered. The order of the elements in a set is decided by an internal mechanism.
- Sets do not allow repetition of elements and also sometimes the order in which the elements are present in the set might differ from the order in which we have inserted.
- We can perform operations like subtraction, union and intersection between two sets.

Below is an example that has been discussed at the timestamp 20:40 in the video.

```
: a = {10, 30, 20, 40, 5}
print(a)
set([40, 10, 20, 5, 30])

: print(type(a))          #print type of a
<type 'set'>

s = {10, 20, 20, 30, 30, 30}
print(s)                  #automatically set won't consider duplicate elements
{10, 20, 30}

print(s[1]) #we can't print particular element in set because
            #it's unorder collections of items

-----
TypeError: Traceback (most recent call last)
<ipython-input-23-ad7511dba6cd> in <module>()
----> 1 print(s[1]) #we can't print particular element in set because
      2         #it's unorder collections of items

TypeError: 'set' object does not support indexing
```

In the above example, the first cell clearly shows us that the order in which the elements are inserted is different from the order in which they are displayed.

The third cell clearly shows us that a set doesn't allow repetition of elements. It means irrespective of how many times an element occurs, it will be stored only once in the set.

The fourth cell clearly shows us that a set object doesn't support indexing.

Python Dictionary

- Dictionary is an unordered collection of key-value pairs.
- Same like a set, the elements in a dictionary are also enclosed between curly braces, but the major difference here is the elements in a dictionary are present in **key-value pair** format.
- The order in which the key-value pairs are added to a dictionary is sometimes different from the order in which they appear.
- Dictionary doesn't allow duplication of keys, but allows duplication of values. It means a key once used should not be used again for another value, whereas the value once used can be used again for another key.

- The term present to the left side of the colon(:) is the **key** and the term present to the right side of the colon(:) is the value. There should be no keys repeated.

Below is an example discussed at the timestamp 24:25 in the video.

```
|: d = {'a': "apple", 'b': "bat"}  
|: print d['a']
```

```
apple
```

```
: d = {'a': "apple", 'b': "bat"}  
: print(d['c'])
```

```
---  
KeyError                                Traceback (most recent call last)  
t)  
<ipython-input-60-744a428435ea> in <module>()  
    1 d = {'a': "apple", 'b': "bat"}  
----> 2 print(d['c'])  
  
KeyError: 'c'
```

If the entered '**key**' is not present in the dictionary, then it throws an error as shown above.

Conversion between Data Types

We can perform conversions from one data type to another using different available type casting functions like int(), float(), str(), etc.

Below are the examples that were discussed at the timestamp 26:56

```
float(5)      #convert interger to float using float() method
```

```
5.0
```

```
int(100.5)   #convert float to integer using int() method
```

```
100
```

```
str(20)      #convert integer to string
```

```
'20'
```

Conversion to and from string must contain compatible values.

```
: int('10p')
-----
ValueError                                Traceback (most recent call last)
<ipython-input-27-b6ad1e4c556d> in <module>()
----> 1 int('10p')

ValueError: invalid literal for int() with base 10: '10p'
```

```
: user = "satish"
lines = 100

print("Congratulations, " + user + "! You just wrote " + str(lines) + " lines of code")
#remove str and gives error
```

Congratulations, satish! You just wrote 100 lines of code

We can convert one sequence to other

```
: a = [1, 2, 3]

print(type(a))      #type of a is list

s = set(a)          #convert list to set using set() method

print(type(s))      #now type of s is set

<type 'list'>
<type 'set'>

: list("Hello")     #convert String to List using List() method

: ['H', 'e', 'l', 'l', 'o']
```

4.7 Standard Input and Output

Python Output

In order to display the result, we use the **print()** function.
Below are the examples starting from the timestamp 0:10 in the video.

```
: print("Hello World")
Hello World

: a = 10
print("The value of a is", a) #python 3
print "The value of a is " + str(a)

('The value of a is', 10)
The value of a is 10
```

Both the above print statements give the same result, but in the first one, we are printing a string and an integer side by side whereas in the second statement we are converting the integer into a string and then concatenating both the strings and are printing the result.

Output Formatting

The below examples begin from the timestamp 1:55 in the video.

```
: a = 10; b = 20 #multiple statements in single line.

print("The value of a is {} and b is {}".format(a, b))    #default

The value of a is 10 and b is 20
```

The above statement prints the string with the values of 'a' and 'b' in place of those braces in the same order as they are present in **format()** method.

```
: a = 10; b = 20  #multiple statements in single line

print("The value of b is {1} and a is {0}".format(a, b)) #specify position of arguments

The value of b is 20 and a is 10
```

In this statement, we are displaying the same result, but here instead we are using the indexes of the values in **format()** method. The index of 'a' is 1 and the index of 'b' is 0. Values associated with those variables present in those indexes will be displayed in place of these braces.

```
#we can use keyword arguments to format the string
print("Hello {name}, {greeting}".format(name="satish", greeting="Good Morning"))

Hello satish, Good Morning
```

In this statement, as we are passing the keyword arguments in **format()** method, we have to pass the keys into these braces. In our example, the keys are 'name' and 'greeting'.

```
#we can combine positional arguments with keyword arguments
print('The story of {0}, {1}, and {other}'.format('Bill', 'Manfred',
                                                other='Georg'))
```

The story of Bill, Manfred, and Georg

In this statement, we are passing positional as well as keyword arguments to the **format()** function. One thing to be remembered seriously here is that the keyword arguments should always come after the positional arguments in **format()** method.

Python Input

We have the **input()** function in Python which is used to take the input from the user.

The below example was discussed at the timestamp 5:25 in the video.

```
: num = input("Enter a number: ")
print num

Enter a number: 10
10
```

4.8 Operators

- Operators are the special symbols in Python used for performing mathematical and logical computations.
- The numbers on which the operations are performed are called **operands**

Types of Operators

- 1) Arithmetic Operators
- 2) Relational Operators (Comparison Operators)
- 3) Logical Operators (Boolean Operators)
- 4) Bitwise Operators
- 5) Assignment Operators
- 6) Special Operators

Arithmetic Operators

Arithmetic Operators are used to perform operations like addition, subtraction, multiplication, division, floor division, modulo division, exponent, etc.

Below example begins at the timestamp 0:40.

```
]: x, y = 10, 20
#addition
print(x + y)

#subtraction(-)

#multiplication(*)

#division(/)

#modulo division (%)

#Floor Division (//)

#Exponent (**)
```

Example

a, b = 10 ,20

```
print(x + y) # Prints the sum.  
print(x - y) # Prints the difference.  
print(x * y) # Prints the product.  
print(x / y) # Prints the division.  
print(x % y) # Prints the modulo division.  
print(x // y) # Prints the floor division.  
print(x ** y) # Prints the exponent value.
```

Comparison Operators (Relational Operators)

- Comparison/Relational operators are used to compare two given values. They return only Boolean values. (i.e., either **True** or **False**)
- **>**, **<**, **==**, **!=**, **>=**, **<=** are comparison operators.

Below example begins at the timestamp 4:35 in the video.

```
: a, b = 10, 20  
  
print(a < b) #check a is less than b  
  
#check a is greater than b  
  
#check a is equal to b  
  
#check a is not equal to b (!=)  
  
#check a greater than or equal to b  
  
#check a less than or equal to b
```

True

Example

a, b = 10, 20

```
print(a<b) # Prints True if a is less than b. Otherwise it prints False.  
print(a>b) # Prints True if a is greater than b. Otherwise it prints False.  
print(a==b) # Prints True if a is equal to b. Otherwise it prints False.  
print(a!=b) # Prints True if a is not equal to b. Otherwise it prints False.
```

```
print(a>=b) # Prints True if a is greater than or equal to b. Otherwise it prints False.
```

```
print(a<=b) # Prints True if a is less than or equal to b. Otherwise it prints False.
```

Logical Operators

Logical Operators are **AND**, **OR** and **NOT** operators. Even these operators give only Boolean values as the result.

The below examples begin at timestamp 5:15.

```
: a, b = True, False  
  
#print a and b  
print(a and b)  
  
#print a or b  
  
#print not b
```

```
False
```

Example

a, b = True, False

```
print(a and b)  
print(a or b)  
print(not b)
```

Bitwise Operators

- Bitwise Operators are those operators which act on the operands as if the operands are the strings of binary digits.
- These operators operate bit by bit.
- The bitwise operators are &, |, ~, ^, >>, <<

Types of Bitwise Operations

- 1) Bitwise AND (&)
- 2) Bitwise OR (|)
- 3) Bitwise NOT (~)
- 4) Bitwise XOR (^)
- 5) Bitwise Right Shift (>>)
- 6) Bitwise Left Shift (<<)

```
a, b = 10, 4

#Bitwise AND
print(a & b)

#Bitwise OR

#Bitwise NOT

#Bitwise XOR

#Bitwise rightshift

#Bitwise Leftshift
```

0

The above example begins at the timestamp 6:10

Example

```
a,b = 10,4
print(a&b) # Bitwise AND
print(a|b) # Bitwise OR
print(~b) # Bitwise NOT
print(a^b) # Bitwise XOR
print(a>>2) # Bitwise RIGHT SHIFT
print(a<<3) # Bitwise LEFT SHIFT
```

Assignment Operators

Assignment operators in python are used to assign values to variables.

Types of Assignment Operations

- 1) Simple Assignment (=)
- 2) Addition AND (+=)
- 3) Subtraction AND (-=)
- 4) Multiplication AND (*=)
- 5) Division AND (/=)
- 6) Modulo Division AND (%=)
- 7) Exponent AND (**=)

Below example was discussed at the timestamp 8:45 in the video.

```
: a = 10
a += 10          #add AND
print(a)

#subtract AND (-=)

#Multiply AND (*=)

#Divide AND (/=)

#Modulus AND (%=)

#Floor Division (//=)

#Exponent AND (**=)
```

20

Special Operators

A. Identity Operators

Identity operators are used to check if the given two operands are pointing to the same location.

There are only two identity operators in Python. They are **is** and **is not**.

Below examples have been discussed starting from the timestamp 9:55 in the video.

```
a = 5
b = 5
print(a is b)      #5 is object created once both a and b points to same object

#check is not

True
```

In the above example, the ‘is’ operator is checking if both ‘a’ and ‘b’ are pointing to the same object. As they both are pointing to the same object, it is giving the result as **True**.

```
l1 = [1, 2, 3]
l2 = [1, 2, 3]
print(l1 is l2)

False
```

If two lists have the same elements in the same order, it doesn’t mean they both are pointing to the same object. Hence it is giving the result here as **False**.

```
s1 = "Satish"
s2 = "Satish"
print(s1 is not s2)

False
```

When two strings have the same value and if there are no whitespaces in the strings, then they both point to the same object. Otherwise, they both point to two different objects.

B. Membership Operators

- Membership operators are used to check if a given object is a member of a given data structure or not.
- The two membership operators in Python are **in** and **not in**.

```
lst = [1, 2, 3, 4]
print(1 in lst)      #check 1 is present in a given List or not

#check 5 is present in a given list
```

```
True
```

In the above example, it is checking if the number 1 is present in the list 'lst'. As it is present, the result became **True**.

In case of lists, tuples, sets it checks if the given element is present in those data structures or not. Whereas in case of dictionaries, it checks if the given element/value is present as the key in it. If yes, then it returns **True**. Otherwise it returns **False**.

```
: d = {1: "a", 2: "b"}
print(1 in d)
```

```
True
```

In this example, as the number 1 is present as one of the keys in the dictionary 'd', it returns **True**.

4.9 Control Flow: if else

Python if-else statement

The **if-elif-else** statements are used in python for decision making.

Syntax

```
if <test expression>:  
    statements(s)
```

If the given test expression gives the result as **True**, then only the statement(s) associated with this block are executed.

If the given test expression gives the result as **False**, then the statement(s) associated with this block are not executed.

If the test expression gives any non boolean value as the result, then the values **0** and **None** are interpreted as **False** and all other non zero and non boolean values are interpreted as **True**.

Below example was discussed from the timestamp 3:25 in the video.

```
: num = 10  
if num > 0:  
    print("Positive number")  
else:  
    print("Negative Number")  
  
Positive number
```

In the above example, we are checking if the given number is greater than 0 or not. If yes, then it prints the statement in the '**if**' block. Otherwise, it prints the statement in the '**else**' block.

if-elif-else statement

Syntax

```
if <test-expression-1>:  
    statement(s)  
  
elif <test-expression-2>:  
    statement(s)  
  
else:  
    statement(s)
```

Below is an example discussed at the timestamp 4:15 in the video.

```
num = 0  
  
if num > 0:  
    print("Positive number")  
elif num == 0:  
    print("ZERO")  
else:  
    print("Negative Number")
```

ZERO

In the above example, if the given ‘num’ is greater than 0, then the statement in the ‘if’ block gets executed. If ‘num’ is equal to 0, then the statement in the ‘elif’ block gets executed. If both the conditions fail, then the statement in the ‘else’ block gets executed.

```
num = 10.5

if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative Number")
```

Positive number

```
num1 = 10
num2 = 50
num3 = 15

if (num1 >= num2) and (num1 >= num3):           #logical operator and
    largest = num1
elif (num2 >= num1) and (num2 >= num3):
    largest = num2
else:
    largest = num3
print("Largest element among three numbers is: {}".format(largest))
```

Largest element among three numbers is: 50

In the above program, we are finding out the largest number among the given 3 numbers.

Here we are initializing num1 = 10, num2 = 50, num3 = 15.

If ‘num1’ is greater than or equal to ‘num2’ and if ‘num1’ is greater than or equal to ‘num3’, then the statement in the ‘if’ block gets executed.

If ‘num2’ is greater than or equal to ‘num1’ and if ‘num2’ is greater than or equal to ‘num3’, then the statement in the ‘else’ block gets executed. Otherwise the ‘else’ block statement will get executed.

4.10 Control Flow: while

The while loop in Python is used to iterate over a block of code as long as the test expression is true.

Syntax

```
while test-expression:  
    Body of while
```

The ‘Body of while’ get executed only if the test-expression is evaluated as True. After one iteration, the test-expression is again checked and if it again returns **True**, then again the same ‘Body of while’ will get executed. This looping continues until the test-expression returns **False**.

Below is an example that is shown at timestamp 1:36 in the video.

```
#Find product of all numbers present in a List  
  
lst = [10, 20, 30, 40, 60]  
  
product = 1  
index = 0  
  
while index < len(lst):  
    product *= lst[index]  
    index += 1  
  
print("Product is: {}".format(product))
```

```
Product is: 14400000
```

In this example, we want to compute the product of all the values in the given list. We are traversing through the list with the help of indexes.

So we are starting the index from 0 and are initializing the variable ‘product’ to 1 which is used to store the result and are checking if the index is less than the length of the list. If the condition is satisfied, we are the element at that index position is multiplied to the variable ‘product’ and we are incrementing the index by 1.

This process keeps repeating until the test condition fails and finally we are printing the result which is the product of all the numbers in the given list.

while loop with else

We can have an optional ‘else’ block for a ‘while’ loop. This ‘else’ block gets executed only if the test condition in the while loop returns **False**.

We also can break the while loop using ‘break’ statement in the body of the loop, but when this ‘break’ statement gets executed, then the control comes out of the loop ignoring this ‘else’ block.

So the ‘else’ block of a ‘while’ loop runs only if the test expression is returning **False** and if there is no ‘break’ statement.

Below is an example shown at the timestamp 6:30 in the given video

```
numbers = [1, 2, 3, 4, 5]

#iterating over the list
index = 0
while index < len(numbers):
    print(numbers[index])
    index += 1

else:
    print("no item left in the list")
```

In the above example, we are incrementing the ‘index’ variable after every iteration. The length of the given list is 5 and once if the value of the ‘index’ variable becomes 5, then the condition `index<len(numbers)` fails and the control comes out of the loop and then executes the ‘else’ block.

Note: If we forget to increment the ‘index’ variable in each iteration, then it runs into infinite looping.

Below is an example of checking whether a given number is a prime. This is explained at timestamp 9:50 in the given video.

```

num = int(input("Enter a number: "))           #convert string to int

isDivisible = False;

i=2;
while i < num:
    if num % i == 0:
        isDivisible = True;
        print ("{} is divisible by {}".format(num,i) )
    i += 1;

if isDivisible:
    print("{} is NOT a Prime number".format(num))
else:
    print("{} is a Prime number".format(num))

```

```

Enter a number: 33
33 is divisible by 3
33 is divisible by 11
33 is NOT a Prime number

```

Procedure:

In the above example, for a number to be a prime, it should not be divisible by any other number except 1 and itself. So if we assume the given number is 'N', then in the range 1 to N, 1 is the first value and 'N' is the last value. We have to check if any value in between these two values (i.e., >1 and $<N$), can divide 'N' leaving a remainder 0. If we are able to find any such value, then we have to declare the number 'N' as a non-prime, otherwise, we can declare it as a prime.

We are using an indicator variable '**isDivisible**' which is boolean to indicate whether the number is a prime or not.

First we are initializing the '**isDivisible**' variable to **False** before beginning the iteration process. In the iteration process, if we find any value (say 'i') where $N\%i==0$, then immediately we have to change the value of '**isDivisible**' variable to **True**.

After the completion of the iteration process, if the value of '**isDivisible**' remains **False**, then it means there is no value in the range that could divide the given number leaving a remainder of 0 and hence we can declare the number as a prime. Otherwise we have to declare it as a non-prime.

4.11 Control Flow: for

- The for loop in Python is used to iterate over sequences (like lists, tuples, etc) or other iterable objects.
- Iterating over a sequence is called a **Traversal**.

Syntax

```
for element in sequence:  
    Body of for
```

Here, element is the variable that takes the value of the item inside the sequence on each iteration. Loop continues until we reach the last item in the sequence.

Below example's explanation begins at the timestamp 0:55 in the video.

```
#Find product of all numbers present in a list  
  
lst = [10, 20, 30, 40, 50]  
  
product = 1  
#iterating over the list  
for ele in lst:  
    print(type(ele))  
    product *= ele  
  
print("Product is: {}".format(product))  
  
<class 'int'>  
<class 'int'>  
<class 'int'>  
<class 'int'>  
<class 'int'>  
Product is: 12000000
```

In the above example, the variable 'ele' denotes each element in the iterable 'lst' in every iteration, till the end of looping process .

range() function

- We can generate a sequence of numbers using the range() function.
- For example, range(10) will generate numbers from 0 to 9 (10 numbers).
- We can also define the start, stop and step size as range(start,stop,step size). step size defaults to 1 if not provided.

- This function does not store all the values in memory, it would be inefficient. So it remembers the start, stop, step size and generates the next number on the go.

Below examples are discussed in the video starting from the timestamp 3:50 onwards.

```
#print range of 10
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

This above code snippet is an example for iterating over a sequence of elements without storing them. Here as we haven't mentioned any 'step-size', the default step size would be 1.

```
#print range of numbers from 1 to 20 with step size of 2
for i in range(0, 20, 5):
    print(i)
```

```
0
5
10
15
```

This code snippet is an example of iterating over a sequence generated using `range()` function with a step size of 5.

For loop with else

- A `for` loop can have an optional `else` block as well.
- The `else` part is executed if the items in the sequence used in for loop exhausts.

- **break** statement can be used to stop a for loop. In such case, the **else** part is ignored. Hence, a for loop's else part runs if no break occurs.

The below example was discussed from the timestamp 9:50.

```
: numbers = [1, 2, 3]

#iterating over the list
for item in numbers:
    print(item)
else:
    print("no item left in the list")

1
2
3
no item left in the list
```

```
: for item in numbers:
    print(item)
    if item % 2 == 0:
        break
else:
    print("no item left in the list")

1
2
```

The first code snippet is an example of the scenario where there is no **break** statement and the iteration over the entire list is over and hence the **else** block got executed.

The second code snippet is an example of the scenario where we encounter a **break** statement and the iteration process gets terminated without completion. Hence the **else** block doesn't get executed.

Below is an example of the program to check if a given number is a prime using **for** loops only. The timestamp for the explanation of this code is 10:35.

```
: index1 = 20
index2 = 50

print("Prime numbers between {0} and {1} are :".format(index1, index2))

for num in range(index1, index2+1):      #default step size is 1
    if num > 1:
        isDivisible = False;
        for index in range(2, num):
            if num % index == 0:
                isDivisible = True;
        if not isDivisible:
            print(num);
```

```
Prime numbers between 20 and 50 are :
23
29
31
37
41
43
47
```

4.12 Control Flow: break and continue

In Python, the **break** and **continue** statements can alter the flow of any loop.

Loops iterate over a block of code until the test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking the test expression. The **break** and **continue** statements are used in these cases.

The below example was explained in the video starting from the timestamp 1:28

```

numbers = [1, 2, 3, 4]
for num in numbers:           #iterating over list
    if num == 4:
        break
    print(num)
else:
    print("in the else-block")
print("Outside of for loop")

```

```

1
2
3
Outside of for loop

```

In this code snippet, we see the loop running until the value of ‘num’ becomes equal to 4. Once after it reaches 4, then the condition **num==4** returns **True** and thereby the break statement gets executed and the control comes out of the loop and executes the statements outside this block.

Example for usage of break statement in Prime number program

```

: num = int(input("Enter a number: "))           #convert string to int

isDivisible = False;

i=2;
while i < num:
    if num % i == 0:
        isDivisible = True;
        print ("{} is divisible by {}".format(num,i) )
        break; # this line is the only addition.
    i += 1;

if isDivisible:
    print("{} is NOT a Prime number".format(num))
else:
    print("{} is a Prime number".format(num))

```

```

Enter a number: 16
16 is divisible by 2
16 is NOT a Prime number

```

continue statement

The continue statement deals with not performing any operation, just by allowing the next iteration to happen.

Generally while performing looping, for certain values/indexes if we do not want to perform any operation and just to allow the next iteration to take place, we use the **continue** statement.

```
#print odd numbers present in a list
numbers = [1, 2, 3, 4, 5]

for num in numbers:
    if num % 2 == 0:
        continue
    print(num)
else:
    print("else-block")
```

```
1
3
5
else-block
```

In the above code snippet, we want to print only the odd numbers. So while looping over an iterator object(here it is a list), if the element is odd, then we are printing it. Otherwise we are just going forward for the next iteration without performing any additional task using the **continue** statement.

5.1 Lists

Data Structure

- A data structure is a collection of elements that are stored in some way. The most commonly used data structure in Python is the sequence.
- Lists are the sequential data structures, as each element in it is placed in a sequential manner.
- List is a collection of items (strings or integers or float values or other lists)
- Lists are enclosed in [].
- Each item in the list has an assigned index value and we can directly access the elements using their respective indexes.
- Each item in the list is separated by commas and the order in which the items are stored is the same as the order in which they are inserted.
- Lists are mutable. It means the items in a list can be modified as per our requirements.

List Creation

Below example was discussed at the timestamp 1:22 in the video.

```
: emptyList = []

lst = ['one', 'two', 'three', 'four'] # List of strings

lst2 = [1, 2, 3, 4] #List of integers

lst3 = [[1, 2], [3, 4]] # List of lists

lst4 = [1, 'ramu', 24, 1.24] # List of different datatypes

print(lst4)

[1, 'ramu', 24, 1.24]
```

We can create an empty list either using a pair of empty square braces (ie., []) or using the list() function without any arguments. (ie., list())

We can have elements belonging to different data types in a list.

List Length

We use the **len()** function to find out the length of a list.

The below example was discussed at the timestamp 2:45

```
: lst = ['one', 'two', 'three', 'four']

#find Length of a List
print(len(lst))

4
```

List Insert

Below example was discussed at the timestamp 3:38 in the video.

```
: #syntax: lst.insert(x, y)

lst = ['one', 'two', 'four']

lst.insert(2, "three") # will add element y at location x

print(lst)

['one', 'two', 'three', 'four']
```

We use the **insert()** function with 2 parameters to insert elements into the list. The first argument would be the index position at which the element/value has to be inserted and the second argument would be the value which has to be inserted.

List Remove

Below example was discussed at the timestamp 4:56 in the video.

```
#syntax: lst.remove(x)

lst = ['one', 'two', 'three', 'four', 'two']

lst.remove('two') #it will remove first occurrence of 'two' in a given list

print(lst)

['one', 'three', 'four', 'two']
```

- We use the **remove()** method to remove an element from the given list.
- In case, if the specified element is present multiple times in the given list, then it only removes the first occurrence of the given element.

- In case, if the specified element is not at all present in the given list, then it throws an error.

List append and Extend

Below examples are discussed at the timestamp 3:13 and 6:00 respectively in the video.

```
lst = ['one', 'two', 'three', 'four']

lst.append('five') # append will add the item at the end

print(lst)

['one', 'two', 'three', 'four', 'five']
```

```
lst = ['one', 'two', 'three', 'four']

lst2 = ['five', 'six']

#append
lst.append(lst2)

print(lst)

['one', 'two', 'three', 'four', ['five', 'six']]
```

```
lst = ['one', 'two', 'three', 'four']

lst2 = ['five', 'six']

#extend will join the list with list1

lst.extend(lst2)

print(lst)

['one', 'two', 'three', 'four', 'five', 'six']
```

- Whenever we want to append only one element, then we can use either **append()** or **extend()**. Both work the same way.
- But if we want to append one list with another list, then if we use **append()**, then the second list as it is in the form of a list gets appended to the first list. Whereas if we use **extend()**, all the elements in the second list would be appended one after the other to the first list.

List Delete

Below examples are discussed at the timestamp 7:47 in the video.

```
#del to remove item based on index position

lst = ['one', 'two', 'three', 'four', 'five']

del lst[1]
print(lst)

#or we can use pop() method
a = lst.pop(1)
print(a)

print(lst)
```

['one', 'three', 'four', 'five']
three
['one', 'four', 'five']

```
lst = ['one', 'two', 'three', 'four']

#remove an item from List
lst.remove('three')

print(lst)
```

['one', 'two', 'four']

We can remove any element using the **pop()** method. Whenever we use the **pop()** method, then we have to pass the index whose value has to be removed.

We can also remove the elements using the **del** keyword. An example would be **del a[2]** removes the index present at 2nd index position in the list 'a'.

List related keywords in Python

Below is an example discussed at the timestamp 10:00.

```
#keyword 'in' is used to test if an item is in a list
lst = ['one', 'two', 'three', 'four']

if 'two' in lst:
    print('AI')

#keyword 'not' can combined with 'in'
if 'six' not in lst:
    print('ML')
```

AI
ML

The '**in**' and '**not in**' operators are the membership operators used to check whether the given elements are members of the given list.

List Reverse

The below example begins at the timestamp 11:15 in the video.

```
|: #reverse is reverses the entire list
lst = ['one', 'two', 'three', 'four']
lst.reverse()
print(lst)
['four', 'three', 'two', 'one']
```

We use the **reverse()** method to reverse the given list. This operation is performed on the input itself. It doesn't return a copy.

List Sorting

- The easiest way to sort a given list is using the **sorted()** function. The other way is to sort using the **sort()** method.
- The main difference between both of them is that **sorted()** function returns a copy of the sorted form of a list (It doesn't sort the given list) whereas the **sort()** method performs the sort operation on the given list itself. It doesn't return any copy.
- When we use the **sort()** function the original list is changed whereas when we use the **sorted()** function, the original list doesn't change.
- Whenever we want to sort a list, we need to make sure all the elements in the list belong to the same data type.

- If we want to sort the list in descending order either using sorted() function or sort() method, then we have to use the **reverse = True** argument.

The below examples are discussed starting from the timestamp 11:25.

```
#create a list with numbers
numbers = [3, 1, 6, 2, 8]

sorted_lst = sorted(numbers)

print("Sorted list :", sorted_lst)

#original list remain unchanged
print("Original list: ", numbers)
```

Sorted list : [1, 2, 3, 6, 8]
Original list: [3, 1, 6, 2, 8]

```
#print a list in reverse sorted order
print("Reverse sorted list :", sorted(numbers, reverse=True))

#original list remain unchanged
print("Original list :", numbers)
```

Reverse sorted list : [8, 6, 3, 2, 1]
Original list : [3, 1, 6, 2, 8]

```
: lst = [1, 20, 5, 5, 4.2]

#sort the list and stored in itself
lst.sort()

# add element 'a' to the list to show an error

print("Sorted list: ", lst)
```

Sorted list: [1, 4.2, 5, 5, 20]

```
: lst = [1, 20, 'b', 5, 'a']
print(lst.sort()) # sort list with element of different datatypes.
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-18-98d08ff0e3ba> in <module>()
      1 lst = [1, 20, 'b', 5, 'a']
----> 2 print(lst.sort())

TypeError: '<' not supported between instances of 'str' and 'int'
```

List having multiple references

The below example was discussed at the timestamp 15:10

```
lst = [1, 2, 3, 4, 5]
abc = lst
abc.append(6)

#print original list
print("Original list: ", lst)

Original list: [1, 2, 3, 4, 5, 6]
```

We can use multiple references for the same object. All these multiple references, point to the same object. If we make any changes using any of these references, the original object gets affected.

String Split to create a list

The below example was discussed at the timestamp 16:40 in the video.

```
#Let's take a string

s = "one,two,three,four,five"
slst = s.split(',')
print(slst)

['one', 'two', 'three', 'four', 'five']

s = "This is applied AI Course"
split_lst = s.split() # default split is white-character: space or tab
print(split_lst)

['This', 'is', 'applied', 'AI', 'Course']
```

We use the `split()` method to split the given string into a list of strings. If we want to split the given input string on the basis of any alphabet or number or a string pattern or any special character, then we have to enclose it within a single quote and then pass it as an argument in the `split()` method.

- For example, let us assume the input string is
s1 = 'Welcome to the 1st session of Machine Learning and Deep Learning'
- In this example, if we want to split this string with the number '1' as the separator, then the syntax for it would be `s1.split('1')`.
- If we want to split this string with the substring 'of' as the separator, then the syntax for it would be `s1.split('of')`.

- You also can use special characters like comma(,), colon(:), semi-colon(';'), etc as the separators, but these separators should be present in the given input string.
- If we do not specify the separator, then the whitespace would be considered as the default separator.

List Indexing

- We can access the elements of a string using the indexes. The indexes start from **0** and the last value would be (**length of the string-1**).
- Python also supports negative indexing which means traversing in the reverse direction.

Below is an example discussed at the timestamp 18:15

```
lst = [1, 2, 3, 4]
print(lst[1]) #print second element

#print last element using negative index
print(lst[-2])
```

```
2
3
```

List Slicing

- Accessing parts of segments is called slicing.
- The key point to remember is that the end value represents the first value that is not in the selected slice.

Below example is discussed at the timestamp 19:50

```
: numbers = [10, 20, 30, 40, 50, 60, 70, 80]

#print all numbers
print(numbers[:])
```

```
#print from index 0 to index 3
print(numbers[0:4])
```

```
[10, 20, 30, 40, 50, 60, 70, 80]
[10, 20, 30, 40]
```

```
: print (numbers)
#print alternate elements in a list
print(numbers[::2])
```

```
#print elements start from 0 through rest of the list
print(numbers[2::2])
```

```
[10, 20, 30, 40, 50, 60, 70, 80]
[10, 30, 50, 70]
[30, 50, 70]
```

List extend using the '+' operator

The other way to perform the **extend()** operator is using the '+' operator. Using the '+' operator, we can concatenate two lists.
The below example was discussed at the timestamp 23:25.

```
lst1 = [1, 2, 3, 4]
lst2 = ['varma', 'naveen', 'murali', 'brahma']
new_lst = lst1 + lst2

print(new_lst)
```

```
[1, 2, 3, 4, 'varma', 'naveen', 'murali', 'brahma']
```

List Count

We use the **count()** method to find out the number of occurrences of a given element in the given list.

The below example was discussed at the timestamp 24:10

```
numbers = [1, 2, 3, 1, 3, 4, 2, 5]

#frequency of 1 in a list
print(numbers.count(1))

#frequency of 3 in a list
print(numbers.count(3))
```

2
2

List Looping

We can loop through the entire list by considering each element one by one using a 'for' loop. The below example was discussed at the timestamp 25:25

```
#Loop through a list

lst = ['one', 'two', 'three', 'four']

for ele in lst:
    print(ele)
```

one
two
three
four

List Comprehensions

- List comprehensions provide a concise way to create lists.
- Common applications are to make new lists where each element is the result of some operations applied to each member of another sequence or iterable, or to create a subsequence of those elements that satisfy a certain condition.

The below example was discussed at the timestamp 26:40

```

: # without list comprehension
squares = []
for i in range(10):
    squares.append(i**2)    #list append
print(squares)

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

: #using list comprehension
squares = [i**2 for i in range(10)]
print(squares)

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

: #example
lst = [-10, -20, 10, 20, 50]

#create a new list with values doubled
new_lst = [i*2 for i in lst]
print(new_lst)

#filter the list to exclude negative numbers
new_lst = [i for i in lst if i >= 0]
print(new_lst)

#create a list of tuples like (number, square_of_number)
new_lst = [(i, i**2) for i in range(10)]
print(new_lst)

[-20, -40, 20, 40, 100]
[10, 20, 50]
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64), (9, 81)]

```

Nested List Comprehensions

The below example was discussed at the timestamp 31:40

```
: #Let's suppose we have a matrix

matrix = [
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10, 11, 12]
]

#transpose of a matrix without list comprehension
transposed = []
for i in range(4):
    lst = []
    for row in matrix:
        lst.append(row[i])
    transposed.append(lst)

print(transposed)
```

```
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

```
: #with List comprehension
transposed = [[row[i] for row in matrix] for i in range(4)]
print(transposed)
```

```
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

5.2 Tuples (Part 1)

- Tuple is a data structure that is similar to a List.
- The only difference is that Lists are mutable whereas Tuples are immutable. It means, in a tuple, once after we assign the elements, they cannot be changed whereas in a List we can change whenever needed.

Tuple Creation

The below examples were discussed starting from the timestamp 1:10 in the video.

```
: #empty tuple
t = ()

#tuple having integers
t = (1, 2, 3)
print(t)

#tuple with mixed datatypes
t = (1, 'raju', 28, 'abc')
print(t)

#nested tuple
t = (1, (2, 3, 4), [1, 'raju', 28, 'abc'])
print(t)

(1, 2, 3)
(1, 'raju', 28, 'abc')
(1, (2, 3, 4), [1, 'raju', 28, 'abc'])
```

Similar to a list, even a tuple can contain elements belonging to different data types. Also the tuples support indexing.

```
#only parenthesis is not enough
t = ('satish')
type(t)

str
```

```
#need a comma at the end
t = ('satish',)
type(t)
```

```
tuple
```

We cannot manipulate/modify the elements present in a tuple. But if there are any mutable objects(like list) present as an element in a tuple, we could not completely remove that mutable object, but we can manipulate the elements in that mutable object.

In general, we define a tuple using (). But if there is only 1 element in a tuple, then it has to be followed by a comma(,) in the tuple. Otherwise, the interpreter considers the data type, as the data type of that object, but not as a tuple object.

In the above example, we see

t = ('satisf')

This example, even though we have enclosed the element in (), as there is only one element and it is a string, the interpreter considers 't' as a string object, but not as a tuple. If we want to make the interpreter consider this object as a tuple, then the syntax should be

t = ('satisf',)

Note: Whenever the tuple we define consists of only 1 element, then it always has to be followed by a comma.

Accessing Elements in a Tuple

Tuple also supports indexing and slicing similar to a list.

```

[]: t = ('satish', 'murali', 'naveen', 'srinu', 'brahma')
   print(t[1])
   murali

[]: #negative index
   print(t[-1]) #print last element in a tuple
   brahma

[]: #nested tuple
   t = ('ABC', ('satish', 'naveen', 'srinu'))

   print(t[1])
   ('satish', 'naveen', 'srinu')

[]: print(t[1][2])
   srinu

[]: #Slicing
   t = (1, 2, 3, 4, 5, 6)

   print(t[1:4])

   #print elements from starting to 2nd Last elements
   print(t[:-2])

   #print elements from starting to end
   print(t[:])
   (2, 3, 4)
   (1, 2, 3, 4)
   (1, 2, 3, 4, 5, 6)

```

Changing the elements in a tuple

The below examples are discussed from the timestamp 7:22

```

: #creating tuple
t = (1, 2, 3, 4, [5, 6, 7])

t[2] = 'x' #will get TypeError

-----
TypeError                                                 Traceback (most recent call last)
<ipython-input-3-9f4cbf6ee0de> in <module>()
      2 t = (1, 2, 3, 4, [5, 6, 7])
      3
----> 4 t[2] = 'x' #will get TypeError

TypeError: 'tuple' object does not support item assignment

```

As tuples are immutable objects, one cannot modify the elements present in them. But we can modify the values if an object present in a tuple is a mutable object (like a list). Below is an example of it.

```
|: #creating tuple
|: t = (1, 2, 3, 4, [5, 6, 7])
|: 
|: t[2] = 'x' #will get TypeError
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-3-9f4cbf6ee0de> in <module>()
      2 t = (1, 2, 3, 4, [5, 6, 7])
      3
----> 4 t[2] = 'x' #will get TypeError

TypeError: 'tuple' object does not support item assignment
```

```
|: t[4][1] = 'satish'
|: print(t)
(1, 2, 3, 4, [5, 'satish', 7])
```

Concatenation of Tuples

We can perform concatenation of tuples, but the concatenation returns a copy. We cannot perform concatenation operations in place for a tuple.

```
: #concatinating tuples
:
t = (1, 2, 3) + (4, 5, 6)
print(t)
(1, 2, 3, 4, 5, 6)
```

In the above example, we are concatenating the tuple (4,5,6) to the tuple (1,2,3) and are storing the result in 't'.

If we want to create a tuple with a repetition of a single value, then we have to use the '*' operator.

```
: #repeat the elements in a tuple for a given number of times using the * operator.
:
t = ('satish', ) * 4
print(t)
('satish', 'satish', 'satish', 'satish')
```

In the above example, we want to create a tuple with the string 'satish' occurring 4 times. So we are first creating a tuple with the string 'satish' occurring only once and then multiplying it with the number 4 using the '*' operator.

5.3 Tuples (Part 2)

Deletion of a Tuple

We have to use the **del** keyword to delete any python object. The same keyword we can use to delete a tuple.

The below example is discussed from the timestamp 0:05

```
: #we cannot change the elements in a tuple.  
# That also means we cannot delete or remove items from a tuple.  
  
#delete entire tuple using del keyword  
t = (1, 2, 3, 4, 5, 6)  
  
#delete entire tuple  
del t
```

Tuple Count

The number of occurrences of the given element in a given tuple is obtained using the **count()** function.

The below example was discussed starting from the timestamp 0:40

```
: t = (1, 2, 3, 1, 3, 3, 4, 1)  
  
#get the frequency of particular element appears in a tuple  
t.count(1)  
  
: 3
```

In this example, we are checking how many times the number 1 is occurring in the given tuple. As it is present 3 times, the result is obtained as 3.

Finding the index in a tuple

We use the **index()** method to find out the index of a particular element in a given tuple. If the given element occurs multiple times in a given tuple, then it returns the index of the first occurrence of that element in the given tuple.

The below example was discussed starting from the timestamp 1:05

```
t = (1, 2, 3, 1, 3, 3, 4, 1)

print(t.index(3)) #return index of the first element is equal to 3

#print index of the 1

2
```

In this example, as the number 3 is occurring 3 times, the **index()** method returns the index of the first occurrence of 3.

If the given element is not present in the tuple, then it throws an error.

Tuple Membership

We can check if a given element is present in a tuple, using the ‘in’ keyword.

The below example is discussed starting from the timestamp 1:45

```
#test if an item exists in a tuple or not, using the keyword in.
t = (1, 2, 3, 4, 5, 6)

print(1 in t)

True

print(7 in t)

False
```

In the first example, we are checking if the number 1 is present in the tuple ‘t’. As the value is present, it is returning True.

In the second example, we are checking if the number 7 is present in the tuple ‘t’. As the value is not present, it is returning False.

Built-in Functions

Tuple Length

We use the **len()** function to find out the number of elements present in the given tuple.

```
: t = (1, 2, 3, 4, 5, 6)
print(len(t))

6
```

Tuple Sort

We use the **sorted()** function to return the sorted form of a tuple. The **sorted()** function doesn't perform the sort operation in-place. It returns a copy of the sorted form only in the form of a list.

```
: t = (4, 5, 1, 2, 3)

new_t = sorted(t)
print(new_t) #Take elements in the tuple and return a new sorted list
            #(does not sort the tuple itself).

[1, 2, 3, 4, 5]
```

Maximum element

We use **max()** function to return the maximum value from the given tuple.

```
: #get the largest element in a tuple
t = (2, 5, 1, 6, 9)

print(max(t))

9
```

Minimum element

We use the **min()** function to return the minimum value from the given tuple.

```
#get the smallest element in a tuple
print(min(t))
```

1

5.4 Sets

- A set is an unordered collection of items. Every element is unique (no duplicates).
- A set is a mutable data structure. We can add or remove items from it.
- Sets can be used to perform mathematical set operations like union, intersection, symmetric difference, etc
- The order in which the elements are inserted into a set is sometimes different from the order in which they are displayed.

Set Creation

Below are the examples of different ways in which we can create a set. These examples are discussed in the video starting from the timestamp 0:55.

```
: #set of integers
s = {1, 2, 3}
print(s)

#print type of s
print(type(s))

set([1, 2, 3])
<type 'set'>

: #set doesn't allow duplicates. They store only one instance.
s = {1, 2, 3, 1, 4}
print(s)

{1, 2, 3, 4}

: #we can make set from a list
s = set([1, 2, 3, 1])
print(s)

{1, 2, 3}

: #initialize a set with set() method
s = set()

print(type(s))

<class 'set'>
```

If we want to create an empty set, then the syntax for it would be

s = set()

If we just use the below example, then it creates a dictionary object, but not a set object.

s = {}

As a set doesn't allow duplicate elements, even if we pass duplicate values while creating a set, in the final result, we'll get them only once.

We can use elements of different data types in a set and we can modify the values in a set, as it is a mutable object.

Addition of elements to a set

Below examples are discussed in the video starting from the timestamp 2:45.

```
#we can add single element using add() method and  
#add multiple elements using update() method  
s = {1, 3}  
  
#set object doesn't support indexing  
print(s[1]) #will get TypeError
```

```
-----  
TypeError                                 Traceback (most recent call last)  
<ipython-input-4-c52fc339e293> in <module>()  
      4  
      5 #set object doesn't support indexing  
----> 6 print(s[1]) #will get TypeError  
  
TypeError: 'set' object does not support indexing
```

```
#add element  
s.add(2)  
print(s)  
  
{1, 2, 3}
```

```
#add multiple elements  
s.update([5, 6, 1])  
print(s)  
  
{1, 2, 3, 5, 6}
```

```
#add list and set  
s.update([8, 9], {10, 2, 3})  
print(s)  
  
{1, 2, 3, 5, 6, 8, 9, 10}
```

If we want to add the elements one at a time, then we have to use the **add()** method, whereas if we want to add more than one element at a time, then we have to use the **update()** method.

We can use integers, float values, strings, lists, tuples or combination of these objects as the elements in a set.

Removal of elements from a set

Below are the examples that were discussed in the video starting from the timestamp 5:20.

```
: #A particular item can be removed from set using methods,  
#discard() and remove().  
  
s = {1, 2, 3, 5, 4}  
print(s)  
  
s.discard(4)      #4 is removed from set s  
  
print(s)  
  
{1, 2, 3, 4, 5}  
{1, 2, 3, 5}  
  
:  
: #remove an element  
s.remove(2)  
  
print(s)  
  
{1, 3, 5}  
  
:  
: #remove an element not present in a set s  
s.remove(7) # will get KeyError  
  
-----  
KeyError: 7  
Traceback (most recent call last)  
<ipython-input-14-f37cc9806699> in <module>()  
      1 #remove an element not present in a set s  
----> 2 s.remove(7) # will get KeyError  
  
KeyError: 7
```

```
#discard an element not present in a set s  
s.discard(7)  
print(s)  
  
{1, 3, 5}
```

```
#we can remove item using pop() method  
  
s = {1, 2, 3, 5, 4}  
  
s.pop() #remove random element  
  
print(s)  
  
{2, 3, 4, 5}
```

```
s.pop()  
print(s)  
  
{3, 4, 5}
```

```
s = {1, 5, 2, 3, 6}  
  
s.clear()  #remove all items in set using clear() method  
  
print(s)  
  
set()
```

If we want to remove any random element from the set, then we have to use the **pop()** method. Generally **pop()** throws an error only when the given set is empty. If we want to remove any particular element from the set, then we have to use either **remove()** or **discard()** methods.

The main difference between **remove()** and **discard()** is that if any element that has to be removed from the set is not present, then the **remove()** method throws an error whereas the **discard()** method doesn't throw any error.

If we want to remove all the elements at a time from the set, then we have to use the **clear()** method.

Python Set Operations

We majorly perform operations like union, intersection, difference and symmetric difference between the sets.

Below are the examples discussed starting from the timestamp 7:50.

```
: set1 = {1, 2, 3, 4, 5}
set2 = {3, 4, 5, 6, 7}

#union of 2 sets using | operator
print(set1 | set2)

{1, 2, 3, 4, 5, 6, 7}

: #another way of getting union of 2 sets
print(set1.union(set2))

{1, 2, 3, 4, 5, 6, 7}

: #intersection of 2 sets using & operator
print(set1 & set2)

{3, 4, 5}

: #use intersection function
print(set1.intersection(set2))

{3, 4, 5}

: #set Difference: set of elements that are only in set1 but not in set2
print(set1 - set2)

{1, 2}

: #use difference function
print(set1.difference(set2))

{1, 2}
```

```

: """symmetric difference: set of elements in both set1 and set2
#except those that are common in both."""

#use ^ operator

print(set1^set2)

{1, 2, 6, 7}

: #use symmetric_difference function
print(set1.symmetric_difference(set2))

{1, 2, 6, 7}

: #find issubset()
x = {"a", "b", "c", "d", "e"}
y = {"c", "d"}

print("set 'x' is subset of 'y'?", x.issubset(y)) #check x is subset of y

#check y is subset of x
print("set 'y' is subset of 'x'?", y.issubset(x))

set 'x' is subset of 'y' ? False
set 'y' is subset of 'x' ? True

```

To perform the union operation, you have to use either the ‘|’ operator or the **union()** method.

To perform the intersection operation, you have to use either the ‘&’ operator or the **intersection()** method.

To perform the difference operation, you have to use either the ‘-’ operator or **difference()** method.

To perform the symmetric difference operation, you have to use either the ‘^’ operator or the **symmetric_difference()** method.

To check if a given set is a subset of another set, then you have to use the **issubset()** method.

Frozenset

Frozensets have all the characteristics of set data structure, but the only difference is that once if it is created, then we cannot modify the elements/values in it.

As tuples are the immutable lists, the frozensets are the immutable sets.

Frozensets can be created using the **frozenset()** method.

Since the sets are mutable, they are unhashable and cannot be used as the keys in a dictionary whereas the frozensets being immutable and hashable, can be used as the dictionary keys.

Frozenses supports methods like copy(), difference(), intersection(), isdisjoint(), issubset(), issuperset(), symmetric_difference() and union(). Being immutable it does not have methods that add or remove elements. Also similar to a set, the frozenset also does not support indexing.

Below are the examples that are discussed starting from the timestamp 13.30.

```
set1 = frozenset([1, 2, 3, 4])
set2 = frozenset([3, 4, 5, 6])

#try to add element into set1 gives an error
set1.add(5)

-----
AttributeError                                     Traceback (most recent call last)
<ipython-input-28-8f5ea3d0c7e1> in <module>()
      3
      4 #try to add element into set1 gives an error
----> 5 set1.add(5)

AttributeError: 'frozenset' object has no attribute 'add'
```

```
print(set1[1]) # frozen set doesn't support indexing

-----
TypeError                                      Traceback (most recent call last)
<ipython-input-27-8fc108f08ec8> in <module>()
----> 1 print(set1[1]) # frozen set doesn't support indexing

TypeError: 'frozenset' object does not support indexing
```

```
print(set1 | set2) #union of 2 sets  
frozenset({1, 2, 3, 4, 5, 6})
```

```
#intersection of two sets  
print(set1 & set2)  
  
#or  
print(set1.intersection(set2))
```

```
frozenset({3, 4})  
frozenset({3, 4})
```

```
#symmetric difference  
print(set1 ^ set2)  
  
#or  
print(set1.symmetric_difference(set2))
```

```
frozenset({1, 2, 5, 6})  
frozenset({1, 2, 5, 6})
```

5.5 Dictionary

- Dictionary is an unordered collection of items.
- In the other data structures discussed so far, we had only the values present in the form of elements/items whereas in a dictionary we have the items in the form of key:value pairs
- Similar to the set data structure, as the dictionary is an unordered data structure, we cannot access the elements using indexes.

Dictionary Creation

The below examples are discussed starting from the timestamp 1:25

```
#empty dictionary
my_dict = {}

#dictionary with integer keys
my_dict = {1: 'abc', 2: 'xyz'}
print(my_dict)

#dictionary with mixed keys
my_dict = {'name': 'satish', 1: ['abc', 'xyz']}
print(my_dict)

#create empty dictionary using dict()
my_dict = dict()

my_dict = dict([(1, 'abc'), (2, 'xyz')])      #create a dict with list of tuples
print(my_dict)

{1: 'abc', 2: 'xyz'}
{'name': 'satish', 1: ['abc', 'xyz']}
{1: 'abc', 2: 'xyz'}
```

- We can create an empty dictionary either using empty braces (ie., {}) or using the dict() function without passing any arguments.
- The items in a dictionary are stored in the form of key:value pairs. The keys of a dictionary should be hashable and no duplicate keys are allowed in a dictionary.
- We can have each key belonging to a different data type and this is supported by a dictionary.
- We can either separately define the key-value pairs inside a dictionary with the keys and values separated by a colon(:), or we can pass the key-value pairs either in the form of a tuple or a list.

- We also can use tuples and frozensets as the keys in a dictionary as they are hashable whereas lists and sets could not be used as the keys in a dictionary as they are not hashable.

Dictionary Access

Below examples are discussed starting from the timestamp 7:40 in the video.

```
my_dict = {'name': 'satish', 'age': 27, 'address': 'guntur}

#get name
print(my_dict['name'])

satish

#if key is not present it gives KeyError
print(my_dict['degree'])

-----
KeyError                                  Traceback (most recent call last)
<ipython-input-5-c5aba24e1656> in <module>()
      1 #if key is not present it gives KeyError
----> 2 print(my_dict['degree'])

KeyError: 'degree'

#another way of accessing key
print(my_dict.get('address'))

guntur

#if key is not present it will give None using get method
print(my_dict.get('degree'))
```

None

We can access the values from a dictionary with the help of their corresponding keys. The syntax for it would be **<dictionary-name>[<key-name>]**.

The same value can be accessed using the **get()** method of the dictionary class where we have to pass the **key** as an argument in the **get()** method.

If the specified key is not present in the dictionary, then it throws **KeyError**.

Add or Modify the dictionary items

The below examples are discussed starting from the timestamp 9:50 in the video.

```
my_dict = {'name': 'satish', 'age': 27, 'address': 'guntur'}  
  
#update name  
my_dict['name'] = 'raju'  
  
print(my_dict)  
  
{'name': 'raju', 'age': 27, 'address': 'guntur'}  
  
  
#add new key  
my_dict['degree'] = 'M.Tech'  
  
print(my_dict)  
  
{'name': 'raju', 'age': 27, 'address': 'guntur', 'degree': 'M.Tech'}
```

In the above example, if we want to modify the value of a key present in the dictionary, then it is as simple as assigning a value to the key.

my_dict['name'] = 'raju'

In this example, it looks as if we are assigning the value 'raju' to the key 'name' in the dictionary.

In case, if we do not have the specified key in the dictionary, then it creates the key-value pair in the dictionary.

Deletion/Removal of elements from a dictionary

The below examples are discussed starting from the timestamp 11:20 in the video.

```
: #create a dictionary
my_dict = {'name': 'satish', 'age': 27, 'address': 'guntur'}

#remove a particular item
print(my_dict.pop('age'))

print(my_dict)
```

```
27
{'name': 'satish', 'address': 'guntur'}
```

```
: my_dict = {'name': 'satish', 'age': 27, 'address': 'guntur'}

#remove an arbitrary key
my_dict.popitem()

print(my_dict)
```

```
{'name': 'satish', 'age': 27}
```

```
: squares = {2: 4, 3: 9, 4: 16, 5: 25}

#delete particular key
del squares[2]

print(squares)
```

```
{3: 9, 4: 16, 5: 25}
```

```
: #remove all items
squares.clear()

print(squares)
```

```
{}
```

```

squares = {2: 4, 3: 9, 4: 16, 5: 25}

#delete dictionary itself
del squares

print(squares) #NameError because dict is deleted
-----
NameError Traceback (most recent call last)
<ipython-input-16-355e8277492b> in <module>()
      4 del squares
      5
----> 6 print(squares) #NameError because dict is deleted

NameError: name 'squares' is not defined

```

- If we want to remove a key-value pair from the dictionary, then we have to use the **pop()** method and should pass the key as an argument in it. The specified key-value pair gets removed and the value associated with that key would be returned.
- The other way to use the same job is by using the **del** keyword. The syntax for it would be **del <dictionary_name>[<key>]**. This approach doesn't return any value.
- If we want to remove a random key-value pair, then we have to use the **popitem()** method of the dictionary class.
- If we want to remove all the key-value pairs, then we have to use the **clear()** method.
- If we want to delete the whole dictionary then the syntax would be **del <dictionary_name>**

Dictionary Methods

The below examples are discussed starting from the timestamp 13:35

```

squares = {2: 4, 3: 9, 4: 16, 5: 25}

my_dict = squares.copy()
print(my_dict)

{2: 4, 3: 9, 4: 16, 5: 25}

```

- We use the **copy()** method to return another copy of the given dictionary.

```
#fromkeys(seq[, v]) -> Return a new dictionary with keys from seq and value equal to v (defaults to None).
subjects = {}.fromkeys(['Math', 'English', 'Hindi'], 0)
print(subjects)

{'Math': 0, 'English': 0, 'Hindi': 0}
```

- We use the **fromkeys()** method by passing a sequence and a value as the arguments, in order to create a new dictionary with the elements in the sequence as the keys and the specified value, as the value for those keys.

```
subjects = {2:4, 3:9, 4:16, 5:25}
print(subjects.keys()) #return a new view of the dictionary keys

dict_keys([2, 3, 4, 5])
```

- We use the **keys()** method to return all the keys in the form of a list.

```
subjects = {2:4, 3:9, 4:16, 5:25}
print(subjects.values()) #return a new view of the dictionary values

dict_values([4, 9, 16, 25])
```

- We use the **values()** method to return all the values in the form of a list.

```
: subjects = {2:4, 3:9, 4:16, 5:25}
print(subjects.items()) #return a new view of the dictionary items (key, value)

dict_items([(2, 4), (3, 9), (4, 16), (5, 25)])
```

- We use the **items()** method to return all the key-value pairs as a list of tuples.

```
: #get List of all available methods and attributes of dictionary
d = {}
print(dir(d))

['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattr__',
 '__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__ne__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__',
 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
```

- We use the **dir()** method to display a list of all the available methods and attributes of a Dictionary.

Dictionary Comprehension

5.6 Strings

- A string is a sequence of characters.
- Computers do not deal with the characters. They only deal with the binary numbers. Even though we may see the characters on the screen, internally they are stored and are manipulated as 0's and 1's.
- The conversion of character to a number is called encoding and the reverse process is called decoding. ASCII and Unicode are the most popularly used encoding techniques.
- In Python, a string is a sequence of unicode characters.

How to create a string?

- Strings can be created by enclosing the characters either inside single quotes or double quotes or triple quotes.
- Mostly we use triple quotes only when we want to represent the multi-line strings or the docstrings.

Below example is discussed at the timestamp 0:40 in the video.

```
myString = 'Hello'  
print(myString)  
  
myString = "Hello"  
print(myString)  
  
myString = """Hello"""  
print(myString)
```

```
Hello  
Hello  
Hello
```

How to access the characters in a string?

- We can access the individual characters using the indexing and the range of characters using slicing.
- Indexing starts from **0** and ends with the **(total length of the string-1)**. Whenever we try to access any element with an index outside the above specified range, then it throws an error.
- We have to use only integers as the indexes to access the characters of a string. We cannot use float values or any other characters as the indexes.

- Python also allows negative indexing, but the traversal happens in the reverse order.

Below examples are discussed in the video starting from the timestamp 1:35 and 2:55 respectively.

```
: myString = "Hello"

#print first Character
print(myString[0])

#print last character using negative indexing
print(myString[-1])

#slicing 2nd to 5th character
print(myString[2:5])
```

```
H
o
llo
```

```
print(myString[15])
```

```
-----  
IndexError                                     Traceback (most recent call last)
<ipython-input-2-78353fed94bc> in <module>()
----> 1 print(myString[15])

IndexError: string index out of range
```

```
print(myString[1.5])
```

```
-----  
TypeError                                     Traceback (most recent call last)
<ipython-input-3-d32f87fd1591> in <module>()
----> 1 print(myString[1.5])

TypeError: string indices must be integers
```

How to change or delete a string?

- Strings are immutable objects. It means that once after a string is assigned with some values, it cannot be changed.
- Instead we can make reassignments to the same string variable but with a different value. Making changes to the existing values is not possible

Below example starts from the timestamp 3:50.

```
myString = "Hello"
myString[4] = 's' # strings are immutable

-----
TypeError                                 Traceback (most recent call last)
<ipython-input-6-d63a28c2a378> in <module>()
      1 myString = "Hello"
----> 2 myString[4] = 's' # strings are immutable

TypeError: 'str' object does not support item assignment
```

We cannot delete or remove characters from a string. But deleting the string entirely is possible using the keyword **del**.

- We cannot modify elements/substrings of a given string, as strings are immutable objects.
- In order to delete a string, we use the **del** keyword followed by the string name.

The below example follows the above example in the video.

```
del myString # delete complete string

print(myString)

-----
NameError                                 Traceback (most recent call last)
<ipython-input-8-60c083ddb862> in <module>()
----> 1 print(myString)

NameError: name 'myString' is not defined
```

String Operations

Concatenation

- Concatenation is the process of combining two or more strings into a single string.
- In order to combine the strings, we use the '+' operator between two string literals.
- In order to repeat the given string for a specified number of times, then we have to use '*' operator.

The next example was discussed starting from the timestamp 5:30 in the video.

```
s1 = "Hello "
s2 = "Satish"

#concatenation of 2 strings
print(s1 + s2)

#repeat string n times
print(s1 * 3)
```

```
Hello Satish
Hello Hello Hello
```

- As we are using the ‘+’ operator, the strings ‘Hello’ and ‘Satish’ are getting concatenated.
- As we are using the ‘*’ operator followed by the number 3, the string ‘Hello’ is occurring 3 times and all these are getting concatenated.

Iteration through String

As a string is a sequence of characters, we can iterate over the sequence each character-wise. The below example begins at the timestamp 6:20 in the video.

```
count = 0
for l in "Hello World":
    if l == 'o':
        count += 1
print(count, ' letters found')

2 letters found
```

In the above example, we are iterating over the string “Hello World” and in each iteration, the variable ‘l’ denotes each character from the string “Hello World”.

String Membership Test

We can check if a character/substring is a member/part of the given string. We use the ‘in’ operator to check for this purpose. The below example was discussed at the timestamp 7:20 in the video.

```
: print('l' in 'Hello World') #in operator to test membership  
True
```

```
: print('or' in 'Hello World')  
True
```

In the first example, we are checking if the character ‘l’ is present in the string “Hello World”. As the character ‘l’ is present in the given string, it returns True.

In the second example, we are checking if the substring ‘or’ is present in the string “Hello World”. As the substring ‘or’ is present in the given string, it returns True.

String Methods

The below examples are discussed starting from the timestamp 7:50 in the video.

```
"Hello".lower()
```

```
'hello'
```

```
"Hello".upper()
```

```
'HELLO'
```

```
"This will split all words in a list".split()
```

```
['This', 'will', 'split', 'all', 'words', 'in', 'a', 'list']
```

```
' '.join(['This', 'will', 'split', 'all', 'words', 'in', 'a', 'list'])
```

```
'This will split all words in a list'
```

```
"Good Morning".find("Mo")
```

```
5
```

```
s1 = "Bad morning"
```

```
s2 = s1.replace("Bad", "Good")
```

```
print(s1)  
print(s2)
```

```
Bad morning  
Good morning
```

- We use the **lower()** function to convert the given string into lowercase.
- We use the **upper()** function to convert the given string into uppercase.
- We use the **split()** function to split the given string into a list of strings.
- We use the **join()** function to combine all the string elements in a given list into a single string format.
- We use the **find()** function to find out the index of a given character or substring in the given string.
- We can use the **replace()** function with 2 arguments, to replace an old substring with a new substring. But as strings are immutable objects, the **replace()** function doesn't perform any replacements on the original strings. It just returns a copy of the modified string.

Python Program to check if the given string is a palindrome

The below code snippet was explained in the video at the timestamp 12:20 in the video.

```

myStr = "Madam"

#convert entire string to either Lower or upper
myStr = myStr.lower()

#reverse string
revStr = reversed(myStr)

#check if the string is equal to its reverse
if list(myStr) == list(revStr):
    print("Given String is palindrome")
else:
    print("Given String is not palindrome")

```

Given String is palindrome

- In the above example, we are first converting the given string into lower case (it is because the lower case and the uppercase of an alphabet are considered as two different alphabets)
- After converting the string into lowercase, we are reversing the string and then converting both the strings(the original one and the reverse form) into lists of characters(same in the order in which they appear in the strings) and are comparing each element corresponding to the same indexes.
- If all the characters are the same, then it returns True and we can declare the given string as a Palindrome.

Python Program to sort the words in the given string

Below example was discussed at the timestamp 14:00 in the video.

```
myStr = "python Program to Sort words in Alphabetic Order"

#breakdown the string into list of words
words = myStr.split()

#sort the list
words.sort()

#print Sorted words are
for word in words:
    print(word)

Alphabetic
Order
Program
Sort
in
python
to
words
```

- In the above example, we first have to split the given input string into a list of substrings.
- We then have to sort that list using the sort() function and then traverse through the sorted list and print the elements one by one. This sorting is done on the basis of alphabetical order.

6.1 Introduction

Python functions

- A Function is a group of related statements that perform a specific task.
- Functions help us to break the long code into smaller chunks.
- As our programs grow larger and larger, the functions make the code look more organized and manageable.
- It avoids repetition of the code and makes the block of the code reusable.

Syntax

```
def function_name(parameters):
```

.....

Doc String

.....

Statement(s)

- The keyword ‘**def**’ marks the starting of the function header.
- Parameters (the arguments) are the values which we pass to the function.
- The colon(:) marks the end of the function header.
- Docstring is used for describing what the function does. This is optional.
- The **return** statement is used to return a value from the function.

The below example was discussed at the timestamp 1:35 in the video.

Example:

```
def print_name(name):
    """
    This function prints the name
    """
    print("Hello " + str(name))
```

Function Call

Once we have defined a function, we can call it from anywhere

```
print_name('satish')
```

```
Hello satish
```

We have defined the **print_name()** function and called it with the parameter 'satish'. So this parameter is used in the task performed by the function **print_name()** and the result is returned in the same format as that specified in the **return** statement.

Once after a function is defined, we can call it from anywhere and any number of times.

Doc String

- The first string after the function header is called the docstring. It is the documentation string and tells us in a short description what all the function is about.
- Whenever we define any function, it is a good coding practice to mention the docstring.
- Docstring is usually written in triple quotes as it could extend up to multiple lines.

The syntax to print the doc string of the above function **print_name()** is

```
print(print_name.__doc__)
```

o/p:

This function prints the name

return statement

- The **return** statement is used to exit a function and go back to the place from where it was called.

Syntax:

return expression

- Either we can directly return a value or a variable or an expression through the **return** statement.
- If there is no expression passed through the **return** statement or if there is no return statement in a function, then the function returns **None** object.

The below example was discussed at the timestamp 4:30

```

: def get_sum(lst):
    """
        This function returns the sum of all the elements in a list
    """
    #initialize sum
    _sum = 0

    #iterating over the list
    for num in lst:
        _sum += num
    return _sum

: s = get_sum([1, 2, 3, 4])
print(s)

10

: #print doc string
print(get_sum.__doc__)

```

This function returns the sum of all the elements in a list

In the above example, we are passing a list of elements as an input to the **get_sum()** function and the function returns the sum of all the elements in the list.

How Functions Work in Python?

Scope and Lifetime of Variables

- Scope of a variable is the portion of the program upto which a variable gets recognized.
- Variables defined inside a function are not seen from the outside. Hence such variables have local scope.
- Lifetime of a variable is the period through which the variable lies in the memory.

- The lifetime of variables inside a function is as long as the function executes.
- Variables are destroyed once we return from the function.

The below example was discussed at the timestamp

```
global_var = "This is global variable"

def test_life_time():
    """
    This function test the life time of a variables
    """
    local_var = "This is local variable"
    print(local_var)      #print local variable local_var

    print(global_var)     #print global variable global_var

#calling function
test_life_time()

#print global variable global_var
print(global_var)

#print Local variable local_var
print(local_var)
```

```
This is local variable
This is global variable
This is global variable
```

```
NameError                                 Traceback (most recent call last)
<ipython-input-12-d5226680661e> in <module>()
      19
      20 #print local variable local_var
--> 21 print(local_var)

NameError: name 'local_var' is not defined
```

In the above example, 'global_var' is a global variable and it can be accessed either inside a function or outside a function. Whereas 'local_var' is a local variable defined inside a function, so we could access it only inside the function. Once the control comes out of the function, this variable becomes invalid.

Program to compute the HCF of two numbers

The below example was discussed at the timestamp 10:30 in the video.

```
def computeHCF(a, b):
    """
    Computing HCF of two numbers
    """
    smaller = b if a > b else a # concise way of writing if else statement

    hcf = 1
    for i in range(1, smaller+1):
        if (a % i == 0) and (b % i == 0):
            hcf = i
    return hcf

num1 = 6
num2 = 36

print("H.C.F of {0} and {1} is: {2}".format(num1, num2, computeHCF(num1, num2)))
```

H.C.F of 6 and 36 is: 6

6.2 Types of Functions

There are two types of functions. They are

- 1) Built-in functions
- 2) User defined functions

Built-in Functions

1) `abs()`

The `abs()` function is used to find out the absolute value of a given number.

The below example was discussed at the timestamp 0:38 in the video.

```
# find the absolute value
num = -100
print(abs(num))
```

100

2) `all()`

The `all()` function takes an iterable(either a list or a tuple or a set) as an input and returns **True** if all the elements in the iterable are **True**. If any element in the iterable is **False**, then it returns **False**.

Note: The values **0** and **None** are also considered as **False**. Here if the given iterable contains any element as **0** or **False** or **None**, then the `all()` function returns the result as False. Otherwise it returns **True**.

The below examples were discussed at the timestamp 1:40.

```
lst = [1, 2, 3, 4]
print(all(lst))
```

True

```
lst = (0, 2, 3, 4)      # 0 present in list
print(all(lst))
```

False

```
lst = []                  #empty list always true
print(all(lst))
```

True

```
lst = [False, 1, 2]      #False present in a list so all(lst) is False
print(all(lst))
```

False

3) dir()

- The dir() returns a list of the valid attributes of the object.
- If the object supports the dir() method, whenever the method is called with this object as an argument, then it must return a list of all the attributes of that object.
- If the object doesn't support the dir() function, whenever the method is called with this object as an argument, then this method tries to find information from the **dict** attribute (if defined), and from the type object. In this case, the list returned from dir() may not be complete.

The below example was discussed at the timestamp 4:20 in the video.

```
: numbers = [1, 2, 3]
print(dir(numbers))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
 '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__',
 '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rm__':
 '__setattribute__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count',
 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

4) divmod()

The divmod() function takes two values(say **a** and **b**) as an input and returns a pair of values in the form of a tuple. The first value in the tuple would be the **quotient** obtained when **a** is divided by **b** and the second value would be the **remainder** obtained when **a** is divided by **b**.

The below example was discussed at the timestamp 5:23.

```
print(divmod(9, 2)) #print quotient and remainder as a tuple  
#try with other number
```

(4, 1)

5) enumerate()

The enumerate() adds a counter to an iterable and returns it. So whenever we loop through the iterable (with counter added), we get each item in the form of a tuple with a pair of values. The first value would be the index and the second value would be the corresponding element in the iterable.

The below example was discussed at the timestamp 6:08

```
: numbers = [10, 20, 30, 40]  
  
for index, num in enumerate(numbers,10):  
    print("index {0} has value {1}".format(index, num))  
  
index 10 has value 10  
index 11 has value 20  
index 12 has value 30  
index 13 has value 40
```

The first argument in enumerate() should be the iterable to which we want to add the counter. The second argument should be the starting index of the counter. If we want to manually assign the starting index, then we have to pass the required starting value. Otherwise, the default starting index would be 0.

6) filter()

The filter() method constructs an iterator from elements of an iterable for which a function returns True.

The below example was discussed at the timestamp 9:10

```
def find_positive_number(num):
    """
        This function returns the positive number if num is positive
    """
    if num > 0:
        return num
```

```
number_list = range(-10, 10) #create a list with numbers from -10 to 10
print(list(number_list))

positive_num_lst = list(filter(find_positive_number, number_list))

print(positive_num_lst)

[-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In the above example, we are defining the function **find_positive_number()** which returns the input number if it is greater than 0. Otherwise, it returns None.

So now the filter() function will loop through the input iterator and passes each element of the iterator through the function **find_positive_number()** and creates an iterator with only those elements that do not return **None** when passed through the function **find_positive_number()**.

We are passing a range of values in the interval [-10,9] as an input to the filter() and only those values that are greater than 0 are returned.

7) **isinstance()**

The **isinstance()** function checks if the **object** (first argument) is an instance or subclass of **class_name** class (second argument).

Syntax

isinstance(object, class_name)

The below example was explained at the timestamp 12:45.

```
: lst = [1, 2, 3, 4]
print(isinstance(lst, list))

#try with other datatypes tuple, set
t = (1,2,3,4)
print(isinstance(t, list))

True
False
```

In the first example, the **isinstance()** is checking if the object ‘lst’ is an instance of ‘list’ class. If yes, then it returns **True**. Otherwise, it returns **False**. As ‘lst’ is a list, the **isinstance()** is returning **True** in this case.

In the second example, the **isinstance()** is checking if the object ‘t’ is an instance of ‘list’ class. If yes, then it returns **True**. Otherwise, it returns **False**. As ‘t’ is a tuple, but not a list, the **isinstance()** is returning **False** In this case.

8) **map()**

Syntax:

map(function_name, iterable_name)

The **map()** will apply the function ‘function_name’ to each and every element in the iterable ‘iterabl_name’. The result would be of ‘map’ class. If we want to get the result in the form of a list (or) a tuple (or) a set, then we have to apply the **list()** (or) **tuple()** (or) **set()** functions on the **map()** function output.

The below example was discussed at the timestamp 13:50.

```
: numbers = [1, 2, 3, 4]

#normal method of computing num^2 for each element in the list.
squared = []
for num in numbers:
    squared.append(num ** 2)

print(squared)
```

```
[1, 4, 9, 16]
```

```
: numbers = [1, 2, 3, 4]

def powerOfTwo(num):
    return num ** 2

#using map() function
squared = list(map(powerOfTwo, numbers))
print(squared)
```

```
[1, 4, 9, 16]
```

In the above example, we want to compute the square of each element in the given input list. One way is to iterate through the whole list and perform power of 2 operation on each element.

The other way is to define a function **powerOfTwo()** which performs square on the given element. We use this function as the first argument and the list ‘numbers’ as the second argument in the **map()** function.

9) **reduce()**

- **reduce()** function is for performing some computation on a list and returning the result.
- It applies a rolling computation to sequential pairs of values in a list.

The below example was discussed at the timestamp 17:10

```
#product of elements in a list
product = 1
lst = [1, 2, 3, 4]

# traditional program without reduce()
for num in lst:
    product *= num
print(product)
```

24

```
#with reduce()
from functools import reduce # in Python 3.

def multiply(x,y):
    return x*y;

product = reduce(multiply, lst)
print(product)
```

24

In the above example, in the first code snippet, we are performing the product of all the elements in the list.

In the second code snippet, we are performing the same task using the `reduce()` function.

User Defined Functions

- Functions that we define ourselves to do certain specific task are referred as user-defined functions
- If we use functions written by others in the form of a library, it can be termed as library functions.

Advantages of User Defined Functions

- User-defined functions help to decompose a large program into small segments which makes the program easy to understand, maintain and debug.
- If repeated code occurs in a program. Function can be used to include those codes and execute when needed by calling that function.
- Programmers working on large projects can divide the workload by making different functions.

The below example was discussed at the timestamp 21:34

```
def product_numbers(a, b):
    """
    This function returns the product of two numbers
    """
    product = a * b
    return product

num1 = 10
num2 = 20
print "product of {0} and {1} is {2} ".format(num1, num2, product_numbers(num1, num2))

product of 10 and 20 is 200
```

In the above example, **product_numbers()** is a user defined function to perform the product operation between two numbers.

Python program to make a simple calculator that can add, subtract, multiply and division

```
: def add(a, b):
    """
    This function adds two numbers
    """
    return a + b

def multiply(a, b):
    """
    This function multiply two numbers
    """
    return a * b

def subtract(a, b):
    """
    This function subtract two numbers
    """
    return a - b

def division(a, b):
    """
    This function divides two numbers
    """
    return a / b
```

```
: print("Select Option")
print("1. Addition")
print ("2. Subtraction")
print ("3. Multiplication")
print ("4. Division")

#take input from user
choice = int(input("Enter choice 1/2/3/4"))

num1 = float(input("Enter first number:"))
num2 = float(input("Enter second number:"))
if choice == 1:
    print("Addition of {0} and {1} is {2}".format(num1, num2, add(num1, num2)))
elif choice == 2:
    print("Subtraction of {0} and {1} is {2}".format(num1, num2, subtract(num1, num2)))
elif choice == 3:
    print("Multiplication of {0} and {1} is {2}".format(num1, num2, multiply(num1, num2)))
elif choice == 4:
    print("Division of {0} and {1} is {2}".format(num1, num2, division(num1, num2)))
else:
    print("Invalid Choice")

Select Option
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter choice 1/2/3/43
Enter first number:12.2
Enter second number:2.3
Multiplication of 12.2 and 2.3 is 28.059999999999995
```

In the above program, we are defining the functions to perform addition, subtraction, multiplication and division operations separately for each.

We have to enter a number of our choice(in between 1 to 4 inclusive) and the corresponding operation would be performed and the result gets printed.

6.3 Function Arguments

The below example was discussed at the timestamp 0:20

```
: def greet(name, msg):
    """
    This function greets to person with the provided message
    """
    print("Hello {0} , {1}".format(name, msg))

#call the function with arguments
greet("satish", "Good Morning")
```

```
Hello satish , Good Morning
```

```
: #suppose if we pass one argument
greet("satish") #will get an error
```

```
-----
TypeError                                     Traceback (most recent call last)
<ipython-input-3-b48ea98044bf> in <module>()
      1 #suppose if we pass one argument
      2
----> 3 greet("satish") #will get an error

TypeError: greet() missing 1 required positional argument: 'msg'
```

- In this example, we are defining the function greet() which takes two parameters.
- Here whenever we want to call the function greet(), then we have to pass the values as well. The values passed while calling the function will be assigned to the arguments in the function header in the same order as they are present.
- The number of arguments passed while calling the function should be the same as the number of arguments defined in the function.
- In our example, as we have two parameters defined in the greet() function, we are also passing the two values 'satish' and 'Good Morning'. The value 'satish' gets assigned to the 'name' parameter and the value 'Good Morning' gets assigned to the 'msg' parameter and the function gets executed.
- When we are calling the greet() function for the second time, we are passing only 1 value. The first value 'satish' gets assigned to the first argument 'name' and there would be no value for the 'msg' argument. The method signature now is different from the one that was defined. Hence it throws an error. In such cases, whenever we want certain arguments to

hold the same value for multiple function calls and change only a few of the arguments, we have to go with default arguments.

Different Forms of Arguments

1) Default Arguments

We can provide a default value to an argument using the assignment operator (=).

At the time of the function call, if we again pass any value for that argument, then the newly passed value will be assigned to the argument. Otherwise, the default value that was assigned in the function definition would be assigned to this argument.

Such arguments with default values assigned at the time of defining the function itself are called **Default Arguments**.

Below is an example that was discussed at the timestamp 1:32

```
def greet(name, msg="Good Morning"):
    """
    This function greets to person with the provided message
    if message is not provided, it defaults to "Good Morning"
    """
    print("Hello {0} , {1}".format(name, msg))

greet("satish", "Good Night")
```

```
Hello satish , Good Night
```

```
#with out msg argument
greet("satish")
```

```
Hello satish , Good Morning
```

Once we have a default argument, all the arguments to its right must also have default values.

```
def greet(msg="Good Morning", name):
    #will get a SyntaxError : non-default argument follows default argument
```

In this example, we are assigning the value 'Good Morning' to the 'msg' argument. This is the default value assigned to this argument.

At the time of the first function call, we are passing the values 'satish' and 'Good Night'. So these two values get assigned to the two arguments.

But in the second function call, we are passing only the value 'satish' which gets assigned to the 'name' argument. Now there is no value passed

to the ‘msg’ argument. In such cases, the default value ‘Good Morning’ that was assigned in the function definition, will now be assigned to this ‘msg’ argument.

In this example, ‘msg’ is a default argument and ‘name’ is a non-default argument. At the time of function definition and function call, we must make sure the non default arguments come first and then the default arguments, as it is a good programming practice.

2) Keyword Arguments

kwarg allows you to pass keyworded variable length of arguments to a function. You should use **kwarg if you want to handle named arguments in a function.

Below is an example discussed at the timestamp 4:45

```
def greet(**kwargs):
    """
    This function greets to person with the provided message
    """
    if kwargs:
        print("Hello {0} , {1}".format(kwargs['name'], kwargs['msg']))
greet(name="satish", msg="Good Morning")
```

Hello satish , Good Morning

In the above example, whenever we want to manually assign values to each of the parameters in the key-value format, then we call them Keyword arguments.

We have to assign each value with the help of the argument names as the keys while calling the function. We have to mention only **kwarg as an argument in the function definition.

In the function body, this **kwarg would be considered as a dictionary and each of the arguments in it will behave like the keys. The accessing of the values associated with these keys is the same as how we access the values in a dictionary.

Here ‘name’ and ‘msg’ are the keys and kwarg is considered as a dictionary in the function body. The values associated with these keys will be accessed as kwarg[‘name’] and kwarg[‘msg’].

3) Arbitrary Arguments

Sometimes, we do not know in advance the number of arguments

that will be passed into a function. Python allows us to handle this kind of situation through function calls with arbitrary number of arguments.

The below example was discussed at the timestamp 6:50

```
def greet(*names):
    """
    This function greets all persons in the names tuple
    """
    print(names)

    for name in names:
        print("Hello, {} ".format(name))

greet("satish", "murali", "naveen", "srikanth")
```

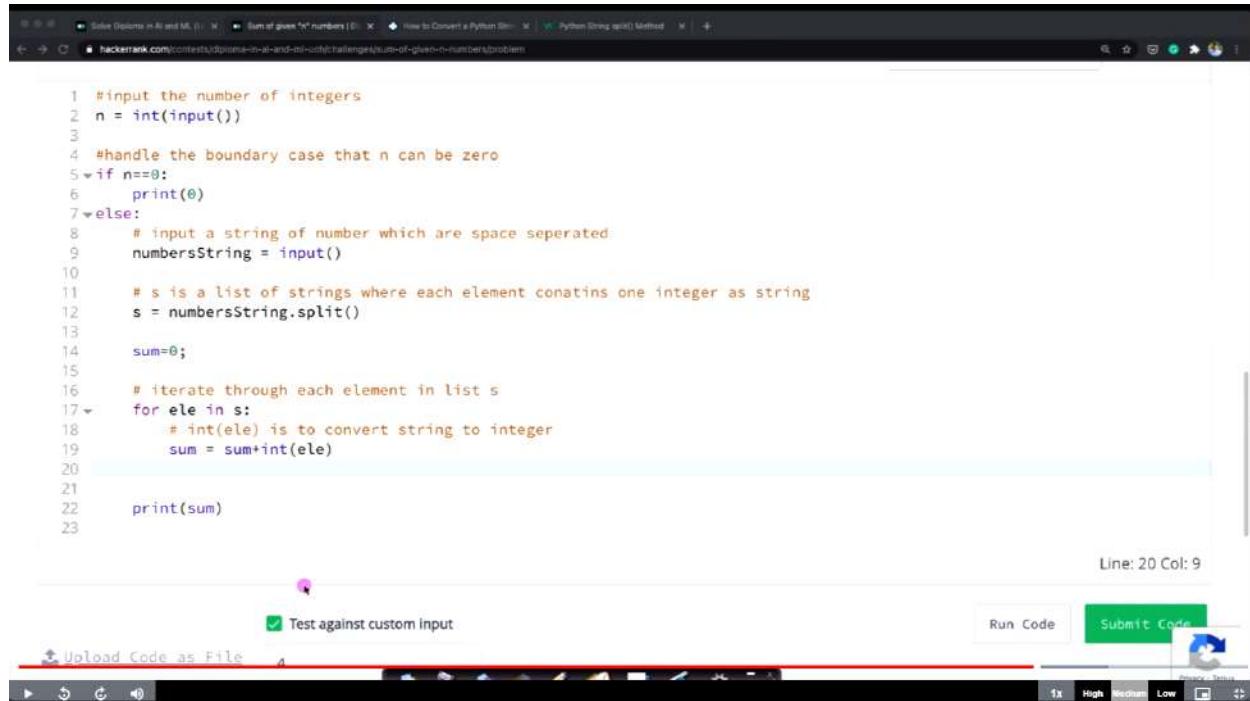


```
('satish', 'murali', 'naveen', 'srikanth')
Hello, satish
Hello, murali
Hello, naveen
Hello, srikanth
```

In this example, we do not have any idea about the number of values we use at runtime. In the function call, we have passed 4 values. In the next function call, we might pass more or less values. In such cases, when we do not have idea about the number of values we pass at the runtime, we go with these arbitrary arguments. It means the argument having an arbitrary length.

7.2 Problem 1: Sum of ‘n’ input numbers

The below example is explained in the video



A screenshot of a code editor window from [hackerank.com](#). The window title is "Sum of given 'n' numbers". The code area contains the following Python script:

```
1 #input the number of integers
2 n = int(input())
3
4 #handle the boundary case that n can be zero
5 if n==0:
6     print(0)
7 else:
8     # input a string of number which are space seperated
9     numbersString = input()
10
11 # s is a list of strings where each element conatins one integer as string
12 s = numbersString.split()
13
14 sum=0;
15
16 # iterate through each element in list s
17 for ele in s:
18     # int(ele) is to convert string to integer
19     sum = sum+int(ele)
20
21
22 print(sum)
```

The status bar at the bottom right shows "Line: 20 Col: 9". Below the code editor are several buttons: "Test against custom input" (with a checked checkbox), "Run Code", "Submit Code", "Upload Code as File", and a zoom level selector set to "1x".

- In this program, the first input we give is the total number of elements in the list.
- In case, if the entered number of elements is 0, then we have to print 0.
- Otherwise, we accept the sequence of values in the form of a string separated by spaces. We initialize a variable ‘sum’ equal to 0.
- We split the string into a list of strings and while traversing through the list, we convert each number in the string format into integer format and then add it to the ‘sum’ variable. Finally we are printing the ‘sum’ value.

7.3 Problem 2: Second largest element in a list

Current Buffer (saved locally, editable)  

Python 3

```
1 # Good idea to write your test cases to build the logic
2 # sample input: 1,2,3,4,6,6 =>1
3 # sample input: 3,2,3 => 2
4 # sample input: 1,1,1 => Error
5 # sample input: 1 => Error
6
7 numberString = input()
8
9 # split the numberString and convert it to a list of numbers
10 l = [ int(i) for i in numberString.split(',') ] # list comprehension
11 n=len(l)
12
13 # boundary case
14 if n<2:
15     print("Error")
16     exit()
17
18 # initialize firstMax and secondMax
19 firstMax=max(l[0],l[1])
20 secondMax=min(l[0],l[1])
21
22
23 # iterate
24 for i in range(2,n):
25     if l[i]>firstMax: # test case: 3,2,4
26         secondMax=firstMax
27         firstMax=l[i]
28     elif l[i]>secondMax and firstMax != l[i]: # test case: 3,2,3
29         secondMax=l[i]
30
31
32 # Boundary case: input: 1,1,1 => there is no second largest element
33 if firstMax==secondMax:
34     print("Error")
35 else:
36     print(secondMax)
```

Line: 12 Col: 1

 Upload Code as File

 Test against custom input

Run Code

Submit Code

7.4 Problem 3: Fizz Buzz

The below example is explained in the video.

A screenshot of a code editor window titled "Fizz Buzz-11 | Problems in Action". The URL in the address bar is "hackerrank.com/problems/prime-in-a-and-mh-unicy/challenges/fizz-buzz-11/problem". The code is written in Python and prints numbers from 1 to n, replacing multiples of 3 with "Fizz", multiples of 5 with "Buzz", and multiples of both with "Fizz Buzz". The code uses a for loop and an if-elif-else structure. The code editor shows line 21 of 21, column 31. Below the code are buttons for "Upload Code as File", "Test against custom input", "Run Code", and "Submit Code". At the bottom, there are navigation icons and a zoom level indicator set to 1x.

```
1 n = int(input())
2
3 separator = "," # separator between each printed value
4
5 for i in range(1,n+1):
6
7     # at the end, print newline and not a comma
8     if i==n:
9         separator = "\n"
10
11    if i%15 == 0:
12        print("FizzBuzz",end=separator)
13        continue
14    elif i%3 == 0:
15        print("Fizz",end=separator)
16        continue
17    elif i%5 == 0:
18        print("Buzz",end=separator)
19        continue
20    else:
21        print(i,end=separator)
```

In this example, we are first taking the input number 'n' from the user at the runtime and are initializing the 'separator' variable to ','.

The condition here is for all the numbers in the range 1 to n(inclusive), if a number is divisible by 3, then we have to print the word '**Fizz**'. If it is divisible by 5, then we have to print the word '**Buzz**'. If it is divisible by both 3 and 5, then we have to print the string '**Fizz Buzz**'. If the number is neither divisible by 3 nor divisible by 5, then we have to print the number as it is.

Each value we print should be separated by a comma(,) and hence we have initialized the separator as ','. While looping through the sequence, we have to change the 'separator' value to '\n', once if it becomes equal to '\n'.

8.1 Sorting & Searching : why bother with these simple tasks?

Sorting & Searching: "why bother about them?"

Unordered $a'_i = a'_i[i]$

$a: \boxed{1\ 3\ 2\ 7\ 6\ 4\ 8\ 9}$

$\left\{ a': \boxed{4\ 2\ 3\ 6\ 7\ 8\ 9} \right.$

\uparrow sorted incr asc order

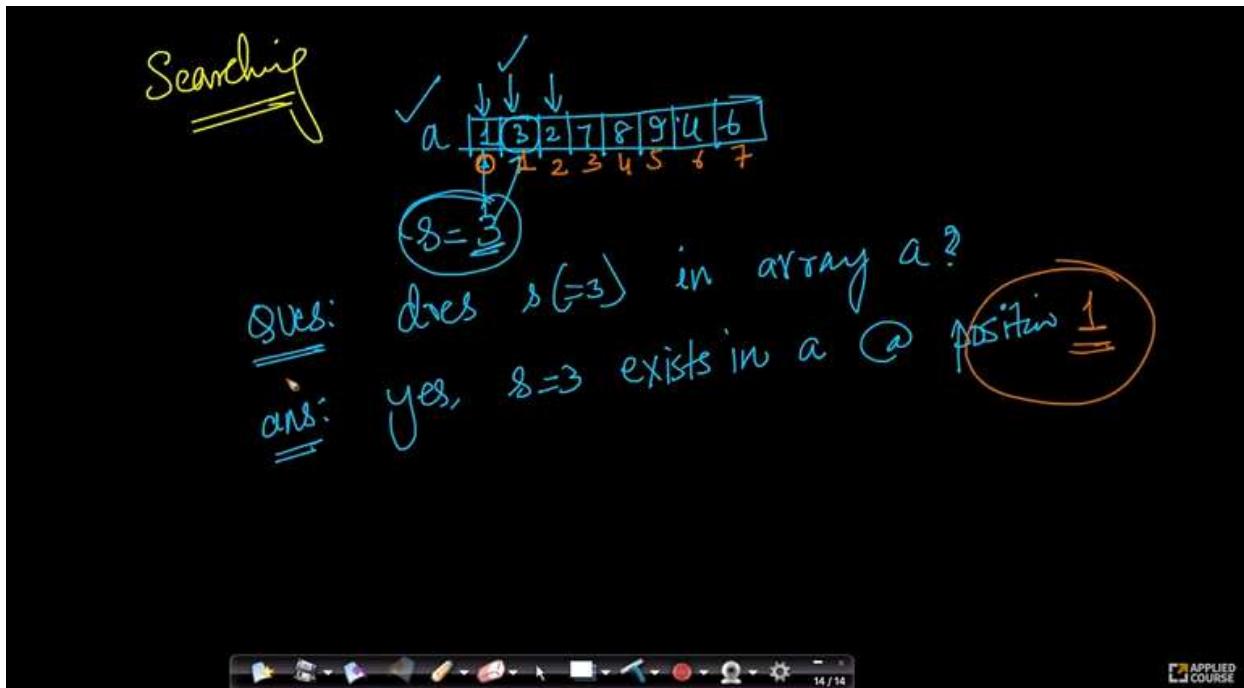
$a'': \boxed{9\ 8\ 7\ 6\ 4\ 3\ 2\ 1}$

\leftarrow sorted desc order

$a''_0 \geq a''_1 \geq a''_2 \geq \dots \geq a''_7$

Timestamp: 3:57

Sorting: Rearranging the elements of an array in either ascending order or descending order.



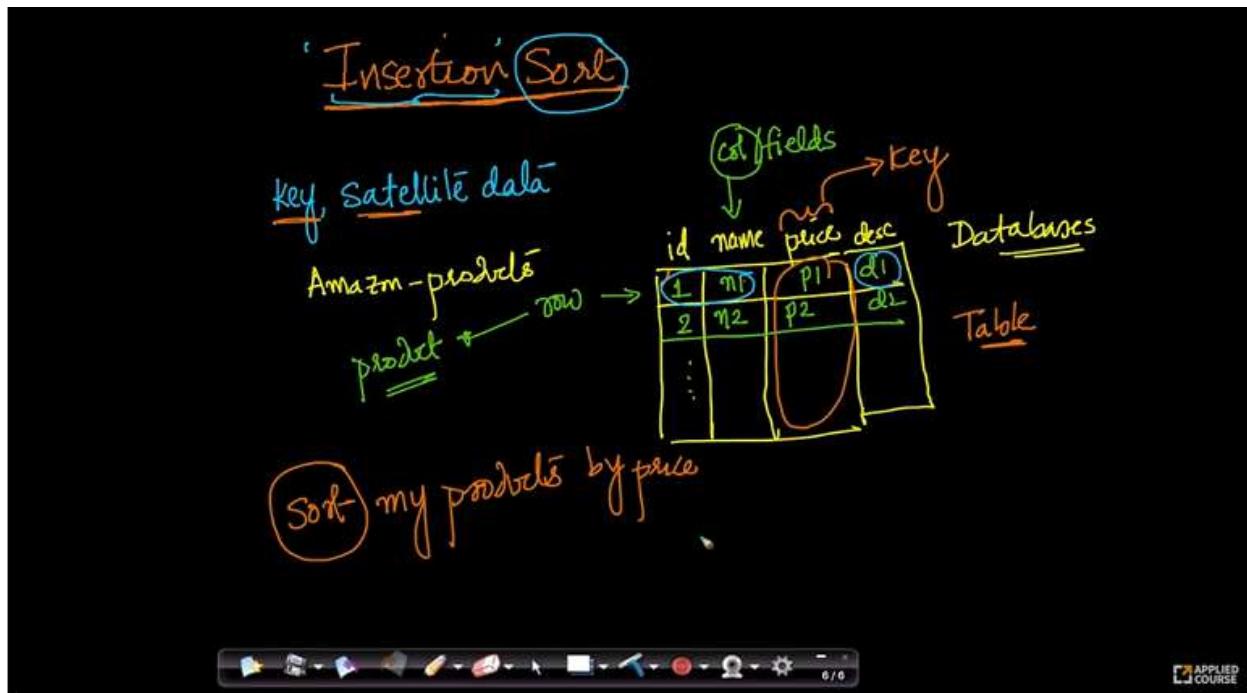
Timestamp: 7:21

Searching: It is the process of finding whether an element occurs in an array or not.

Why bother about Sorting and Searching:

- 1) Searching and sorting are widely used in a lot of real world scenarios for example while Search for a product on amazon, searching for a friend on facebook, sorting the products based on price in amazon search results, ranking the web pages in the google search results, etc.
- 2) These are the most basic operations based on which more complex things can be built.

8.2 Satellite Data and Keys

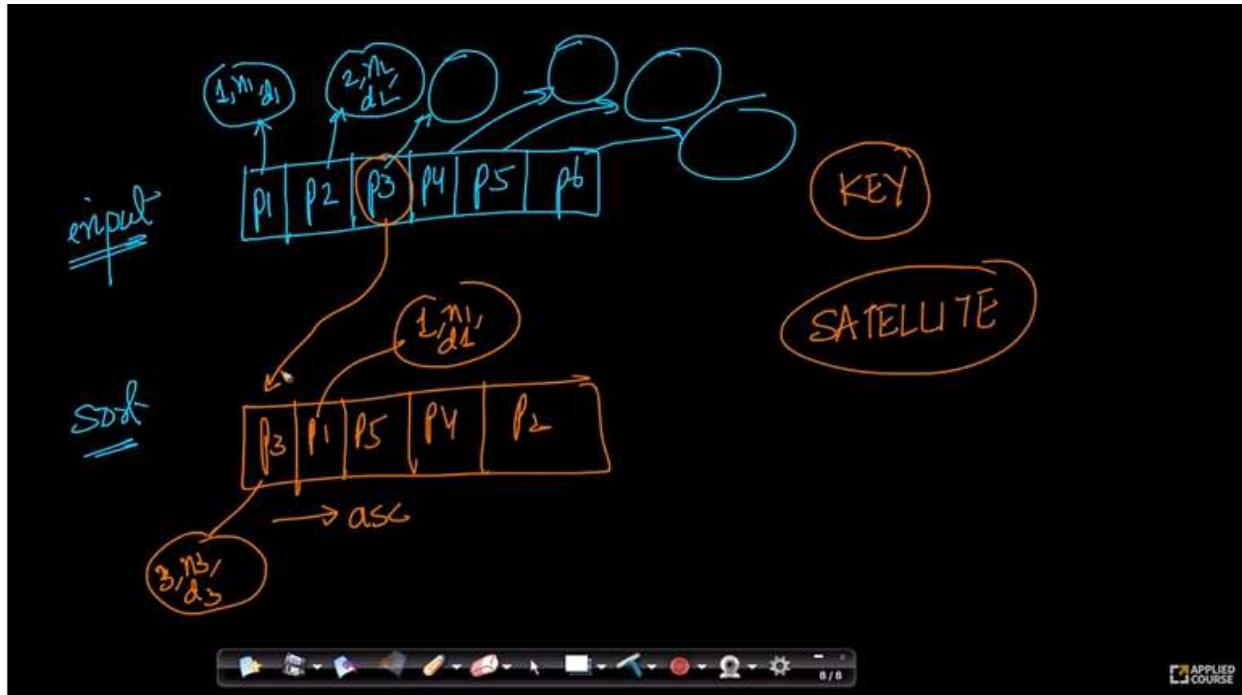


Timestamp: 3:51

Let us say we have amazon products data with id, name, price and description for each product. Suppose we want to sort the amazon products data by price, then price is the key and the rest of the data other than price is called the satellite data.

Key: The key, information or the aspect using which you want to sort the whole table is called the Key.

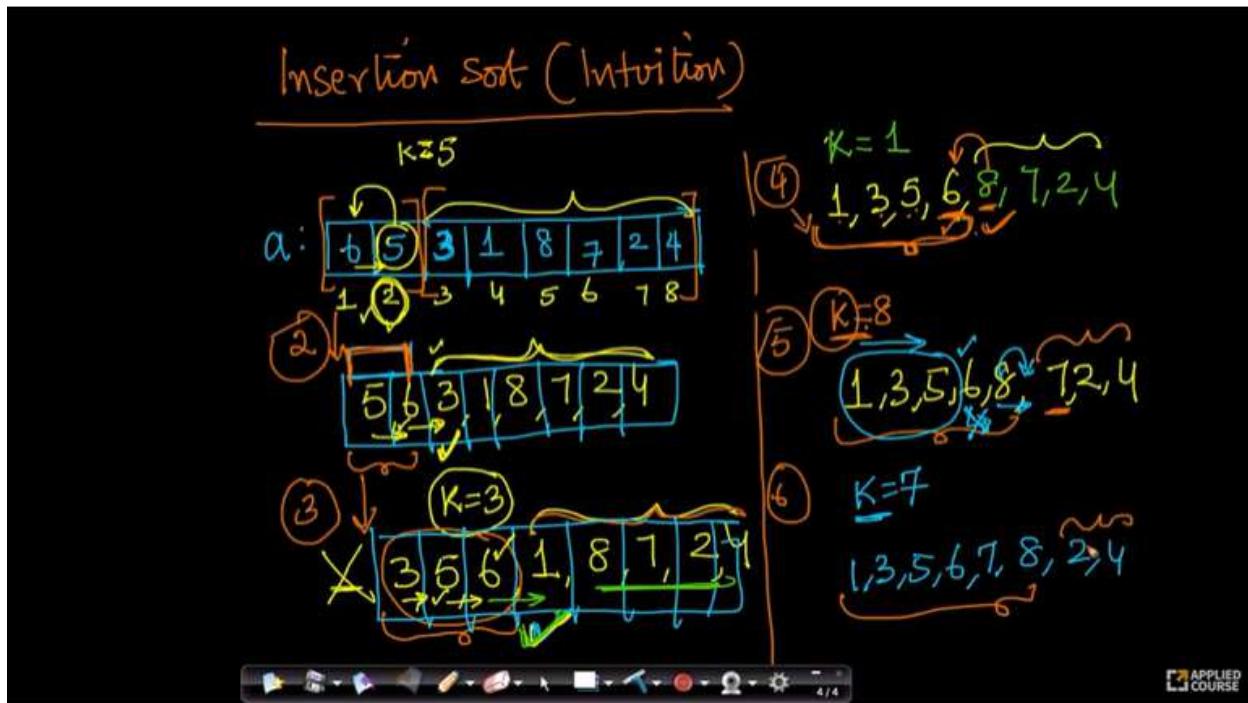
Satellite Data: The rest of the data that is not part of the key is called the satellite data.



Timestamp: 9:07

Whenever we sort an array using the key, the satellite data moves along with it, hence giving its name. (Just as how the moon, a natural satellite of earth, moves along with earth).

8.3 How it works: Card - sorting



Timestamp: 12:04

Note: Unless explicitly stated, sorting an array means sorting in ascending order.

- 1) Insertion Sort works by maintaining two subarrays, one subarray is the sorted part of the array while the other subarray is the non-sorted array.
- 2) It works by taking the first element of the unsorted array in every iteration then inserting it in the appropriate position in the sorted subarray.
- 3) When each element from the unsorted subarray is being added at the appropriate place in the sorted subarray, the size of the sorted subarray increases and the size of the unsorted subarray decreases until the sorted subarray contains all the elements of the array and the unsorted subarray has no elements.
- 4) When we move each element in the unsorted subarray to the sorted subarray, we start by first comparing the first element of the unsorted subarray (let's say K) to the rightmost element in the sorted subarray and then we move left along the sorted subarray and keep comparing them with K .

- 5) When moving from right to left along the sorted subarray and comparing the elements in the sorted subarray to K, we may come across two situations.
- K is less than the element that we are comparing.
 - K is equal or greater than the element that we are comparing.
- a) If K is less than the element we are comparing, then we move K to the left and move that element to the right.
- b) If K is greater than or equal to the element we are comparing, then the sorted subarray along with the element K is perfectly sorted. Hence we can move on with the next element in the unsorted subarray.

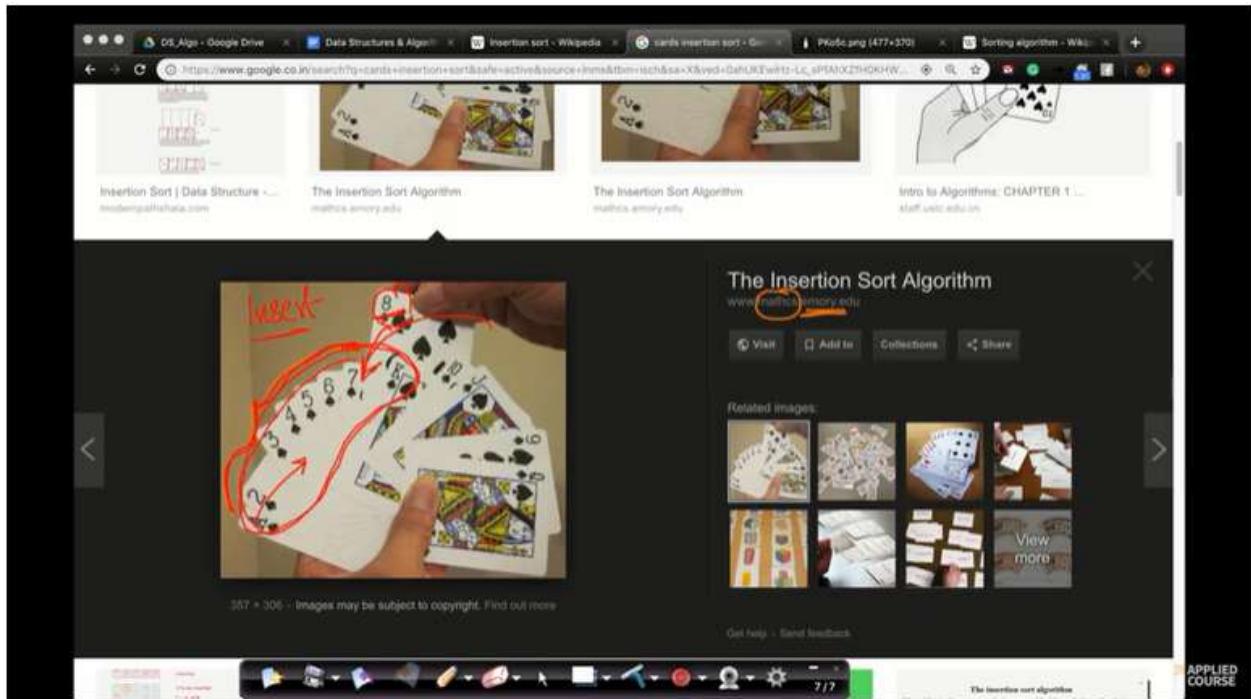
Let us look at both these condition through the below example:

ex) Let us say at some point in our insertion sort we have the array as follows
Where array = [1,3,5,6,8,7,2,4] where [1,3,5,6,8] as the sorted subarray and [7,2,4] as the unsorted subarray

Now since 7, the first element of unsorted subarray, K, is less than 8. We come across condition a.

Hence the array becomes: array= [1,3,5,6,7,8,2,4]

Now since 7, is greater than 6. We come across condition b. Notice that 7 has found its perfect place in the sorted array. Hence we move on with the next element in the unsorted array and make K=2.



Timestamp: 17:56

When we sort a pack of cards, we sort them from left to right. At any time while sorting, let us say we have card A,2,3,4,5,6,7 already sorted, now we insert the current card 8 in its right position in the sorted array, thereby giving the name of this sorting algorithm **Insertion sort**.

Please refer to the gif in the link https://en.wikipedia.org/wiki/Insertion_sort#Algorithm

8.4 Pseudo Code

The slide shows the following content:

✓ Insertion Sort (A)

```
1   for j=2 to A.length {  
2     key = A[j]  
3     // Insert A[j] into sorted A[1...j-1]  
4     i=j-1  
5     while i>0 AND A[i]>key  
6       A[i+1]=A[i]  
7       i=i-1  
8     A[i+1]=key
```

Diagram of array A:

1	2	3	4	5	6	7	8
6	5	3	1	8	7	2	4

Timestamp: 8:04

Please try the above pseudo code for the given example on the right and check whether the array is sorted at the end of the pseudo code.

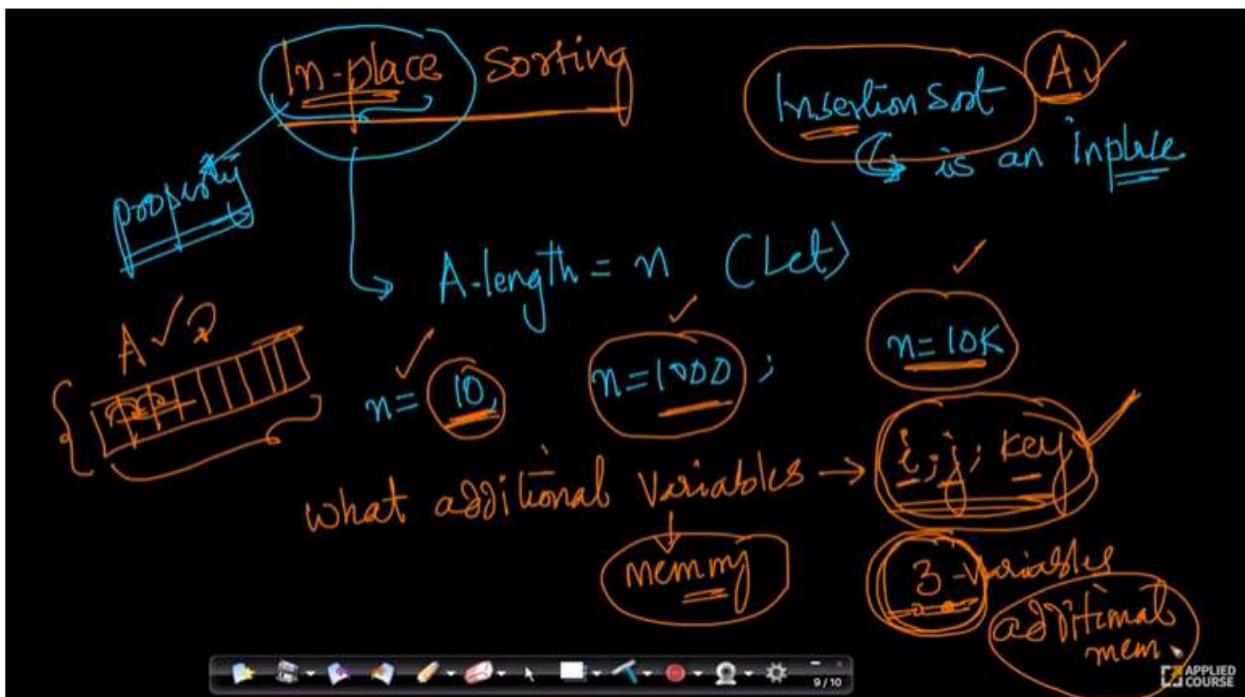
8.6 Correctness

Correctness: In the case of insertion sort, we want to prove that the insertion sort algorithm works perfectly to sort an unsorted array.

One argument to improve the correctness of insertion sort is as follows: At any time during the insertion sort, we have an already sorted subarray and we are taking the first element from the unsorted subarray and inserting it in the sorted subarray at the appropriate place where it belongs such that the already sorted subarray along with the new added element becomes a new sorted subarray.

As the insertion sort progresses, we are adding more and more elements into the sorted subarray and reducing the elements of the unsorted subarray until we are left with the sorted subarray that contains all the elements of the original array and the unsorted subarray doesn't contain any elements. Thereby proving the correctness of insertion sort.

8.7 Inplace Sorting



In Insertion sort, apart from the original array that is to be sorted we are only using three additional variables (i, j, key in the pseudo code) irrespective of the size of the input array.

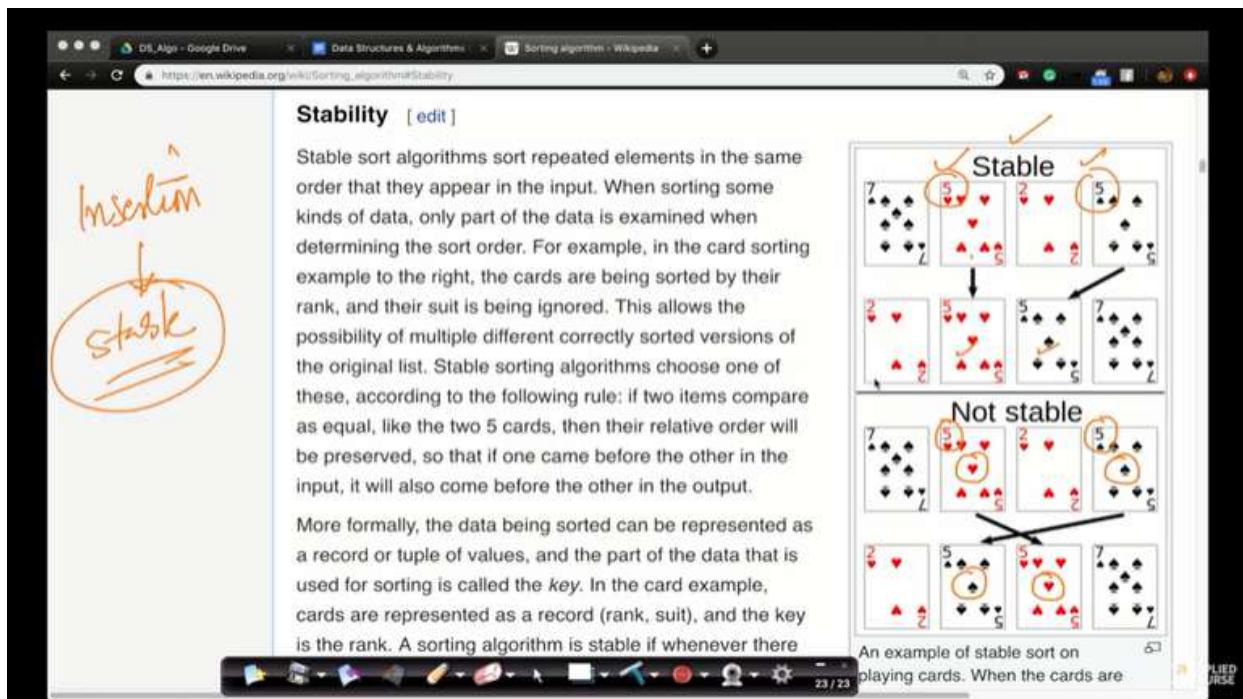
So by doing this, we are using only an additional memory for 3 variables apart from the memory used by the input array. This number(3) of variables is independent of what the size of the input array.

Such algorithms which consume a **constant** amount of **additional** memory irrespective of the input size are said to be **inplace**(meaning: within itself) algorithms. Since Insertion sort agrees the above, it is an inplace algorithm.

8.8 Stable Sort

When the initial order of the element having the same key value is retained even after sorting using a sorting algorithm then such an algorithm is called a stable sorting algorithm and is said to have stability.

Insertion sort is a stable sorting algorithm.



Timestamp: 5:26

Notice in the above diagram, for a stable algorithm that uses the number on the card as the key for sorting, the initial order of the 5 of hearts and 5 of spades is maintained even after sorting.

Whereas a non-stable algorithm **doesn't guarantee** that the initial order will be preserved post sorting as in the example of the above figure.

Why?

price	name
23	apple
36	lenovo
31	acer
31	asus
40	hp
60	lenovo

Sort by name alphabetically

price	name
31	acer
23	apple
31	asus
40	hp
36	lenovo
60	lenovo

Sort by price
STABLE

price	name
23	apple
31	acer
31	asus
36	lenovo
40	hp
60	lenovo

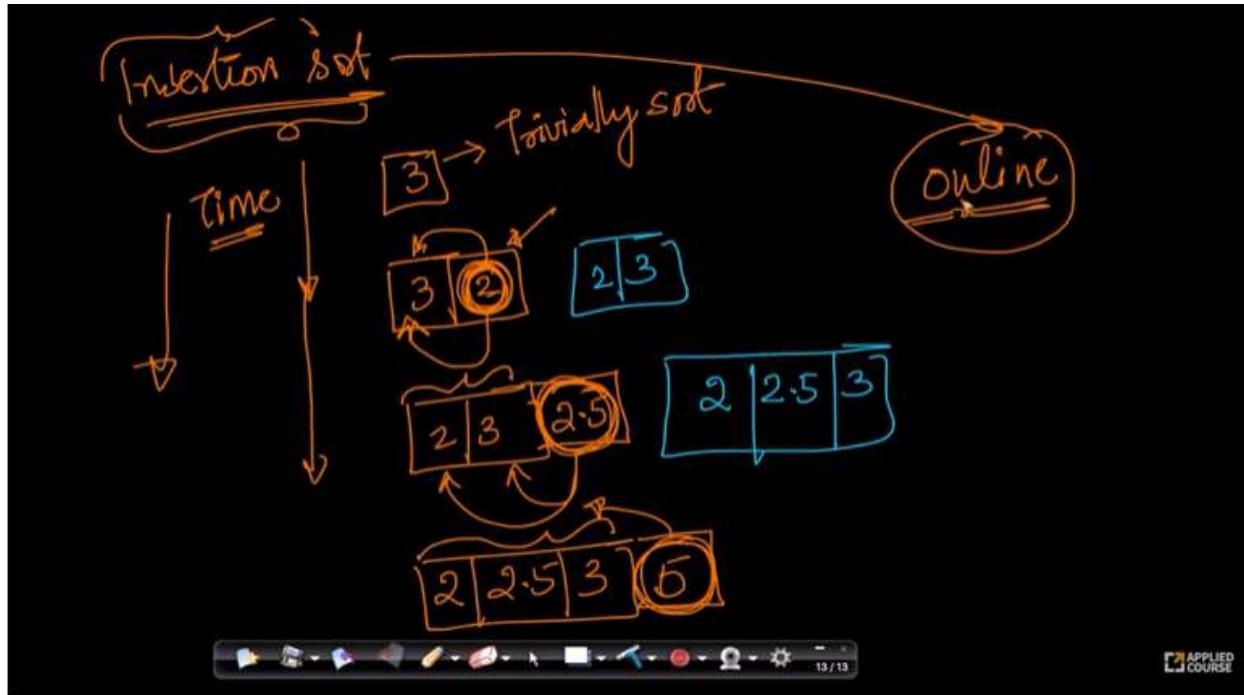
Timestamp: 14:15

Why use stable sorting algorithms: In real world there might be multiple scenarios like the one above, where having the data sorted by price, you also wanted your end results to have the data sorted by other column, in this case, the name of the product.

In these cases you need a stable sorting algorithm like insertion sort that maintains the initial order of the products (in this case the order of Acer and Asus) that have the same key value (here price of 31).even after sorting.

8.9 Online Sorting

Many at times, in real world cases such as the case of online cab services, the data that needs to be sorted may not be given to us at a time. Rather, the data is given to us over a time. If our sorting algorithm can sort data that is given to us over time then the sorting algorithm is said to be an online sorting algorithm.

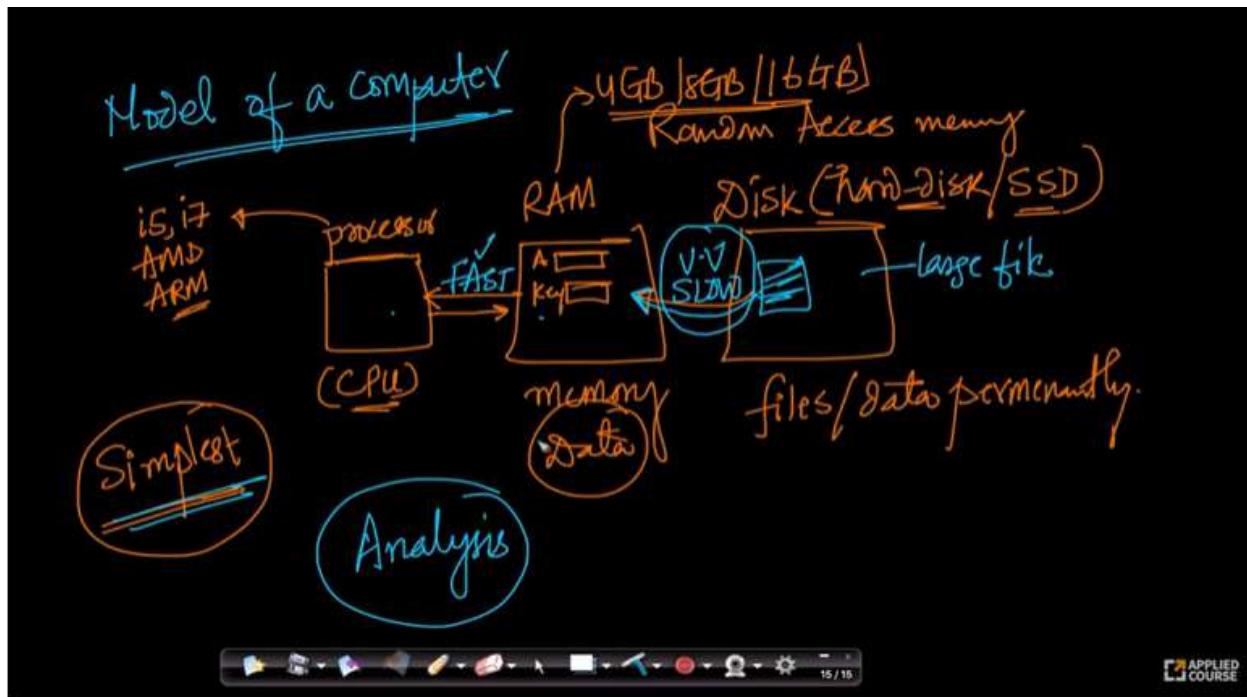


Timestamp: 9:48

For example as shown in the above figure, let us say we were sent expected time for reaching us by different cab drivers and these expected times are sent to us over a period of time. So at any time t , we wanted to sort whatever data we receive till then as shown in the above figure.

Insertion sort, in the way it works by inserting a new element into an already sorted subarray, it can be trivially converted to an online sorting algorithm. Insertion sort is an online learning algorithm that can sort the data as the data arrives over time.

9.1 Model of Computation



Timestamp: 7:38

For analysis of the standard algorithms and datastructures we will use the above simple model of the computer which contains the processor(also called cpu), the RAM and the Disk.

Processor: This is the electrical component responsible for all computations that take place in the computer. It gets the data required for the computation from the RAM and writes the output of the computation to the RAM.

RAM: RAM is a temporary storage device also called memory that stores the data to feed into the processor and collects the output data from the processor. It gets the data from the permanent storage i.e disk and if required can write data to the disk. The data inside the RAM is lost when the computer is shut down.

Disk: Disk is a permanent storage that stores all the data. Whenever this data is required by the processor it supplies that data to the RAM which in turn supplies to the processor.

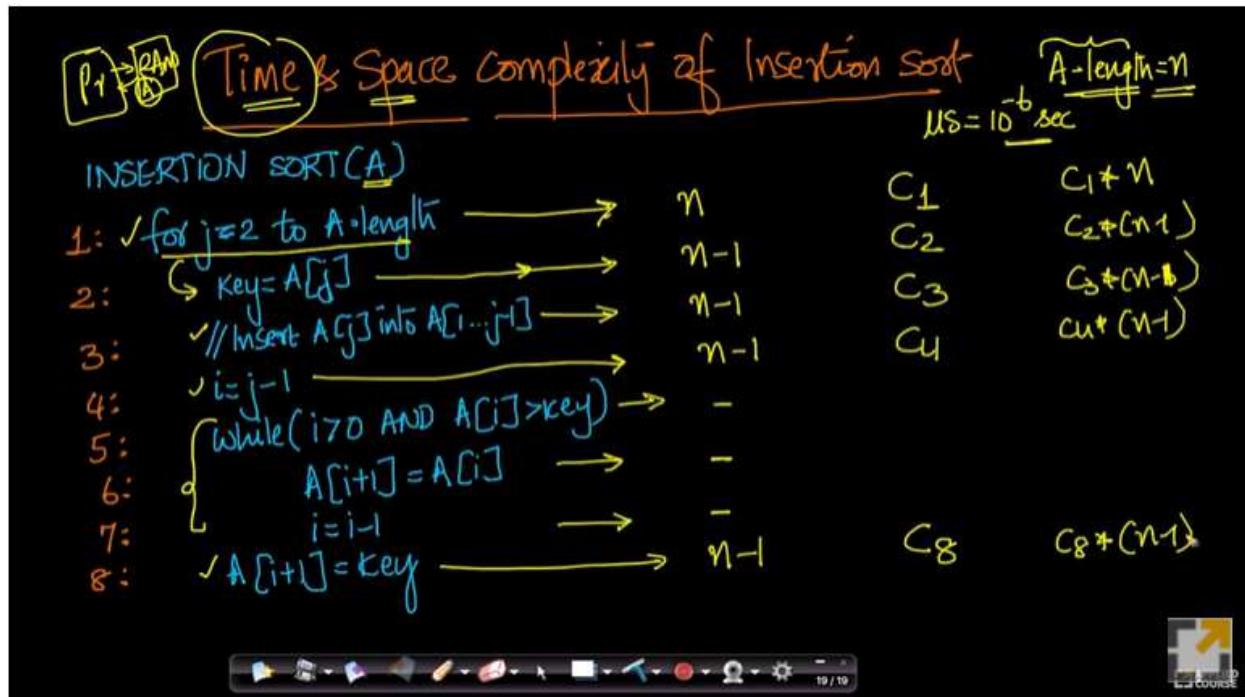
There are other models of computer involving cache(an on-chip memory for storing values that are mostly used), multi-core cpus (capable of parallel computing),etc. But as for the discussion of standard algorithms and datastructures we only use the above simple model of the computer.

9.2 Space and Time Analysis of Insertion Sort-1

P1 F1 Time & Space Complexity of Insertion Sort A-length=n
 $\mu s = 10^{-6} \text{ sec}$

INSERTION SORT(A)

1: $\forall j=2 \text{ to } A.length$	$\rightarrow n$	C_1	$C_1 * n$
2: ↳ $\text{key} = A[j]$	$\rightarrow n-1$	C_2	$C_2 * (n-1)$
3: $\forall // \text{Insert } A[j] \text{ into } A[1..j-1]$	$\rightarrow n-1$	C_3	$C_3 * (n-1)$
4: $\forall i=j-1$	$\rightarrow n-1$	C_4	$C_4 * (n-1)$
5: $\forall \text{while}(i>0 \text{ AND } A[i]>\text{key})$	$\rightarrow -$		
6: $\forall \quad A[i+1]=A[i]$	$\rightarrow -$		
7: $\forall \quad i=i-1$	$\rightarrow -$		
8: $\forall A[i+1]=\text{key}$	$\rightarrow n-1$	C_8	$C_8 * (n-1)$



Timestamp: 6:39

In this video we start analysis the space and time complexity of the insertion sort algorithm based on its pseudo code.

Notice that the first line of the code, the 'for' statement runs for n times while j takes values from 2 to $n+1$ (although the loop executes from 2 to n but at $n+1$ it has to check for condition and terminate). Let us say it takes C_1 amount of time to execute it once. Then the amount of time it takes to run n times is $C_1 * n$.

Since the loop runs for $n-1$ times (from $j=2$ to n) each statement in the loop 2,3,4,8 runs for $n-1$ times(We will discuss the run time of remaining statements in the next video), assuming that the time taken to execute code in lines 2,3,4,8 for one time is C_2,C_3,C_4, C_8 the total time taken for executing code in lines 2,3,4,8 are $C_2*(n-1), C_3*(n-1), C_4*(n-1), C_8*(n-1)$ respectively.

We will discuss the time taken for executing remaining statements and the total time for executing the complete algorithm in the next video.

9.3 Space and Time Analysis of Insertion Sort-2

Time & Space complexity of Insertion sort

$MS = 10^{-6} \text{ sec}$ $A\text{-length} = n$

INSERTION SORT(A)

```

1: for j=2 to A.length
2:   Key=A[j]
3:   // Insert A[j] into A[1..j-1]
4:   i=j-1
5:   while(i>0 AND A[i]>key)
6:     swap(A[i+1]=A[i])
7:     i=i-1
8:   A[i+1]=key
  
```

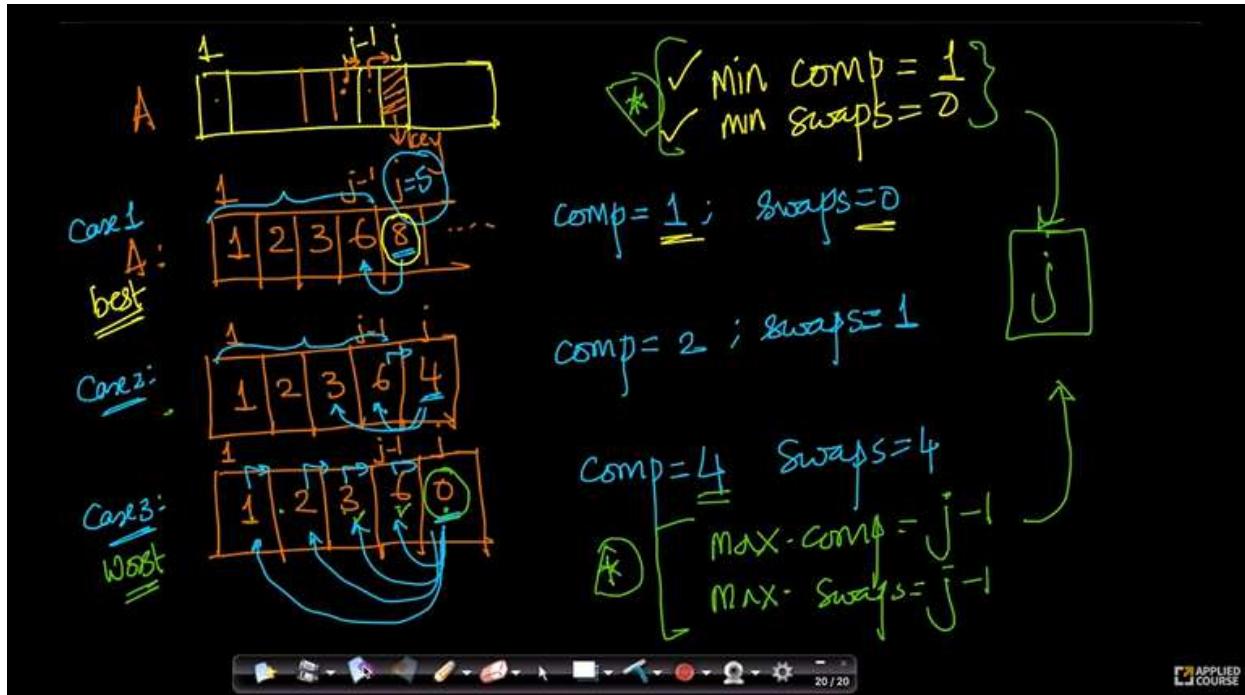
n $n-1$ $n-1$ $n-1$ i $i-1$ $i-1$ $n-1$	c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8	$c_1 * n$ $c_2 * (n-1)$ $c_3 * (n-2)$ $c_4 * (n-1)$ $c_5 * (n-1)$ $c_6 * (n-1)$ $c_7 * (n-1)$ $c_8 * (n-1)$
--	--	--




Timestamp: 2:42

Notice that the operation that is taking place at line 5 is comparison and at line 6 is swap(not two elements swap, but like moving element from one position to other) and at line 7 simple decrement operation.

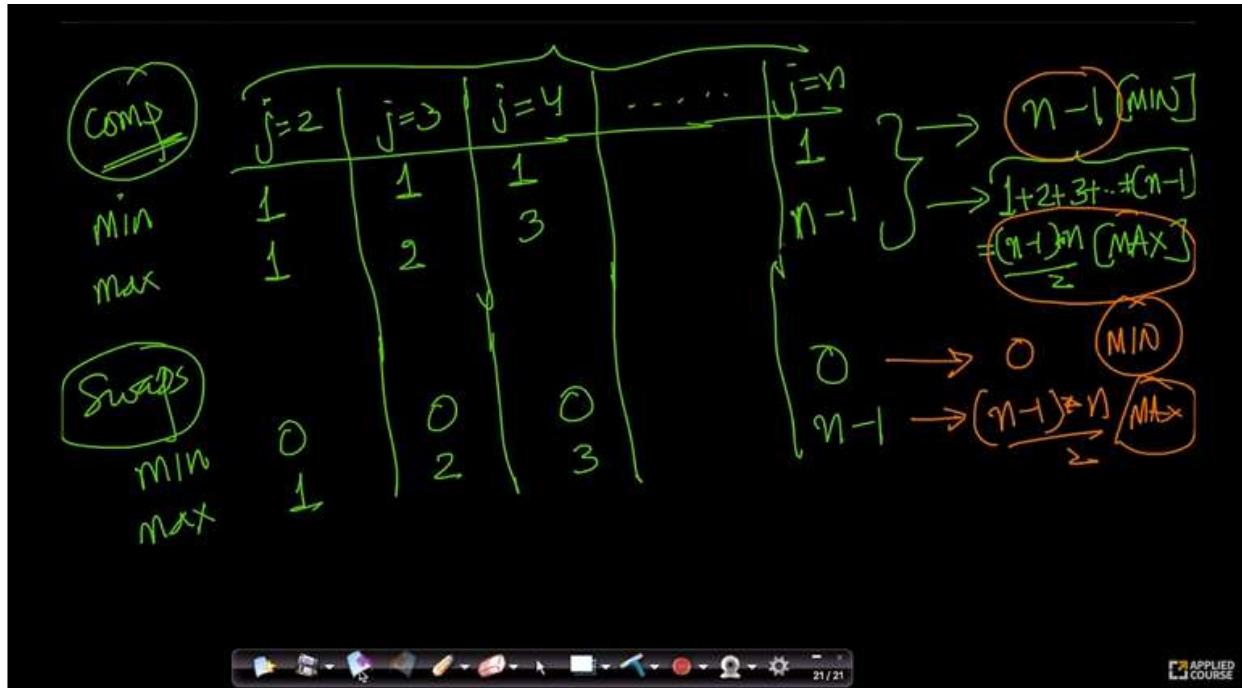
To understand, how many times each operation is executed, please look at the below figure.



Timestamp 7:04

Notice in the above figure, in the best case for any j we need to do 1 comparison and 0 swaps (since in this case $A[j]$ 8 is greater than 6) while in worst case for any j we need to do $j-1$ comparisons and $j-1$ swaps (like in above figure we need to move 0 to the left after comparing with each element)

Notice that j takes values from $j=2$ to $j=n$, hence we have to calculate the total time for both the swap and the comparison for all values of j . Please take a look at the below figure.



As in the above figure,

In best case the number of comparisons is 1 for a single value of j , so for $j=2$ to n . Number of comparisons is $1*(n-1) = n-1$.

In worst case the number of comparisons is $j-1$ for a single value of j , so for $j=2$ to n . Number of comparisons is $(2-1) + (3-1) + \dots + (n-1) = n*(n-1)/2$

In best case the number of swaps is 0 for a single value of j , so for $j=2$ to n . Number of comparisons is $0*(n-1) = 0$

In worst case the number of swaps is $j-1$ for a single value of j , so for $j=2$ to n . Number of comparisons is $(2-1) + (3-1) + \dots + (n-1) = n*(n-1)/2$

Time & Space Complexity of Insertion Sort

$A.length = n$

$MS = 10^{-6} \text{ sec}$

INSERTION SORT(A)

```

1: for j=2 to A.length
2:   Key = A[j]
3:   // Insert A[j] into A[1..j-1]
4:   i=j-1
5:   while(i>0 AND A[i]>key)
6:     Swap(A[i+1]=A[i])
7:     i=i-1
8:   A[i+1]=key
  
```

Time Complexity Analysis:

- $C_1 = C_1 * n$
- $C_2 = C_2 + (n-1)$
- $C_3 = 0$
- $C_4 = C_4 + (n-1)$
- $C_5 = (C_5 * n) + \frac{n(n-1)}{2}$
- $C_6 = 0$
- $C_7 = 0$
- $C_8 = C_8 * (n-1)$

Timestamp: 12:57

Similar to the previous video let us say the time taken for executing the code in line 5,6,7 once be C_5, C_6, C_7 .

In best case, code in line 5 runs $n-1$ times, line 6 0 times and line 7 0 times.

In worst case, code in line 5 runs $n*(n-1)/2$ times, line 6 $n*(n-1)/2$ times and line 7 $n*(n-1)/2$ times.

Note that since 3rd line is a comment, $C_3=0$

So the total time taken for executing all the lines in best case and worst case is as below.

Diagram illustrating the time complexity of Insertion Sort:

Best Case: $T(n) = C_1 * n + C_2 * (n-1) + \dots + C_8 * (n-7)$

Worst Case: $T(n) = \max(C_1 * n, C_2 * (n-1)/2, C_3 * (n-2)/2, \dots, C_8 * (n-7)/2)$

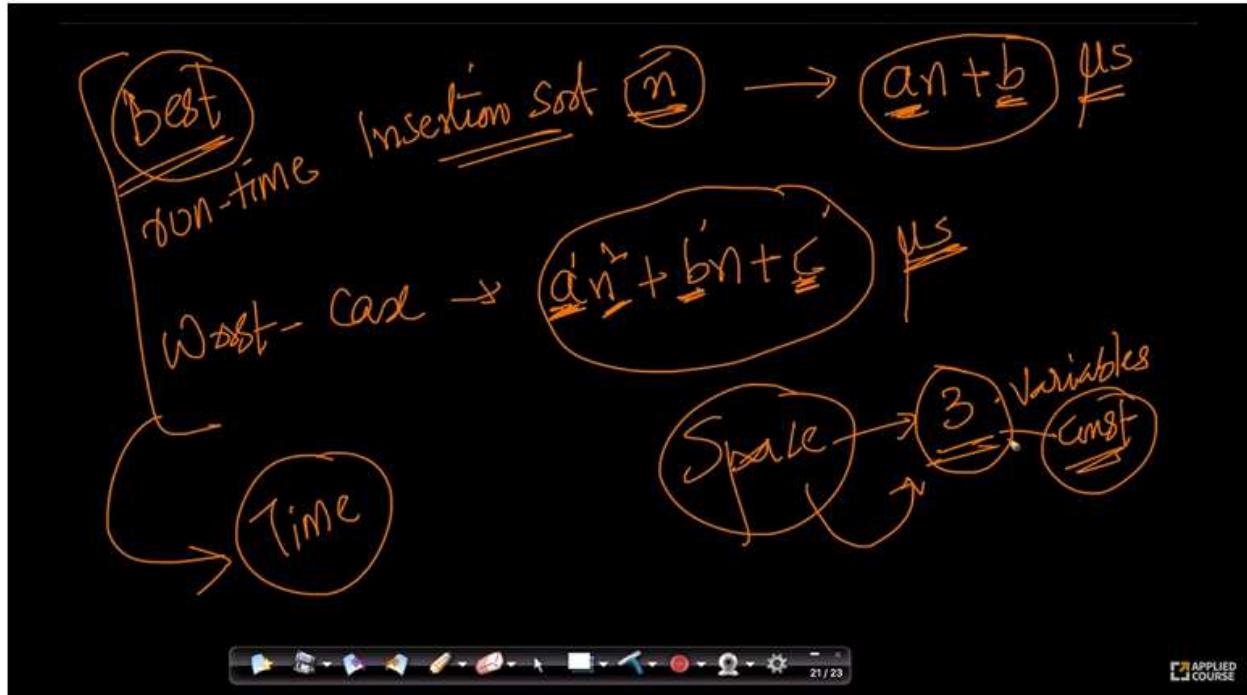
The final result is $a'n^2 + b'n + c$.

Timestamp 18:14

So as shown in the above figure

Best case time taken for executing the insertion sort in best case is $an+b$ for a and b be as some combination of $C_1, C_2 \dots C_8$

Worst case time taken for executing the insertion sort in worst case is $a'n^2 + b'n + c'$ for a', b', c' as some combination of $C_1, C_2 \dots C_8$ and /2's.



Timestamp: 21:41

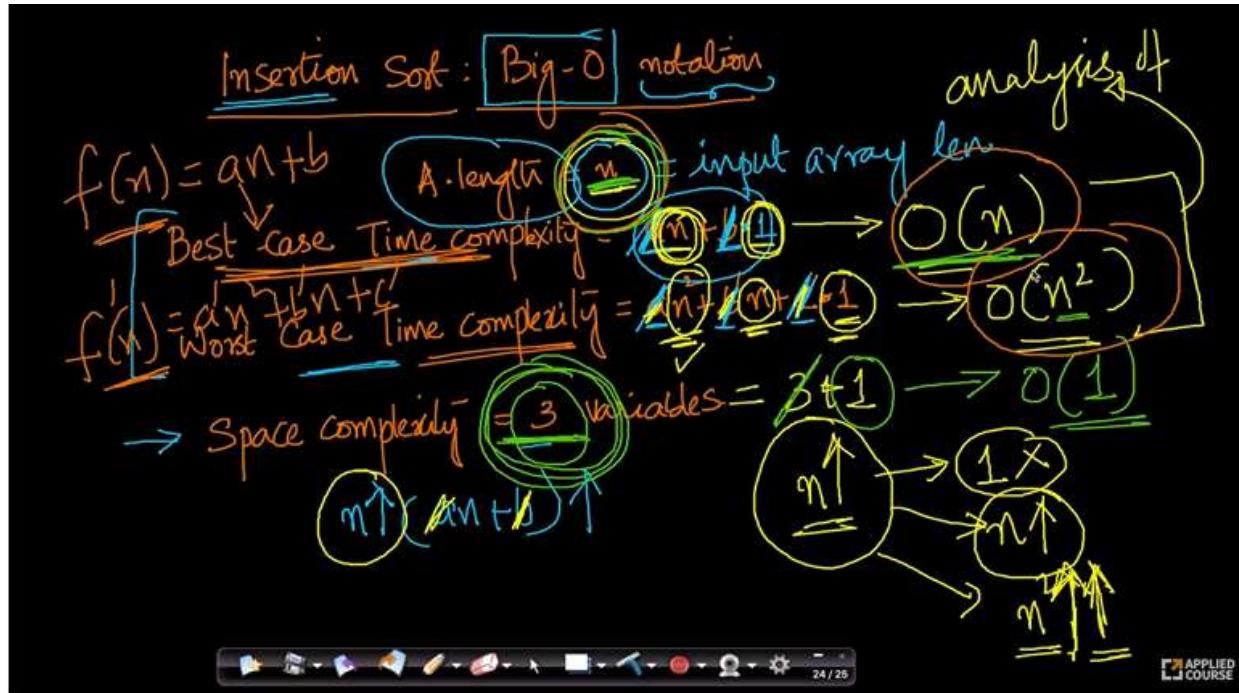
Notice that space and time complexity are expressed in terms of n , the size of the input. As for space complexity, whatever the size of the input, we are only using 3 **additional** variables apart from the actual input array. Since this 3 is independent of the size of the array n we treat it as constant.

Hence concluding, insertion sort has best case time complexity of $a'n + b$, worst time complexity of $a'n^2 + b'n + c'$ and constant (3 variables) space complexity for input size n .

9.4 Insertion Sort: Big O-notation

Remember that in the previous video, we discussed that

"Hence concluding, insertion sort has best case time complexity of $a'n + b$, worst time complexity of $a'n^2 + b'n + c'$ and constant (3 variables) space complexity for input size n ."



Timestamp: 14:21

The best case time complexity of insertion sort $a'n + b$ for the simple case of $a=1$ and $b=1$ is $n+1$. When $n=1$, its value is $1+1=2$, when $n=10$ its value becomes $10+1=11$, when $n=100$ its value becomes $100+1=101$.

Notice that as n increases the best case time complexity of n increases proportional to n . Hence its time complexity is said to be $O(n)$.

The worst case time complexity of insertion sort $a'n^2 + b'n + c'$ for the simple case of $a'=1$, $b'=1$ and $c'=1$ is n^2+n+1 .

When $n=1$, its value is $1+1+1=3$, when $n=10$ its value becomes $100+10+1=111$, when $n=100$ its value becomes $10000+100+1=10101$.

Notice that as n increases the worst case time complexity of n increases proportional to n^2 . Hence its time complexity is said to be $O(n^2)$. Although it also increase with n , the increase with n^2 is much more larger when compared to the increase with n .

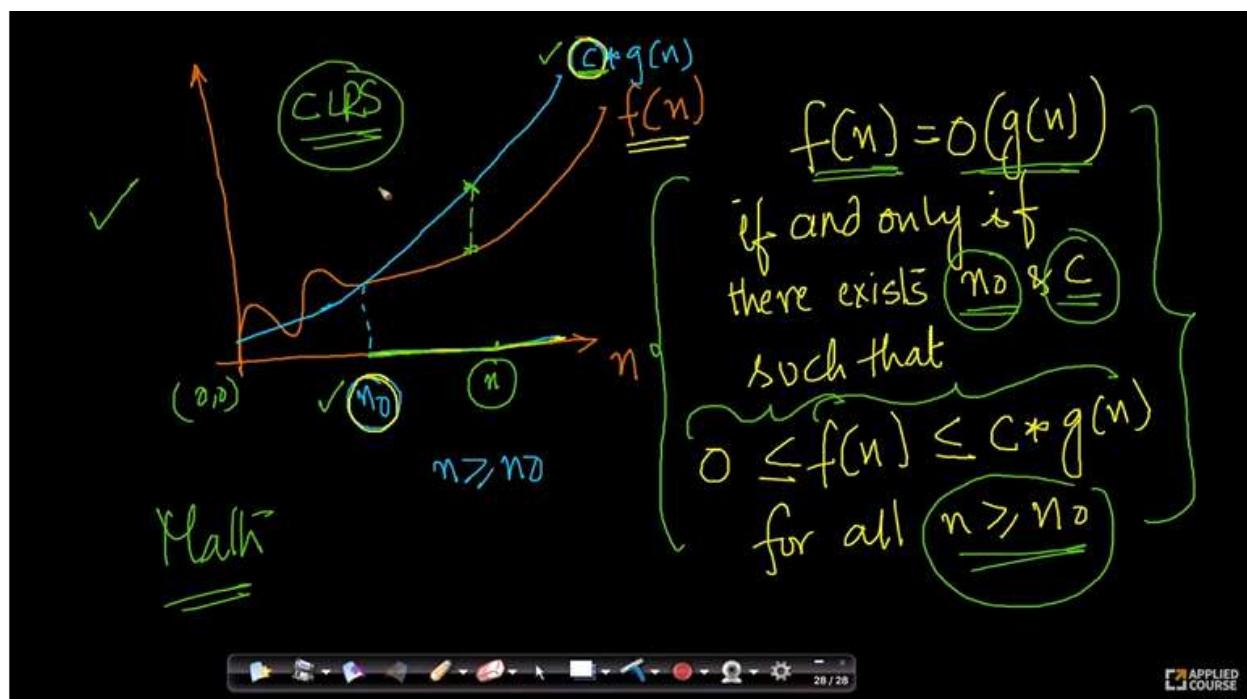
Please have a look at the graphs in the below link to understand how the values of n and n^2 increases as n increases.

<https://stackoverflow.com/questions/23329234/which-is-better-on-log-n-or-on2>

Coming to the space complexity, as n increases the space complexity does not increase and remains same(3 variables) hence insertion sort is said to have constant space complexity and is denoted by $O(1)$.

To conclude, the best case time complexity of insertion sort is $O(n)$, the worst case time complexity of insertion sort is $O(n^2)$ and the space complexity of insertion sort is $O(1)$.

9.5 Notations : Big O



Timestamp: 6:10

As shown in the above figure, for any functions $f(n)$ and $g(n)$, function $f(n)$ is said to be $O(g(n))$ if and only if for some constants n_0 and c , $0 \leq f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.

$f(n) = (2n^2 + 1 \cdot n + 3) \rightarrow O(n^2)$

IO (let)

$2n^2 + 1 \cdot n + 3 \leq (\underline{c}) \cdot n^2 \text{ for all } n \geq \underline{n_0}$

$2n^2 + 1 \cdot n + 3 \leq 10n^2 \text{ for all } n \geq \underline{n_0}$

Timestamp: 10:15

Let us say we have some function $f(n) = 2n^2 + n + 3$ to prove that it is $O(n^2)$ (Note: $g(n) = n^2$), we need to find some c and some n_0 that satisfies the condition:

$$2n^2 + n + 3 \leq c \cdot n^2 \text{ for all } n \geq n_0$$

$$\text{Let } c=10 \text{ then } 2n^2 + n + 3 \leq 10n^2 \text{ for all } n \geq n_0$$

We need to find n_0 that satisfies this condition.

$n \geq 1$ $2n^2 + n + 3 \leq 10n^2$ for all $n \geq n_0$
 $2 + \frac{1}{n} + \frac{3}{n^2} \leq 10$ for all $n \geq 1$

$n=1 \rightarrow 2 + 1 + \frac{3}{1} \leq 10 \checkmark$
 $n=2 \rightarrow 2 + \frac{1}{2} + \frac{3}{4} \leq 10$
 $n=3 \rightarrow 2 + \frac{1}{3} + \frac{3}{9} \leq 10$
 :

30 / 30

APPLIED COURSE

We can observe that for $n=1,2,3$ this inequation holds and as n increases the value of $2n^2 + n + 3$ decreases further hence it will always be less than 10. Hence for all $n \geq 1$, this inequation holds.

Therefore we are able to find such $n_0 = 1$ and $c = 10$ for which the inequation $2n^2 + n + 3 \leq c \cdot n^2$ for all $n \geq n_0$ holds. Thereby proving that $f(n)$ is $O(n^2)$.

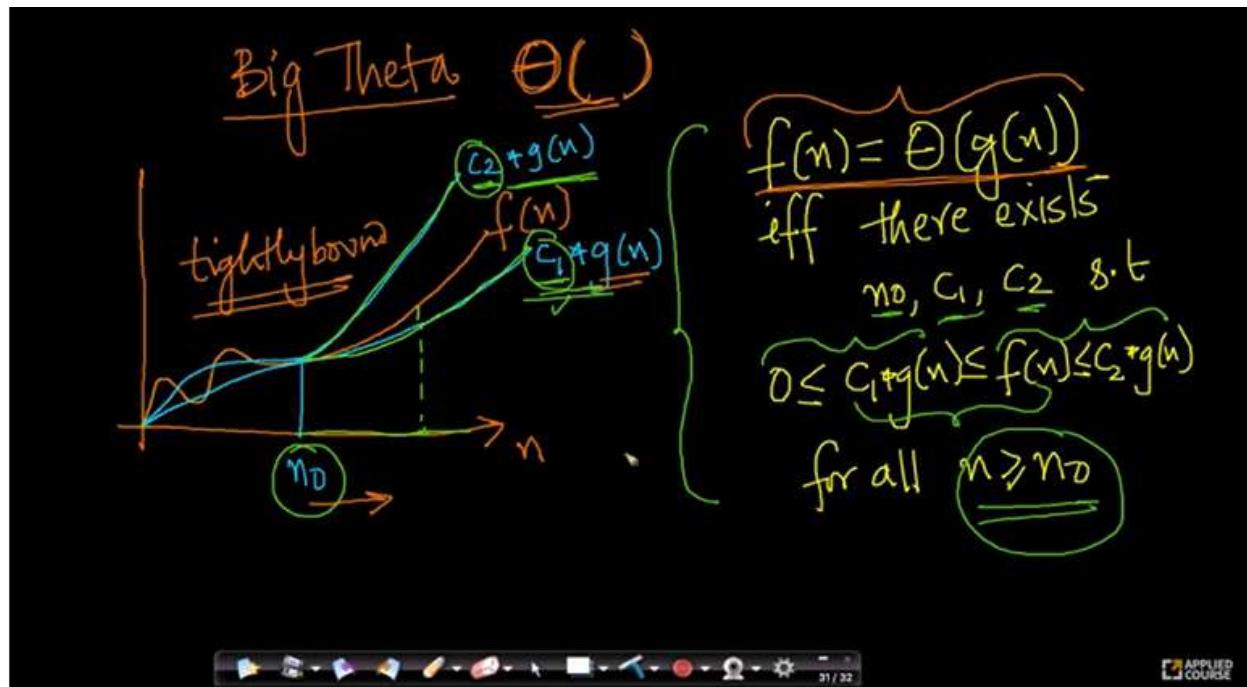
$f(n) = (2n^2 + 1 \cdot n + 3) \rightarrow O(n^2)$
 $\underbrace{a}_{2}, \underbrace{b}_{1}, \underbrace{c}_{3}$ $g(n)$
 $\checkmark 2n^2 + 1 \cdot n + 3 \leq \underline{c} \cdot n^2 \text{ for all } n \geq n_0$
 $2n^2 + 1 \cdot n + 3 \leq \underline{10} \cdot n^2 \text{ for all } n \geq n_0$
 $f(n) = O(n^2)$ $f(n) \leq \underline{10} \cdot n^2 \text{ for all } n \geq 1$

29 / 30

APPLIED COURSE

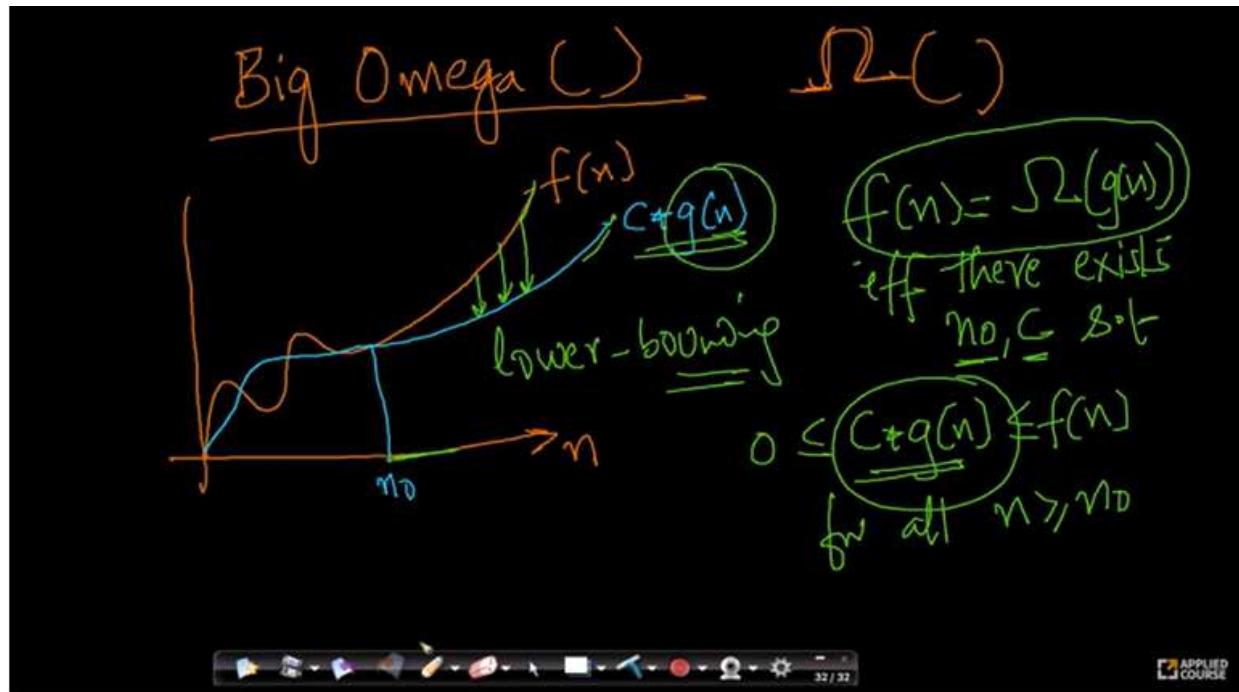
Similarly for any values of a' , b' and c' we can find n_0 and c for which the inequation satisfies.
 Therefore $f(n) = a' n^2 + b'n + c$ is always $O(n^2)$.

9.6 Notations : Big Omega, Theta



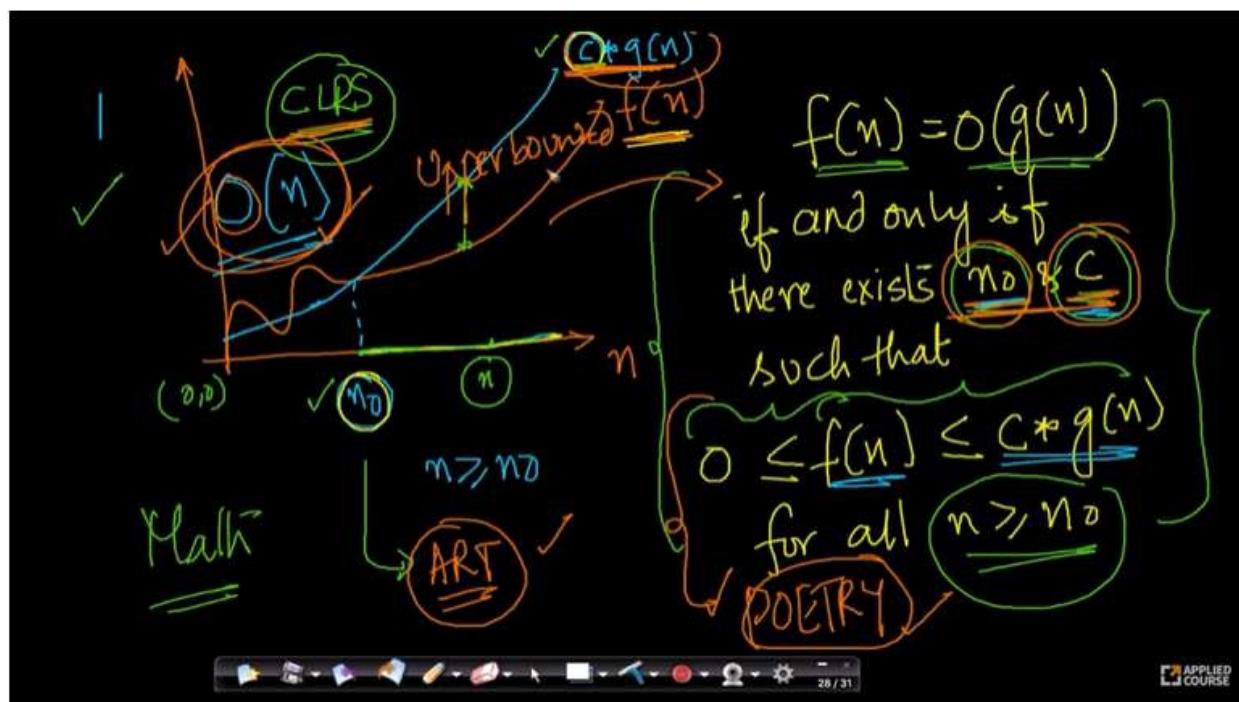
Timestamp: 4:39

If there exists two functions $f(n)$ and $g(n)$, then $f(n)$ is said to be $\Theta(g(n))$ if and only if there exists constants n_0, c_1, c_2 such that $0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n)$ for all $n \geq n_0$. We can think of $f(n)$ being tightly bounded (meaning both upper bounded and lower bounded) between $c_1 * g(n)$ and $c_2 * g(n)$ incase of $f(n) = \Theta(g(n))$.



Timestamp: 8:03

If there exists two functions $f(n)$ and $g(n)$, then $f(n)$ is said to be $\Omega(g(n))$ if and only if there exists constants n_0, c such that $0 \leq c * g(n) \leq f(n)$ for all $n \geq n_0$. We can think of $f(n)$ being lower bounded by $c * g(n)$ incase of $f(n) = \Omega(g(n))$.



Timestamp: 5:39

Incase of $f(n)=O(g(n))$, $f(n)$ is upper bounded by some constant $c^*g(n)$.

9.7 Notations : Small O, Omega, Theta

Small-Oh & Small-Omega

$f(n) = o(g(n))$ iff $\forall c > 0 \exists n_0 > 0$ such that $0 \leq f(n) < c \cdot g(n) \quad \forall n > n_0$

e.g.: $2n = O(n)$ $\boxed{2n < c \cdot n} \quad \begin{matrix} \forall n > n_0 \\ \exists c > 0 \end{matrix}$

Big-Oh $O(\Theta)$

Big-Omega

Big-Theta

iff = if and only if
 \forall = for all
 \exists = there exists

$O()$ \rightarrow upper bound

Timestamp: 23:59

Definition1: If there are two functions $f(n)$ and $g(n)$ then $f(n) = o(g(n))$ (Note $o!=O$) if and only if for all $c>0$ there exists a constant $n_0 > 0$ such that $0 \leq f(n) < c \cdot g(n)$ for all $n \geq n_0$

Alt - defn

$$f(n) = o(g(n)) \text{ iff } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Timestamp: 13:54

Alternate definition: If there are two functions $f(n)$ and $g(n)$ then $f(n) = o(g(n))$ if and only if

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$$

For example:

Function $f(n) = 2n^2$ is $O(n^2)$ but not $o(n^2)$.

Small- Ω regas (ω)

(1) $f(n) = \underline{\omega}(g(n))$ iff $g(n) = o(f(n))$

$\checkmark 2n = o(n^2) \Rightarrow n^2 = \omega(n)$

$\omega(2n) = \omega(n)$

(2) $f(n) = \underline{\omega}(g(n))$ iff $\exists c > 0$ such that $0 \leq c \cdot g(n) < f(n) \quad \forall n > n_0$

ω, Ω
 $\exists c > 0$

$0, 0$

$\frac{n^2}{2} = \omega(n); \quad \frac{n^2}{2} \neq \omega(n^2)$

APPLIED COURSE

Definition 1: If there exists two functions $f(n)$ and $g(n)$, then $f(n) = \omega(g(n))$ if and only if $g(n) = o(f(n))$.

Definition 2: If there exists two functions $f(n)$ and $g(n)$, then $f(n) = \omega(g(n))$ if and only if for all $c > 0$ there exists a constant $n_0 > 0$ such that $0 \leq c \cdot g(n) < f(n)$ for all $n > n_0$.

(3) $f(n) = \underline{\omega}(g(n))$ iff $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

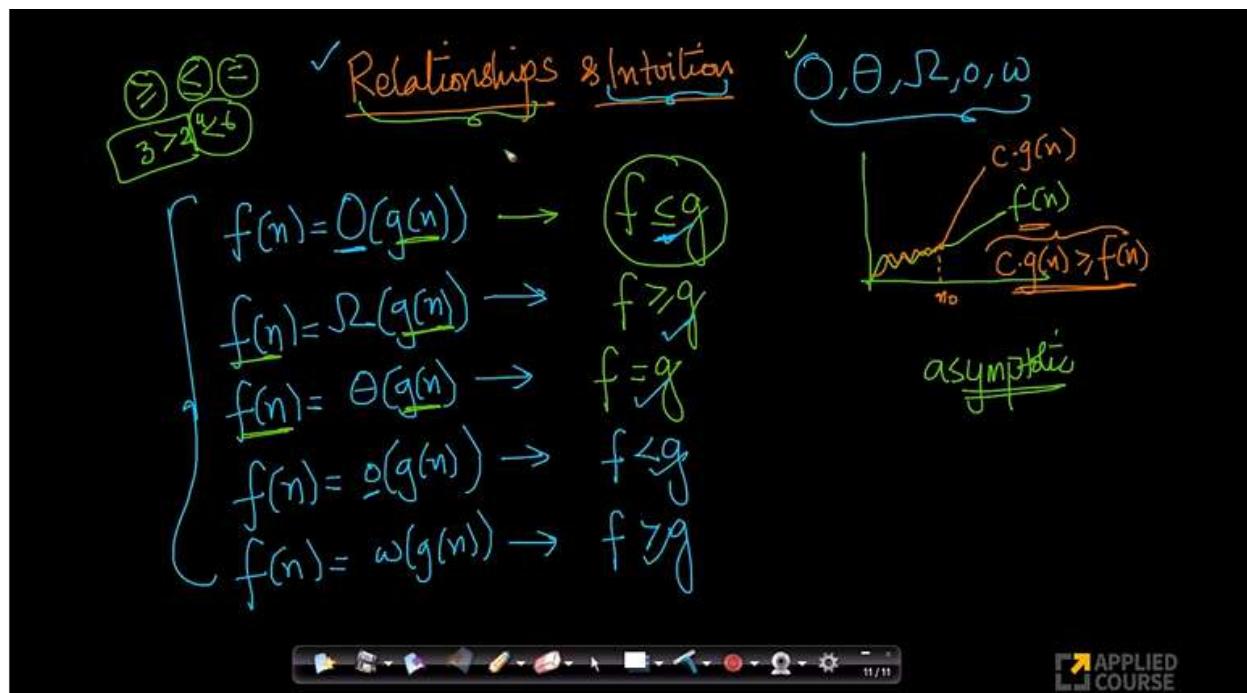
APPLIED COURSE

Timestamp: 20:36

Definition 3: If there exists two functions $f(n)$ and $g(n)$, then $f(n) = \omega(g(n))$ if and only if
 $\lim_{n \rightarrow \infty} f(n) / g(n) = 0$.

The key difference between O and ω is that the condition should be satisfied **for any** constant c in O but should satisfy **for all** $c > 0$ in case of ω . The same difference applies in case of Ω and ω as well.

9.8 Relationship between various notations



Timestamp: 4:53

Our various notations can be thought of as $f \leq g$, $f \geq g$, etc as shown in the above figure.

$f(n) = O(g(n))$ can be denoted as $f \leq g$ since f is upper bounded by g .

$f(n) = \Omega(g(n))$ can be denoted as $f \geq g$ since f is lower bounded by g .

$f(n) = \Theta(g(n))$ can be denoted as $f = g$ since f is tightly bounded by g .

$f(n) = o(g(n))$ can be denoted as $f < g$ since f is strictly upper bounded by g (referring to $0 \leq f(n) < c^*g(n)$ in the definition of o)

$f(n) = \omega(g(n))$ can be denoted as $f > g$ since f is strictly lower bounded by g (referring to $0 \leq c^*g(n) < f(n)$ in the definition of ω)

if $f(n) = O(g(n))$ AND $g(n) = \Omega(h(n))$
then $f(n) = O(h(n))$

If $f = g$ & $g = h$ then $f = h$

$\checkmark [O, \Theta, \Omega, o, \omega]$

Transitive
basic algebra

$f > g \& g > h$
 $f > h$

APPLIED COURSE

Timestamp: 8:35

All the notations $O, \Theta, \Omega, o, \omega$ hold transitive property as shown in the figure.

Reflexive

$$\left\{ \begin{array}{l} f(n) = \Theta(f(n)) \rightarrow f = f \checkmark \\ f(n) = O(f(n)) \rightarrow f \leq f \checkmark \\ f(n) = \Omega(f(n)) \rightarrow f \geq f \checkmark \end{array} \right.$$

* $\left\{ \begin{array}{l} f(n) \neq o(f(n)) \rightarrow f \not\sim f X \\ f(n) \neq \omega(f(n)) \rightarrow f \not\succ f X \end{array} \right.$

Timestamp: 10:55

Reflexive property only holds for O, Θ, Ω but doesn't hold for o, ω as shown in the figure.

Symmetry

$$\left\{ \begin{array}{l} f(n) = \Theta(g(n)) \text{ iff } g(n) = \Theta(f(n)) \\ f = g \text{ iff } g = f \checkmark \end{array} \right.$$

* $\left\{ \begin{array}{l} O \rightarrow f \leq g \text{ iff } g \leq f X \\ \Omega \rightarrow f \geq g \text{ iff } g \geq f X \\ o \quad > \quad < \quad X \\ \omega \quad < \quad > \quad X \end{array} \right.$

Timestamp: 12:57

Symmetric property only holds for Θ as shown in the figure.

Transpose Symmetry

$$f(n) = O(g(n)) \text{ iff } g(n) = \Omega(f(n))$$
$$\underline{f \leq g} \quad \text{iff} \quad \underline{g \geq f}$$

define ω →

$$f(n) = o(g(n)) \text{ iff } g(n) = \omega(f(n))$$
$$\underline{\underline{f < g}} \quad \text{iff} \quad \underline{\underline{g > f}}$$


Timestamp: 15:42

Transpose symmetry property holds for the pairs of (O, Ω) and (o, ω) as shown in the figure.

Trichotomy
 a, b
 $a > b ; a \leq b ; a = b$ → one of these three

$O, \Theta, \Omega, o, \omega$ does not hold

$f(n) = o(g(n))$
 $f(n) = \omega(g(n))$
 $f(n) = \Theta(g(n))$

$n = f(n)$
 $n^{(1+\sin(n))} = g(n)$

$\sin(n)^{+1}$ 



APPLIED COURSE

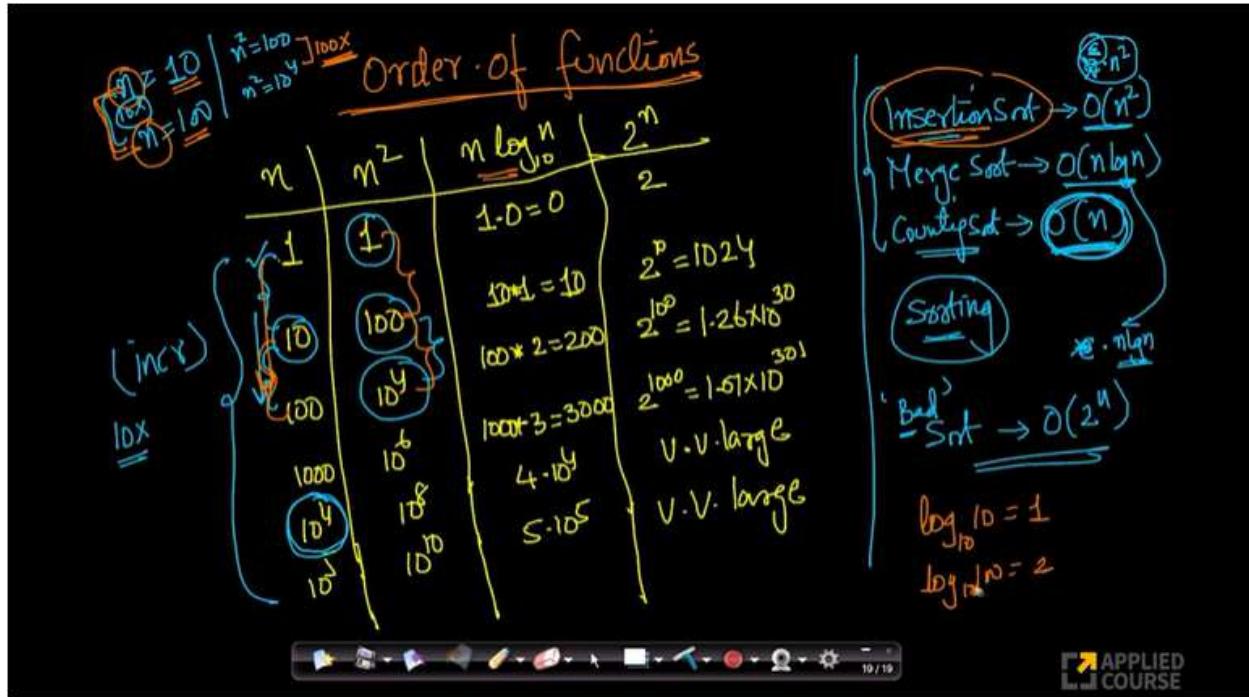
Timestamp: 18:22

Trichotomy is the property by which we can say for any two numbers a, b either of $a > b$, $a < b$ and $a = b$ must be true.

This property does not hold for our notations $O, \Theta, \Omega, o, \omega$.

An example to prove this is $f(n) = n$ and $g(n) = n^{(1+\sin(n))}$, notice that $g(n)$ takes values n^0, n^1, n^2 hence cannot be expressed as $g < f$ or $g > f$ or $g = f$.

9.9 Order of common functions & real world applications



Timestamp: 5:45

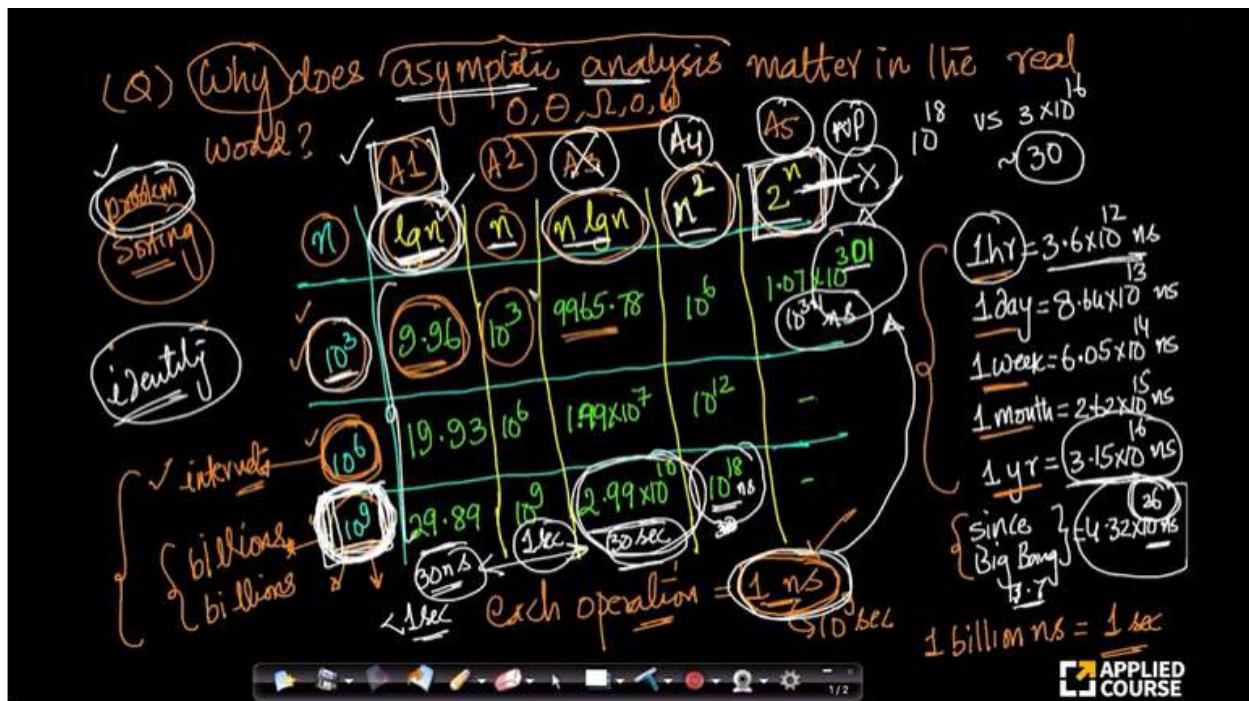
Notice that as in the above figure, as n increases, 2^n increases much more than n^2 which increases more than $n \log n$ which increases more than n .

Please refer the `order_of_common_functions` section in the below link that shows different time complexities in increasing order.

Comparing these computation complexities of the algorithms helps us choose the right algorithm that can save both run-time (time complexity) and memory requirement (space-complexity) for solving a problem.

https://en.wikipedia.org/wiki/Big_O_notation#Orders_of_common_functions

9.10 Why does asymptotic analysis matter in real world?



Timestamp: 15:24

Often in real world where some countries like India and China have populations of billions, the value of n might be around 10^9 , in such cases while algorithms like $\log n$ and n can take from a few seconds to days some algorithms like 2^n can take time very much longer than the time since universe existed.

Hence making an extra effort to come up with algorithms of lesser time complexity can vastly save our time. Hence they are very important in real-world scenarios.

10.1 Solved Problem: Polynomials

Q) $f(n) = n^2 + n + 1$
 $f(n) = O(g(n))$
 $f(n) = \Omega(h(n))$
 $f(n) = \Theta(k(n))$

what are the valid functions for $g(n)$, $h(n)$ & $k(n)$?

Timestamp: 1:32

Let us say $g(n) = n^2$.

So in order for $f(n) = O(g(n))$, there should exist constants n_0, c such that $0 \leq f(n) \leq c \cdot g(n)$ for all $n > n_0$.

We know that $n^2 + n + 1 \leq n^2 + n^2 + n^2$ (since $n \leq n^2$ and $1 \leq n^2$ for $n \geq 1$)

Hence $n^2 + n + 1 \leq 3n^2$. Hence we are able to find $n_0 = 1$ and $c = 3$ that satisfies the above condition. Hence $f(n) = n^2 + n + 1$ is $O(n^2)$.

Similarly we can prove that $f(n) = n^2 + n + 1$ is also $O(n^3)$.

Hence possible values of $g(n)$ are $n^2, n^3, n^4, \text{etc.}$

Similarly we can find $h(n)$ and $k(n)$ that satisfy $f(n) = \Omega(h(n))$ and $f(n) = \Theta(k(n))$ respectively. But in case of Ω , we have to find constants c and n_0 such that $0 \leq c \cdot h(n) \leq f(n)$ for all $n > n_0$. In case of Θ , it should satisfy both O and Ω .

generalization

$$f(n) = a_0 + a_1 n^1 + a_2 n^2 + a_3 n^3 + \dots + a_m n^m$$

$a_m > 0$

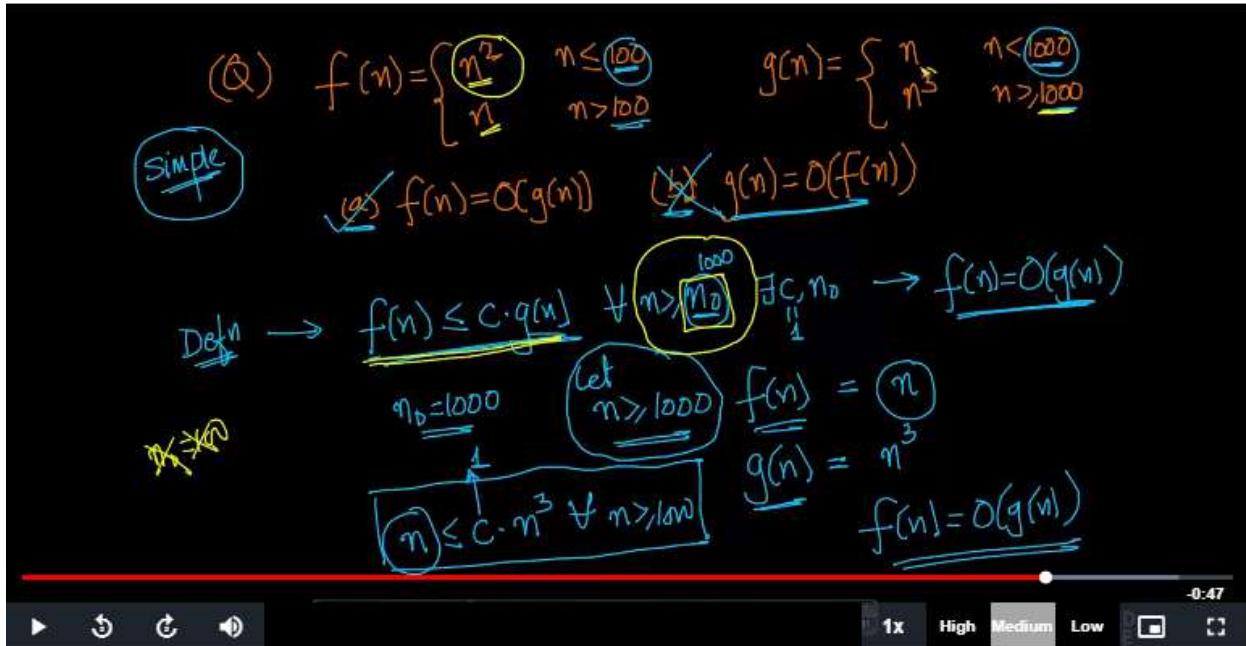
$$\begin{cases} f(n) = O(n^m) ; O(n^{m+1}) ; O(n^{m+k}) & k > D \\ f(n) = \Omega(n^m) ; \Omega(n^{m-1}) ; \Omega(n^{m-k}) & k > D \\ f(n) = \Theta(n^m) \end{cases}$$

Shootout

Timestamp: 16:00

Please refer to the generalization solutions for O , Ω and Θ when we were given function $f(n)$ of the above form.

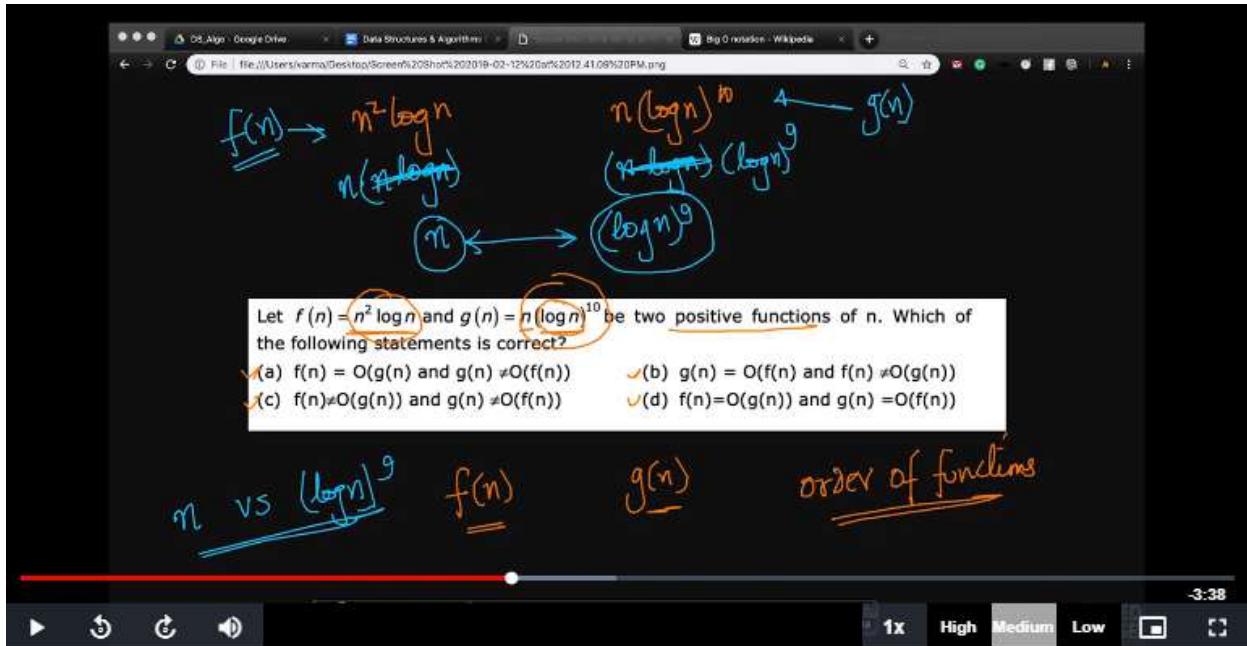
10.2 Solved Problem: $n > n_0$ case



Timestamp: 4:18

As shown in the figure, according to definition of O , $f(n)=O(g(n))$ if and only if $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. The equation should satisfy for all $n \geq n_0$. So as in the above case $f(n) \leq g(n)$ for all $n \geq n_0$ when $c=1$ and $n_0=1000$. This fails to hold for $g(n) \geq f(n)$ when $n \geq 1000$ hence that is not true.

10.3 Solved Problem-1



Timestamp 2:26

As shown in the above figure, to compare $f(n)$ and $g(n)$

$$f(n) = n^2 \log n$$

$$\rightarrow f(n) = n \log n * n$$

$$\rightarrow n \text{ and } (\log n)^9 \quad (\log n \text{ cancel on both sides})$$

\rightarrow from the reference table we know that $n > (\log n)^9$

Hence $f(n) > g(n)$. Hence $g(n) = O(f(n))$ and $f(n) \neq O(g(n))$

10.4 Solved Problem-2

(Q) $f_1(n) = 2^n$; $f_2(n) = n^{3/2}$; $f_3(n) = n \lg_2 n$; $f_4 = n^{\log n}$

what is the increasing order of asymptotic complexity?

order of functions

$n \lg n$	$n^{3/2}$
$n \cdot n^{1/2}$	
$\lg n$	n^b

8:14

Timestamp: 1:13

As shown in the above figure when to compare

$$f_2(n) = n^{3/2} \text{ and } f_3(n) = n \lg n$$

$$\rightarrow n \cdot n^{1/2} \text{ and } n \cdot \lg n$$

$$\rightarrow n^{1/2} \text{ and } \lg n$$

\rightarrow from the reference

https://en.wikipedia.org/wiki/Big_O_notation#Orders_of_common_functions

we know $n^{1/2} >= \lg n$

Hence $f_2(n) > f_3(n)$

Similarly, coming to f_4 and f_2 .

$$f_4(n) = n^{\log n} \text{ and } f_2(n) = n^{3/2}$$

We know that $n^a > n^b$ if $a > b$

Hence we compare powers

$\rightarrow \log n$ and $3/2$

We know that since $3/2$ is a constant it doesn't increase with n but $\log n$ increases

Hence $\log n > 3/2 \rightarrow f_4(n) > f_2(n)$.

Until now we got $f_4(n) > f_2(n) > f_3(n)$. Now let us compare $f_4(n)$ and $f_1(n)$.

$$f_4(n) = n^{\log n}$$
 and $f_1(n) = 2^n$

Taking log on both of them since we know $\log a > \log b$ if $a > b$

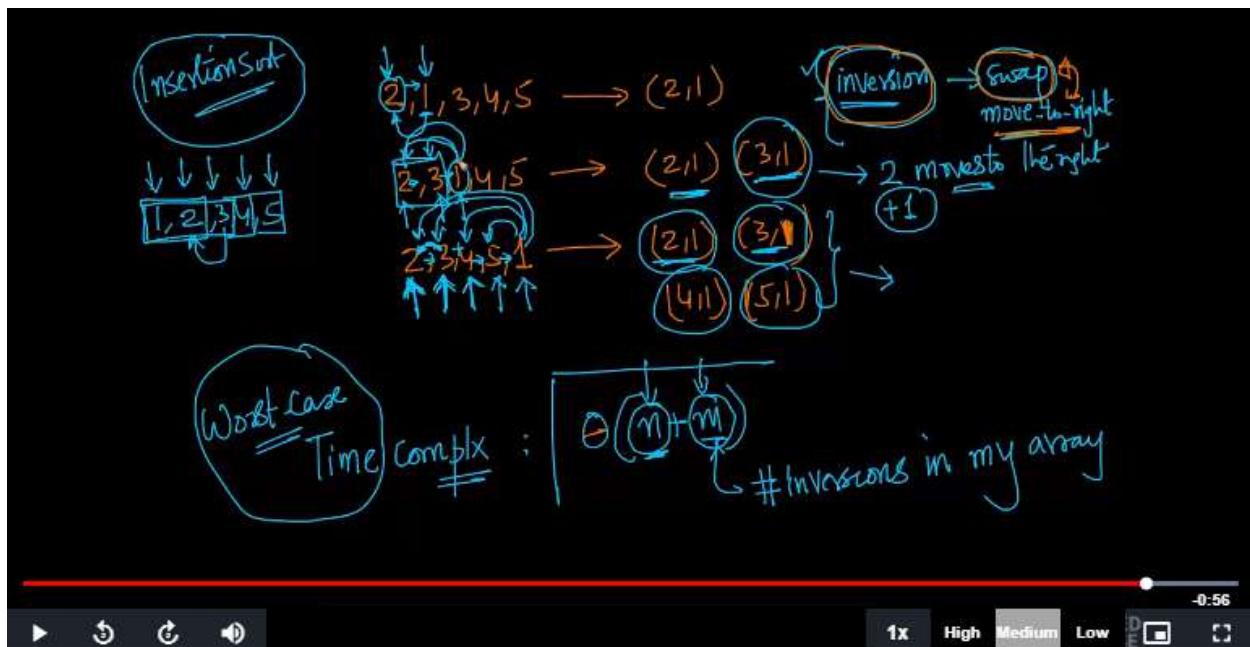
$\rightarrow \log n * \log n$ and n

$\rightarrow (\log n)^2$ and n

Again from our reference we know that $n > (\log n)^2$. Hence $f_1(n) > f_4(n)$.

So the required order is $f_1(n) > f_4(n) > f_2(n) > f_3(n)$

10.5 Solved Problem-3



Timestamp: 11:18

For any pair of indices i, j in an array A if $i < j$ and $A[i] > A[j]$, then it is called an inversion.

In insertion sort as shown in the above figure the number of inversions= no of swaps that need to be made

Hence for insertion sort the time complexity can also be thought of as $\Theta(n+m)$ where m is the number of inversions and n is the size of the input array.

In a permutation a_1, \dots, a_n , of n distinct integers, an inversion is a pair (a_i, a_j) such that $i < j$ and $a_i > a_j$.

What would be the worst case time complexity of the Insertion Sort algorithm, if the inputs are restricted to permutations of $1, \dots, n$ with at most n inversions?

A. $\Theta(n^2)$
 B. $\Theta(n \log n)$
 C. $\Theta(n^{1.5})$
 D. $\Theta(n)$

$\Theta(n+m) \quad m=n$
 $\Theta(n+n) = \Theta(2n) = \underline{\underline{\Theta(n)}}$



Timestamp: 10:58

Since the number of inversions given is atmost n , then the time complexity becomes
 $\Theta(n+n)=\Theta(2n)$

10.6 Solved Problem-4

1. $(n+k)^m = O(n^m)$ ✓ k & m are constants

2. $2^{n+1} = O(2^n)$ which of them are true

3. $2^{2n+1} = O(2^n)$

① $(n+k)^m = n^m + \binom{m}{1}n^{m-1}k + \binom{m}{2}n^{m-2}k^2 + \dots + k^m$

$O(n^m)$ $\underline{\underline{\Theta(n^m)}}$ $\underline{\underline{O(2^n)}}$



Timestamp: 2:39

For problem 1 , we know that a polynomial as shown in the above figure has a time complexity of $O(n^m)$.

For problem 2, for 2^{n+1} to be $O(2^n)$, $2^{n+1} \leq c \cdot 2^n$ for some c , since 2^{n+1} can be written as $2 \cdot 2^n$, the equation satisfies for constant $c=2$. Hence $2^{n+1}=O(2^n)$.

For problem 3,

Let us assume $2^{2n+1}=O(2^n)$ then the below equation should hold true

$$2^{2n+1} \leq c \cdot 2^n$$

$$\rightarrow 2 \cdot 2^{2n} \leq c \cdot 2^n$$

$$\rightarrow 2^{2n} \leq 2^n \text{ (since constants doesn't matter)}$$

$$\rightarrow 2n \leq n \text{ (since } \log a \leq \log b \text{ iff } a \leq b\text{)}$$

Since $2n \leq n$ is never true for $n \geq 1$, our assumption about $2^{2n+1}=O(2^n)$ cannot be true, hence the answer for problem3 is false.

10.7 Solved Problem-5

Consider the following functions:

- $f(n) = 2^n$
- $g(n) = n!$
- $h(n) = n^{\log n}$

Which of the following statements about the asymptotic behavior of $f(n)$, $g(n)$ and $h(n)$ is true?

- A. $f(n) = O(g(n))$; $g(n) = O(h(n))$
- B. $f(n) = \Omega(g(n))$; $g(n) = O(h(n))$
- C. $g(n) = O(f(n))$; $h(n) = O(f(n))$
- D. $h(n) = O(f(n))$; $g(n) = \Omega(f(n))$

Timestamp: 4:30

We know from the previous proofs and our reference table $n^{\log n} < 2^n < n!$.

$n^{\log n} < 2^n$ can be proved using $n^{\log n} < 2^n \rightarrow \log n < n$.

$2^n < n!$ can be checked by running through few values of n.

Hence in the question above

- a) $f(n)=O(g(n))$ is true and $g(n)=O(h(n))$ is false.
- b) $f(n)=\Omega(g(n))$ is false and $g(n) = O(h(n))$ is false.
- c) $g(n) = O(f(n))$ is false and $h(n) = O(f(n))$ is true.
- d) $h(n)=O(f(n))$ is true and $g(n) = \Omega(f(n))$ is true.

Hence the answer is d.

10.8 Solved Problem-6

(Q) `int UNKNOWN(int n)`

{ `int i, j, K = 0;`

`for (i = n/2; i <= n; i++)`

 { `for (j = 2; j <= n; j = j + 2)`

 { `K = K + n/2;`

 }

 }

 }

return (K);

What's the return value of this func?

(a) $\Theta(n^2)$

(b) $\Theta(n^2 \log n)$

(c) $\Theta(n^3)$

(d) $\Theta(n^2 \log^2 n)$

Timestamp: 4:41

The outer for loop with iterator i runs for $n-n/2 = n/2$ times. While the inner loops runs from $j=2$ to n with increments of 2 times so j takes values 2,4,8...n , so j takes $\log n$ values hence inner loop repeats for $\log n$ times, each time k is incremented by $n/2$.

Hence at the end the value of $K=n/2*\log n*n/2$ which is $\Theta(n^2 \log n)$.

10.9 Solved Problem-7

Consider the following C-program fragment in which i , j and n are integer variables.

```
for( i = n, j = 0; i > 0; i /= 2, j += i );
```

Let $\text{val}(j)$ denote the value stored in the variable j after termination of the for loop. Which one of the following is true?

- A. $\text{val}(j) = \Theta(\log n)$
- B. $\text{val}(j) = \Theta(\sqrt{n})$
- C. $\text{val}(j) = \Theta(n)$
- D. $\text{val}(j) = \Theta(n \log n)$

$i = i/2$
 $j = j + i$

$\text{Val}(j) = n \left\{ \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots + \frac{1}{n} \right\}$

Timestamp: 4:10

As shown in the above figure, when $i=n$ $j=0$ when $i=n/2$ $j=0+n/2$ when $i=n/4$ $j=0+n/2+n/4$ similarly when $i=n$ $j=0+n/2+n/4+\dots+1$.

$$\rightarrow j = n(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots) = n(1 - 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots)$$

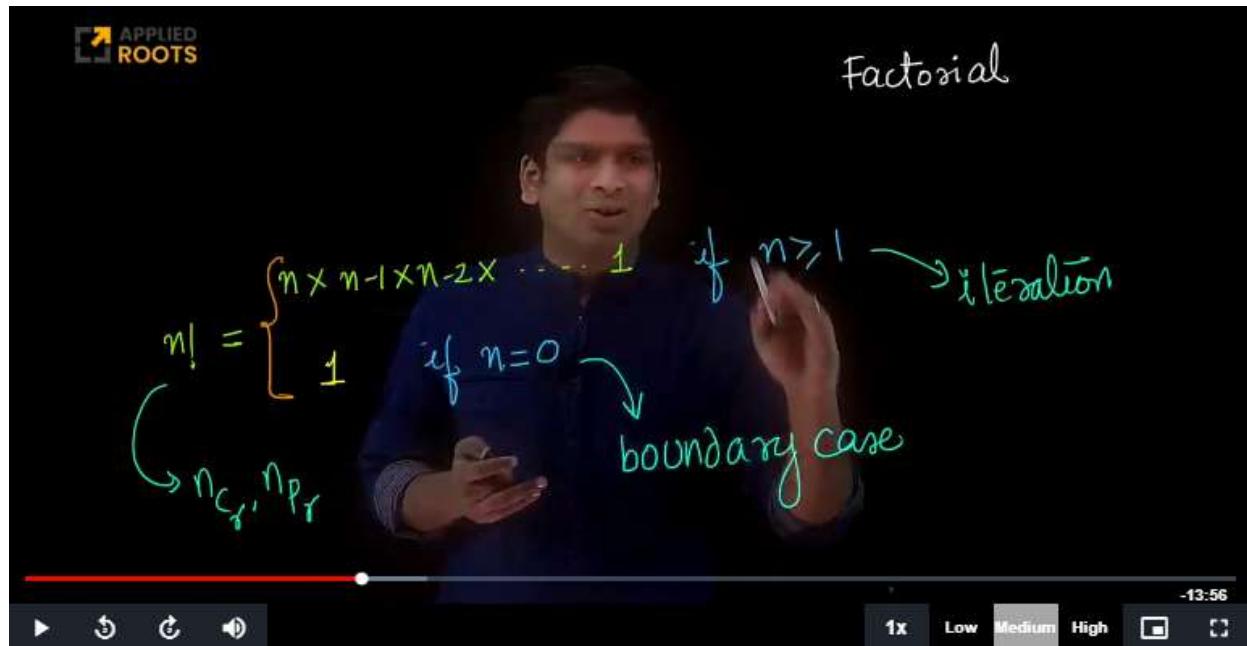
$\rightarrow j \leq n(2-1)$ (due to geometric series $1 + \frac{1}{2} + \frac{1}{4} + \dots = 1/(1-\frac{1}{2})$ but since our series is finite, we are using \leq)

Hence $\text{val}(j) = \Theta(n)$

11.1 Introduction to Recursion

In this video we will see how we can convert factorial algorithm iterative implementation to recursive implementation.

Please look at the formulation of factorial of a number n using iterative approach:



Timestamp: 5:42

Factorial of a number n is implemented using iterative programming as below:

The video player interface includes a play button, volume control, and a progress bar at -13:04. Below the video are playback controls: 1x, Low, Medium, High, and a full-screen icon.

Timestamp: 6:14

Notice that factorial does not exist for negative number hence we are doing error handing by displaying “Error”.

Notice that $n=0$ is a boundary case hence we are handling it by returning 1 without any computation.

For $n \geq 1$, we are implementing an iterative algorithm that stores the result $1 * 2 * \dots * n$.

The factorial of a number n using recursive approach is as follows:

APPLIED ROOTS

Recursion

$$n! = \begin{cases} n \times (n-1)! & \text{if } n \geq 2 \rightarrow \text{recursive case} \\ 1 & \text{if } n=1 \text{ or } n=0 \end{cases}$$

boundary case
termination

7:51

Timestamp: 11:25

Notice that recursion is followed by breaking up a bigger problem into smaller subproblem(s) and using these smaller subproblems to solve the bigger problem.

As shown in the above figure, $n!$ can be computed by calculating $(n-1)!$ and then multiplying it by n . So the problem of calculating $n!$ has now become the problem of calculating $(n-1)!$ and just multiplying it by n .

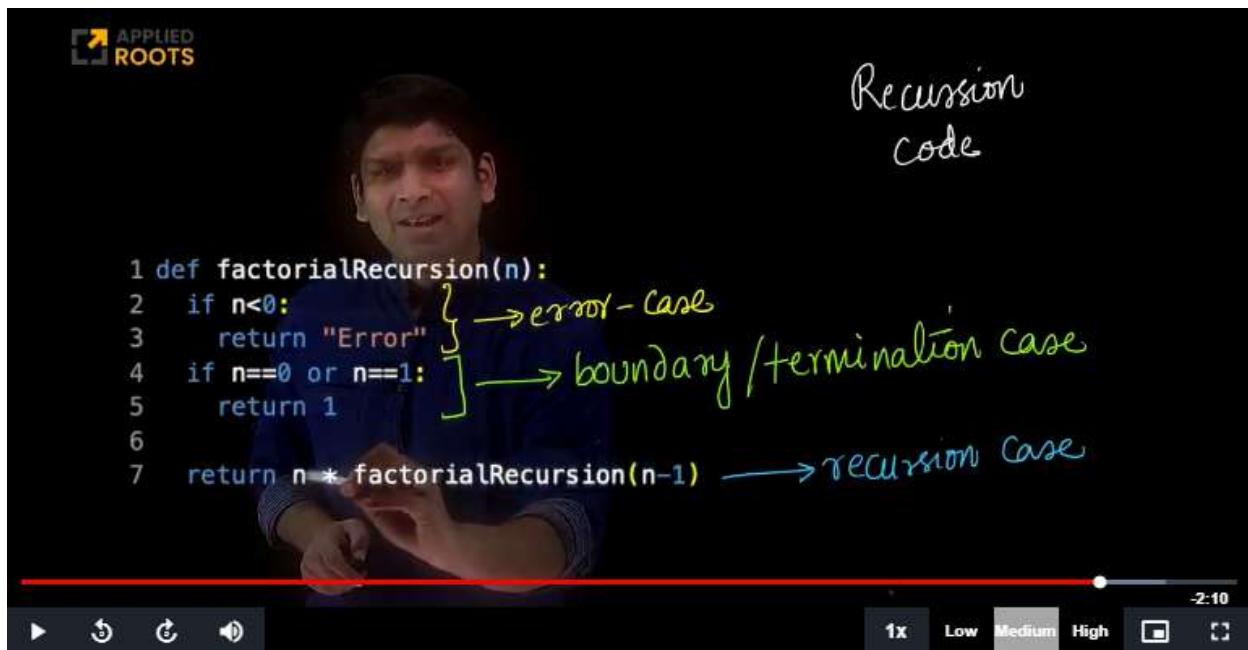
APPLIED ROOTS

$$n! = 5! \rightarrow 5 \times 4! \rightarrow 4 \times 3! \rightarrow 3 \times 2! \rightarrow 2 \times 1! \rightarrow 1 (\text{termination})$$

6:13

Timestamp: 13:05

Notice that in the above figure and in our formulation $n=0$ and $n=1$ are called termination or boundary cases since as shown in the above figure, we keep on splitting our bigger problem i.e computation of $5!$ Into smaller subproblem such as $4!$ and so on until we reach $1!$ And then we begin to trace back and solve our bigger problems hence $n=0$ and $n=1$ are called boundary or termination cases.

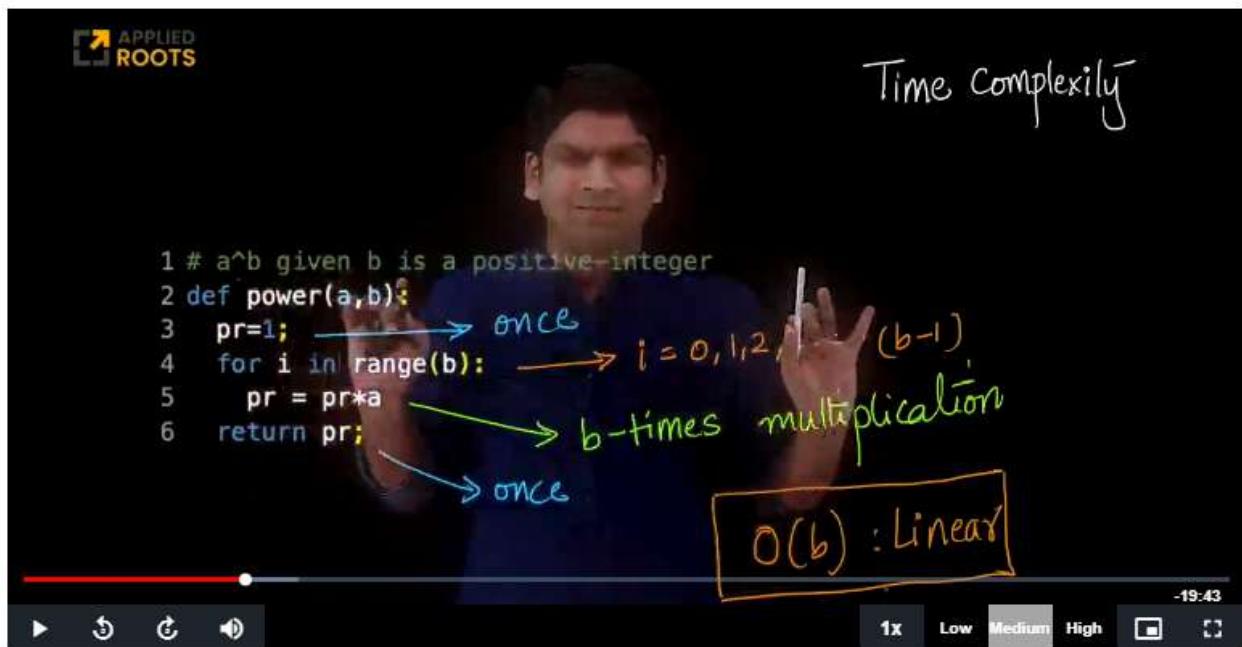


Timestamp: 17:06

In the above figure we can see how to implement factorial using recursion. Notice how we handled our error and boundary cases and how to defined our recursion case.

11.2 Power(a,b) using recursion

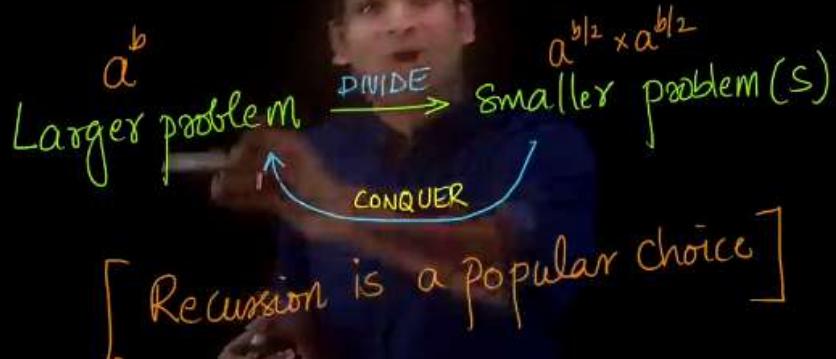
In this video, we look at how we can calculate a^b using recursion.



Timestamp: 4:27

In the above figure we can see how can implement b power of a number a (a^b) using iterative algorithm in $O(b)$ time complexity.

Divide & Conquer



-15:08

1x Low Medium High

Timestamp: 9:02

Divide and Conquer is a popular strategy using which we can divide a larger problem into smaller subproblems (divide) and combining them in some form to solve the larger problem(conquer). For implementing this strategy in code, recursion is a popular choice.

Recursive power (a^b)

$$a^b = \begin{cases} a^{b/2} \times a^{b/2} & \text{if } b \text{ is even} \\ a^{\lfloor b/2 \rfloor} \times a^{\lfloor b/2 \rfloor} \times a & \text{if } b \text{ is odd} \\ a & \text{if } b=1 \end{cases}$$

written using examples

-7:09

1x Low Medium High

Timestamp: 17:02

a^b can be solved using recursion as shown above. Notice that $\lfloor x \rfloor$ for a number indicates floor of x meaning an integer less than x and closer to x . For ex: $\lfloor 2.5 \rfloor$ is 2.

Please try for some example values of b , to check it works. Notice that we are solving the bigger problem of calculating a^b by dividing into smaller subproblems of calculating $a^{b/2}$ and $a^{b/2}$ (incase if b is even, it holds similarly when b is odd). Using this strategy once you compute $a^{b/2}$, you need not calculate the second $a^{b/2}$ but can use it directly many times once computed. To understand more let us look at the code implementation.

Applied Roots

Recursive a^b

```
1 # a^b given b is a positive-integer
2 import math
3
4 def powerRec(a,b):
5     if b==1:           → Case 3
6         return a
7     if b%2 == 0: # b is even   → Case 1
8         tmp = powerRec(a,b/2)
9         return tmp * tmp
10    else: # b is odd        → Case 2
11        tmp = powerRec(a,math.floor(b/2))
12        return tmp*tmp*a
13
```

6:14

Timestamp: 17:57

Notice that once we computed $a^{b/2}$ we are simply storing it in tmp variable and using it to compute a^b thus saving performing many multiplications. Notice how we wrote recursion for different cases and how we handled boundary case.

```
[68] 1 import timeit  
2  
3 startTime = timeit.default_timer()  
4 power(20,10000)  
5 endTime = timeit.default_timer()  
6 print("The time difference is :", endTime - startTime)  
  
The time difference is : 0.013566157000241219
```



```
1 import timeit  
2  
3 startTime = timeit.default_timer()  
4 powerRec(20,10000)  
5 endTime = timeit.default_timer()  
6 print("The time difference is :", endTime - startTime)  
  
The time difference is : 0.00044880299901706167
```

Timestamp: 20:17

Notice that the time taken for solving the problem iteratively took 0.0135 time where as recursively it took 0.0004 time. This 30x time saving comes from our how we divided our problem into subproblems and using them instead of recomputing and recursion helped us implement it.

11.3 Asymptotic Analysis using Recursion Tree method

In this video we will perform the time complexity analysis of the above recursive solution to calculate a^b using recursion tree method.

APPLIED ROOTS

Recursive a^b

```

1 # a^b given b is a positive integer
2 import math
3
4 def powerRec(a,b):
5     if b==1:    → O(1)
6         return a
7     if b%2 == 0: # b is even   → O(1)
8         tmp = powerRec(a,b/2) → recursive { ? }
9         return tmp * tmp
10    else: # b is odd
11        tmp = powerRec(a,math.floor(b/2)) → recursive { ? }
12        return tmp*tmp*a
13

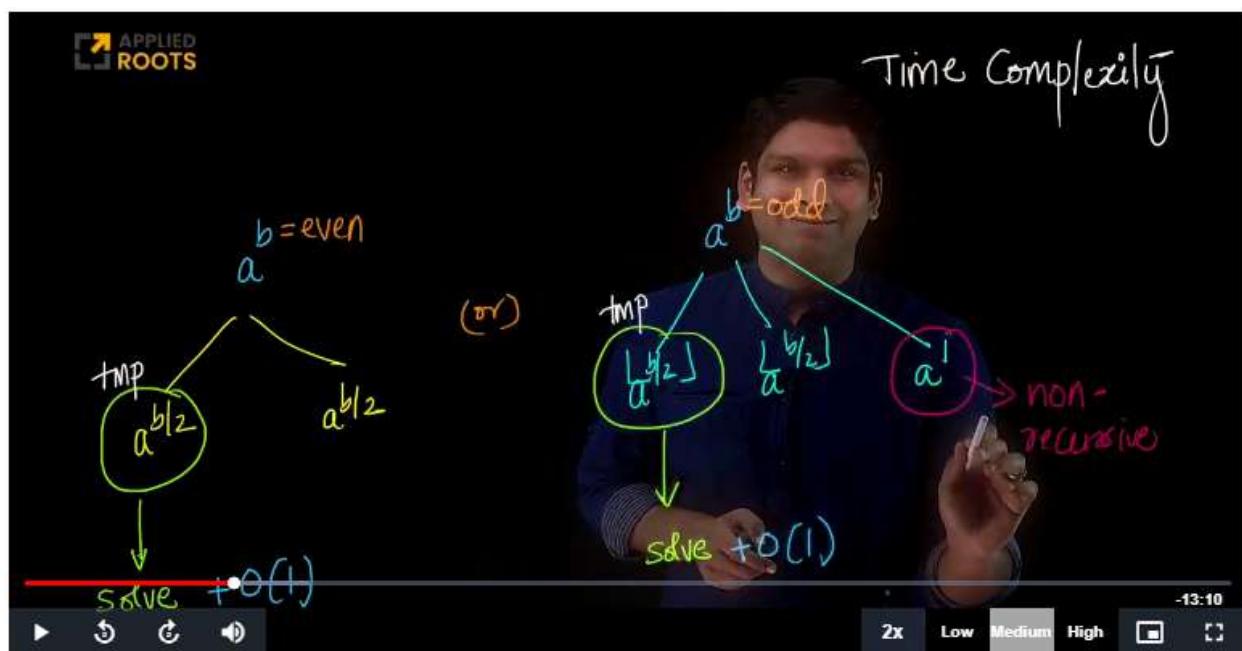
```

-15:25

▶ ⏴ ⏵ 🔍 🔊 2x Low Medium High

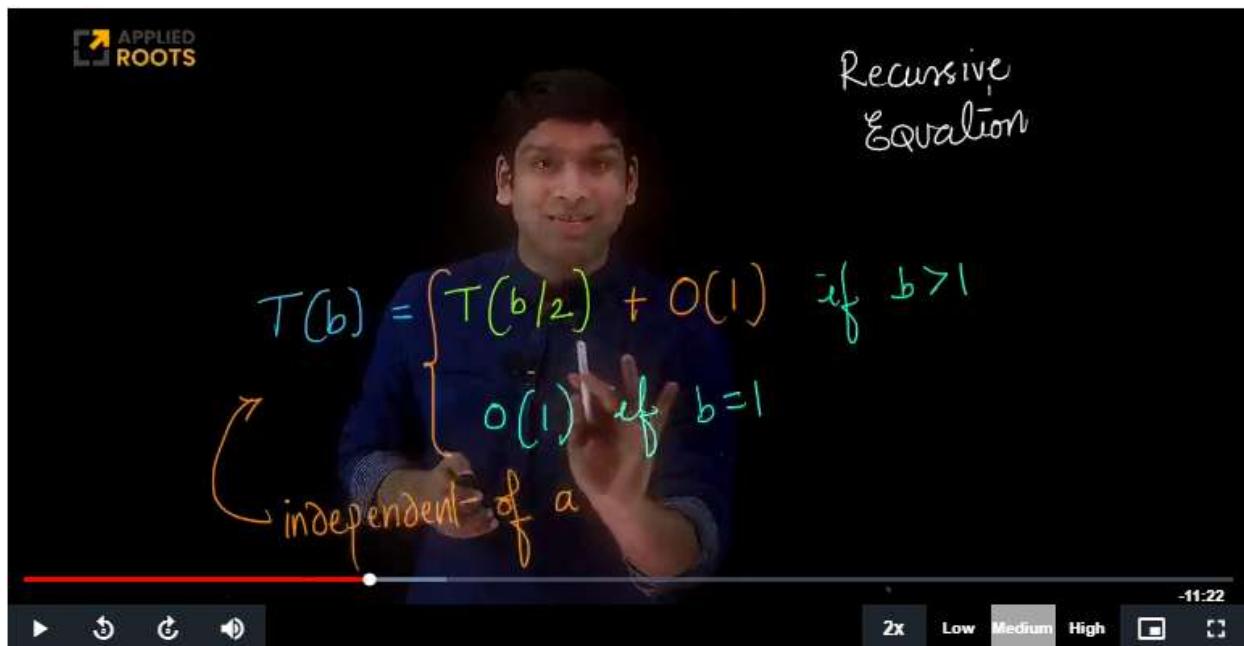
Timestamp: 0:24

Notice in the above figure, in the recursive implementation of a^b , the boundary case when $b=1$ takes $O(1)$ time and code in line 7,9,12 take $O(1)$ time each. Let us understand the time taken for recursion.



Timestamp: 2:46

Notice that in the above figure the amount of time taken for calculating a^b is the time required for solving $a^{b/2}$ and $O(1)$ time for calculating $\text{tmp} * \text{tmp}$ (the same can be extended when b is odd since there is only an extra multiplication with a which takes $O(1)$).



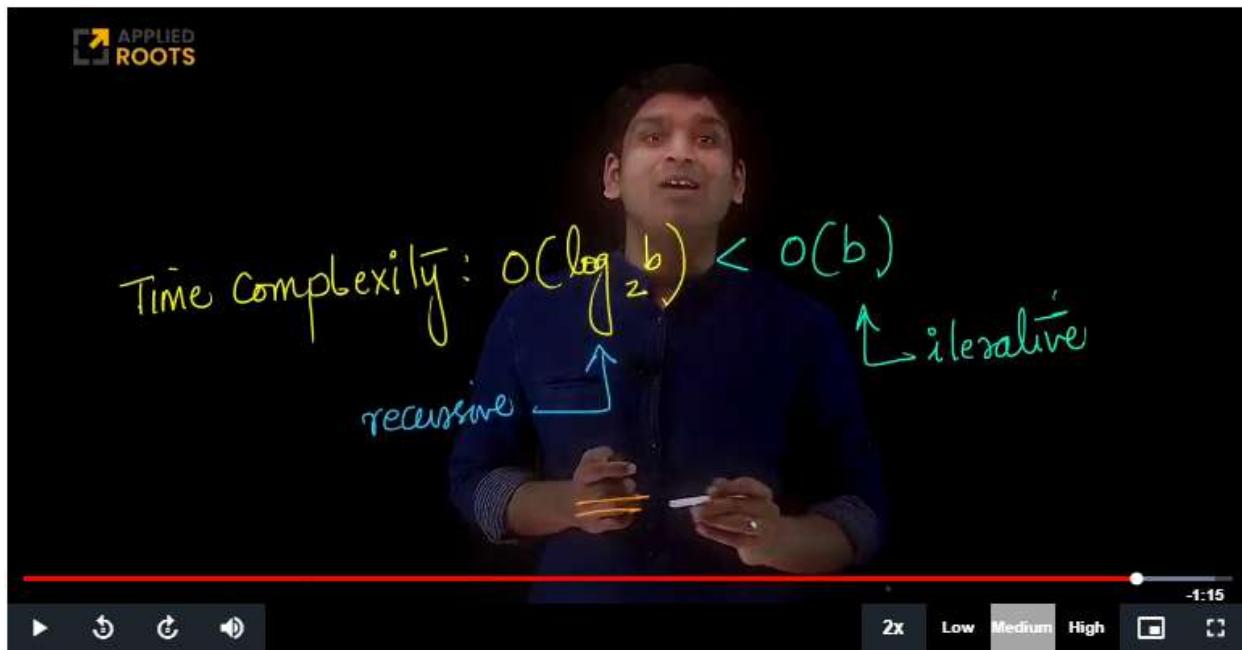
Since the time required to solve the problem is independent of a and depends only on b . It can be expressed as in the above figure.

Timestamp: 10:51

Notice that in the above tree representation of our recursive function, at each level we are taking only $O(1)$ time for that level only plus the time taken for the lower levels. The problem of solving a^b is divided into solving $a^{b/2}$ and multiplying with itself, the problem of solving $a^{b/2}$ is further divided into solving $a^{b/4}$ and multiplying with itself and so on.

This process is repeated for K times until we reach the boundary condition where $b/2^K=1$, solving which gives $K=\lg(b)$ (note: $\lg(b)$ is symbol for representing $\log_2 b$)

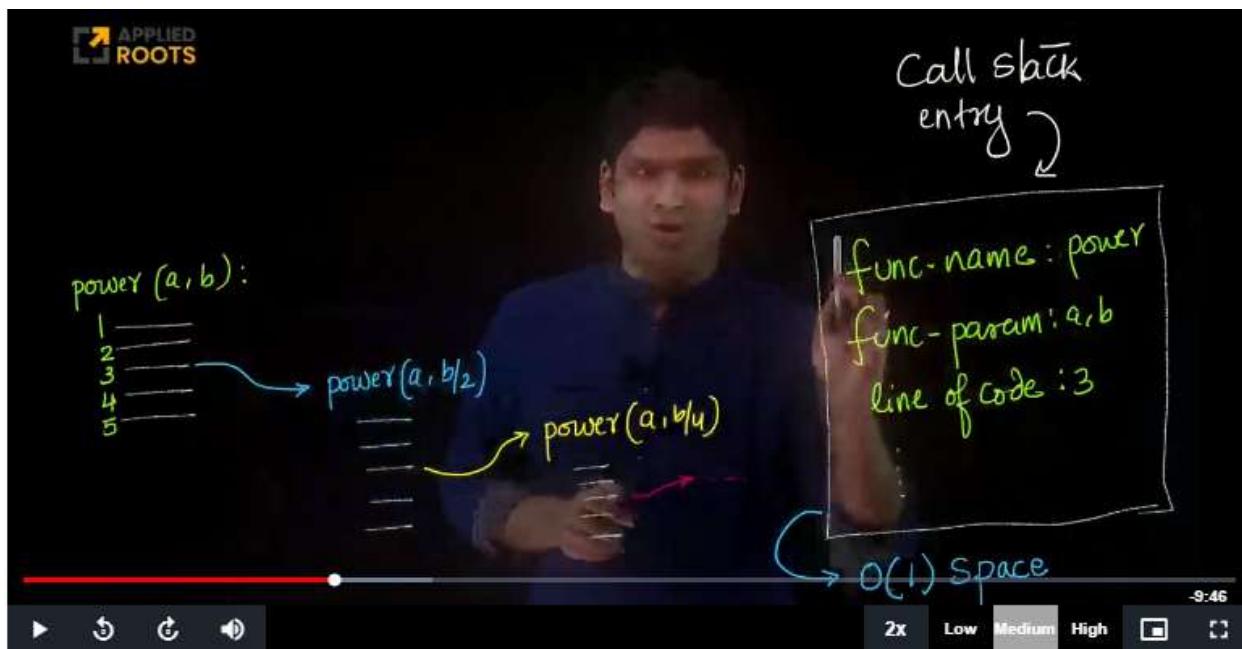
Hence this process is repeated for K times each taking $O(1)$ time and hence the time complexity of recursive implementation of a^b is $K \cdot O(1)$ which is $O(K)$ which is $O(\lg b)$



Timestamp: 14:40

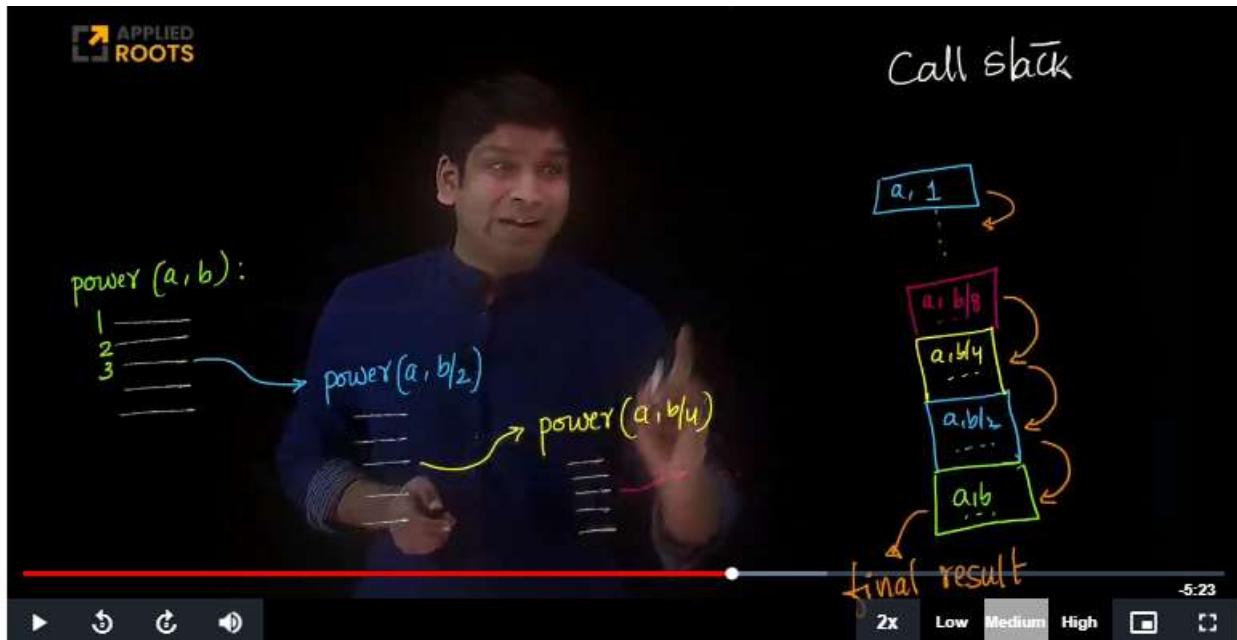
Notice from the above figure that the time complexity of our recursive implementation is less than the time complexity of iterative implementation thereby getting us our required speed up of 30x as in the previous video.

11.4 Call-Stack for Recursion



Timestamp: 3:23

Notice as in the above figure for each recursion of our program, a space called call stack entry is created which contains the function name, the function parameters, lines of code, etc which occupies $O(1)$ space.



Timestamp: 7:46

Notice that as in the above figure, for each recursion call a new call stack entry corresponding to the new recursion call is added on top of the previous call stack entry as a stack until the boundary condition is reached. Once the boundary condition is reached and when it comes out of that recursive call, the call stack entry corresponding to that recursive call is deleted until the final result is calculated.

```
1 # a^b given b is a positive-integer
2 import math
3
4 def powerRec(a,b):
5     if b==1:
6         return a
7     if b%2 == 0: # b is even
8         tmp = powerRec(a,b/2)
9         return tmp * tmp
10    else: # b is odd
11        tmp = powerRec(a,math.floor(b/2))
12        return tmp*tmp*a
13
```

Timestamp: 9:06

When we look at our recursive program it is easy to get confused that we are using $O(1)$ space since it looks like we are only creating an extra variables tmp but this is not the case. We have to account for each call stack entry because even though we are not explicitly writing code to create these call stack entries programming languages such as python implicitly create these entries.

Since in our previous video, we learned that our recursive program has $K=lg b$ recursive calls and since space complexity for each of them is $O(1)$. The total space complexity for our recursive program becomes $O(lg b)$. Remember that for our iterative algorithm the space complexity is just $O(1)$ since we are only creating extra space for storing i and pr .

Hence,

Time complexity of iterative : $O(b)$

Space complexity of iterative: $O(1)$

Time complexity of recursive: $O(lg b)$

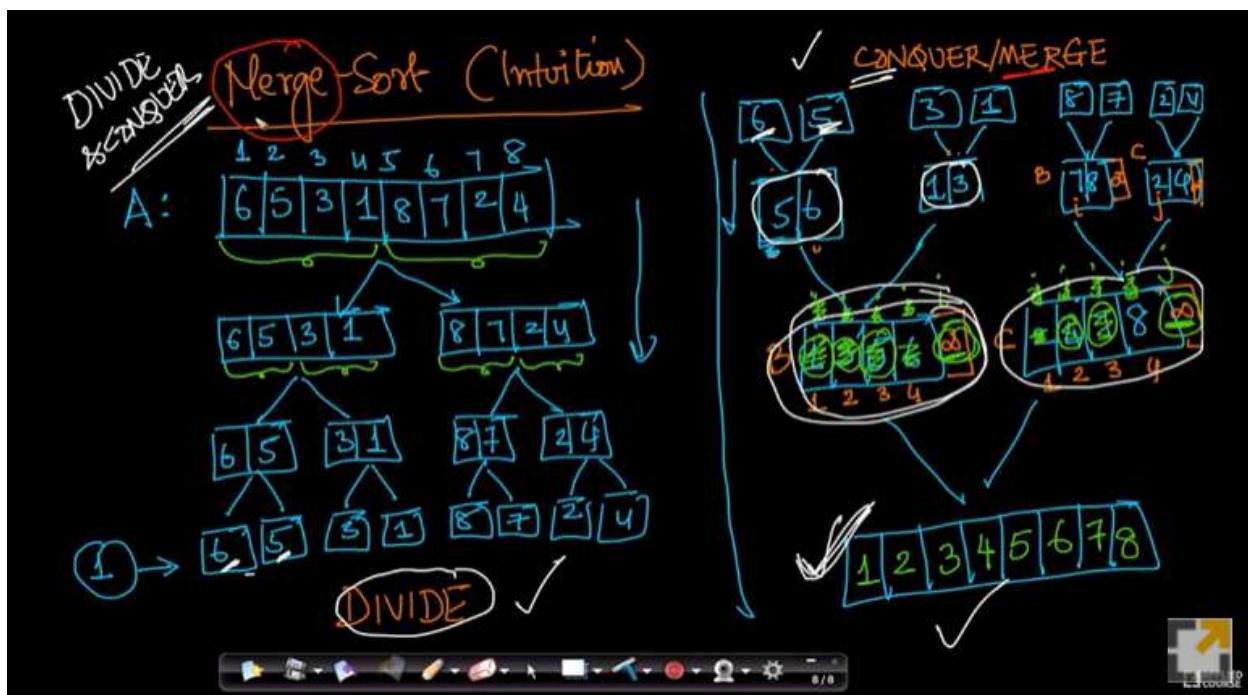
Space complexity of recursive: $O(lg b)$

Hence there is a trade-off between space and time complexity for iterative implementation and recursive implementation. The reason for choosing recursive implementation over the iterative implementation is since ram sizes are growing each day hence space complexity shouldn't be an issue but we want low latency application hence time complexity is much more important.

11.6 Merge-Sort - Why?

Having learnt insertion sort, we know that its worst case complexity is $O(n^2)$, if we can come up with any other sorting algorithm that can solve our sorting problem with a better time complexity it would save our time. Hence we learn merge sort that has $O(n \lg n)$ time complexity. Along with reducing time complexity it is also known to work with large sizes of data that cannot be stored in ram but are present in hard disk. Sorting these large sizes of data using insertion sort is not easy.

11.7 Merge-Sort - Intuition



Timestamp: 13:46

As shown in the above figure, merge sort follows divide and conquer approach. Every divide and conquer approach has two stages 'divide': the stage in which bigger problems are divided into smaller subproblems and 'conquer': the stage in which these smaller subproblems are solved and combined to get the final result.

In merge sort algorithm in divide stage the given array is broken continuously into smaller arrays of size nearly(when odd size) half until the size of each becomes 1. These smaller arrays are

then combined so that each combination is sorted, so as more and more pieces are combined the greater the size of the sorted array until the final result when the entire array is sorted.

Let us look at how these smaller arrays are combined. Let us say we have smaller arrays A=[5,6] and B=[1,3] let us say we store the result in array C.

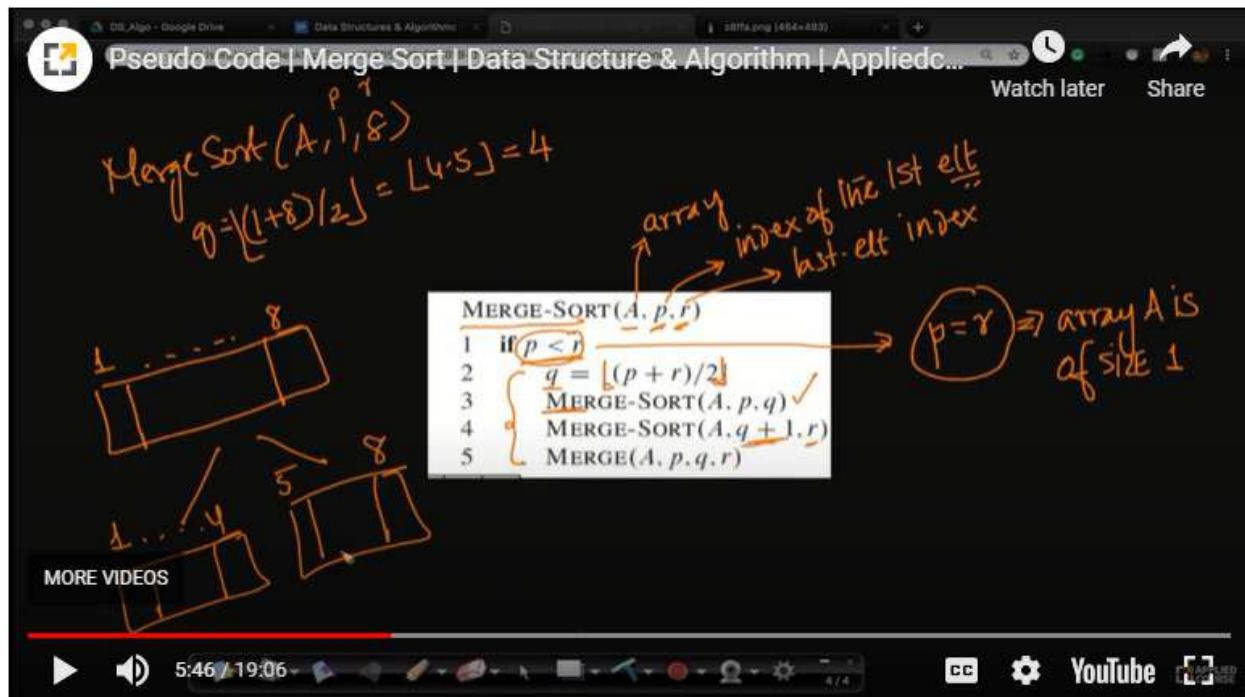
First we compare the first elements of A and B and the smaller one is stored in C.
So now A=[5,6] , B=[3] and C=[1].

Next we compare again the first elements of A and B and the smaller one is stored in C.
So now A=[5,6] , B=[] and C=[1,3].

Now since there are no elements in B all the elements in A are added to C.
So now C=[1,3,5,6]. This is done using markers i,j for arrays A,B to note the first element in each array.

Merge sort names comes from how we are merging these smaller arrays to former larger arrays.
Please refer to the merge sort gif in wikipedia to understand more.

11.8 Merge-Sort - Pseudo Code



Timestamp: 5:46

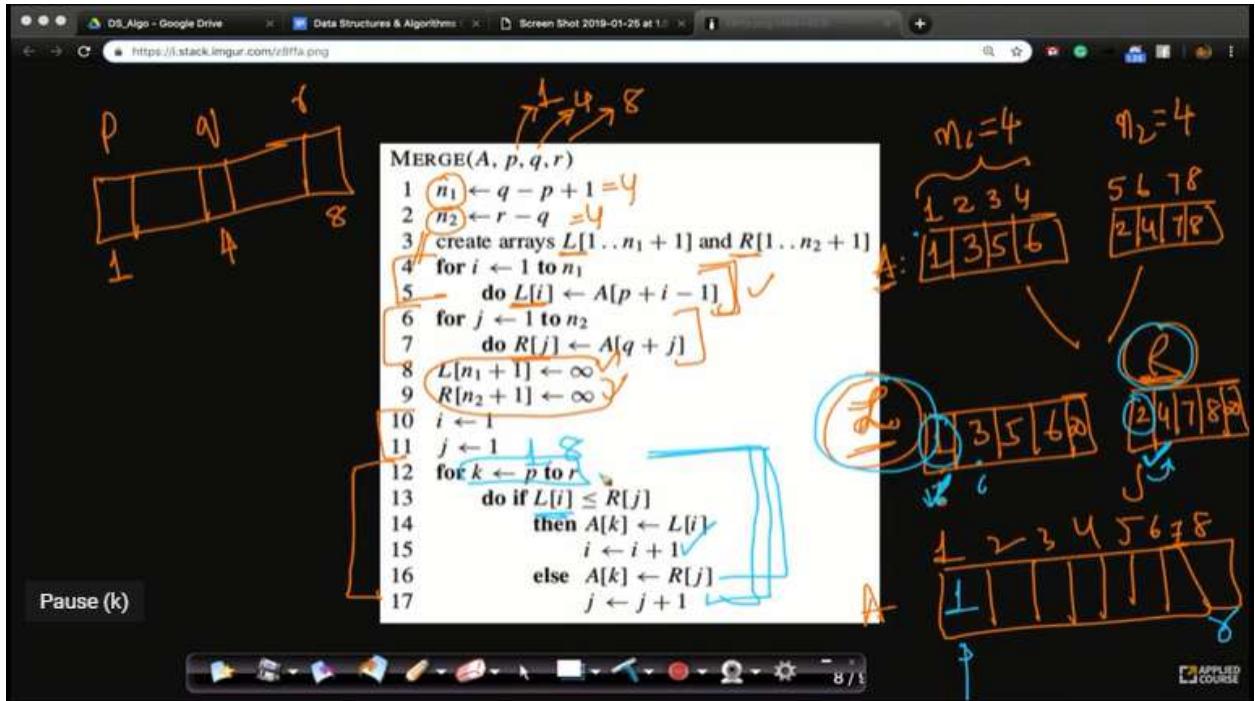
The pseudo code for merge sort looks as in the above figure.

A is the given array to be sorted, p is the index of the first element and r is the index of the last element.

So $p < r$ condition checks whether the size of the array is 1 (since if array contains only one element then $p=r$). If the size of array is 1 then it is already sorted otherwise we have to perform the following operations.

- i) merge sort on the first half of the array
- ii) merge sort on the second half of the array
- iii) merging both the halves.

The below figure shows the pseudo code for merge operation.



Timestamp: 16:42

N1,n2 store the size of each smaller array.

We are creating separate arrays for left half and right half using the elements of the main array so that we don't overwrite the main array

We are adding infinity at the end of both arrays so that when one of the array completes we still have to compare the firsts of both arrays and add the least one. Since infinity is greater than all numbers so once one array is completed we can continuously add elements from the other array.

We fill the original array starting from index p with the smaller of firsts of both arrays until we have filled index r.

The bigger picture is how merge sort is dividing bigger array to smaller arrays and then combining them in sorting order thus sorting the bigger array.

11.9 Merge-Sort - Analysing time and space complexity

The screenshot shows a presentation slide with a black background. At the top, there is a navigation bar with icons for back, forward, search, and other controls. The main content area contains handwritten mathematical notes and pseudocode.

Handwritten notes:

$$T(n) = 2 \cdot T(n/2) + T(\text{merge arrays of size } n/2)$$
$$T(n/2) = 2 \cdot T(n/4) + T(\text{merge arrays of size } n/4)$$

Pseudocode:

```
MERGE-SORT(A, p, r)
1   if p < r
2       q = ⌊(p + r)/2⌋
3       MERGE-SORT(A, p, q)
4       MERGE-SORT(A, q + 1, r)
5       MERGE(A, p, q, r)
```

At the bottom right, there is a small logo for "APPLIED COURSE".

Timestamp: 6:40

As shown in the figure, due to the recursive equation of merge sort, the time taken for sorting array of size n $T(n)$ is the sum of time taken to sort each of the smaller arrays $2 \cdot T(n/2)$ ($T(n/2)$ each due to size $n/2$) as well as the time taken for merging two arrays of size $n/2$ $T(\text{merge arrays of size } n/2)$, similarly it can be extended to arrays of size $n/4$ and so on.

The video player interface shows the following details:

- Title:** Analyzing time & space complexity | Merge Sort | Data Structure & Algorithms
- Annotations:**
 - Space complexity: $O(n)$ (circled), $n/2$, $n/2$, $C_1 \cdot n/2$, $C_2 \cdot n$.
 - Time complexity: $O(n)$ (circled), $C \cdot n/2$, $C \cdot n/2$, $C_1 \cdot n/2$, $C_2 \cdot n$.
 - Handwritten notes: n , $n/2$, $n/2$, n .
- Code:**

```

MERGE(A, p, q, r)
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  create arrays L[1.. $n_1 + 1$ ] and R[1.. $n_2 + 1$ ]
4  for  $i \leftarrow 1$  to  $n_1$ 
5    do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7    do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p$  to  $r$ 
13   do if  $L[i] \leq R[j]$ 
14     then  $A[k] \leftarrow L[i]$ 
15      $i \leftarrow i + 1$ 
16   else  $A[k] \leftarrow R[j]$ 
17      $j \leftarrow j + 1$ 

```
- Diagram:** A diagram illustrating the merging of two arrays, L and R, into array A. It shows array A partitioned into three sections: L, M (merging region), and R. The merging process involves copying elements from L and R into A, with indices i and j indicating the current elements being compared.
- Player controls:** Back, forward, volume, search, etc.
- Timestamp:** 12:28 / 2012
- YouTube logo:** YouTube

Timestamp: 12:28

From the above figure we can see that for merging two arrays of size $n/2$, we need some constant $c \cdot n/2$ time for creating array L, $c \cdot n/2$ time for creating array R and for running the loop some constant $c^2 \cdot n$ times to fill the original array A.

For space complexity the extra space that is created is for some variable such i,j which $O(1)$ and for creating arrays L, R each of size $n/2$. So the space complexity is $2 \cdot O(n/2) + O(1)$ which $O(n)$.

Hence the total time complexity for merge operation is $O(n)$ while the space complexity is also $O(n)$.



Analyzing time & space complexity | Merge Sort | Data Structure &...

Watch later Share

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + O(n)$$

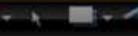
$$S(n) = C + O(n)$$



MORE VIDEOS



16:10 / 20:12



YouTube



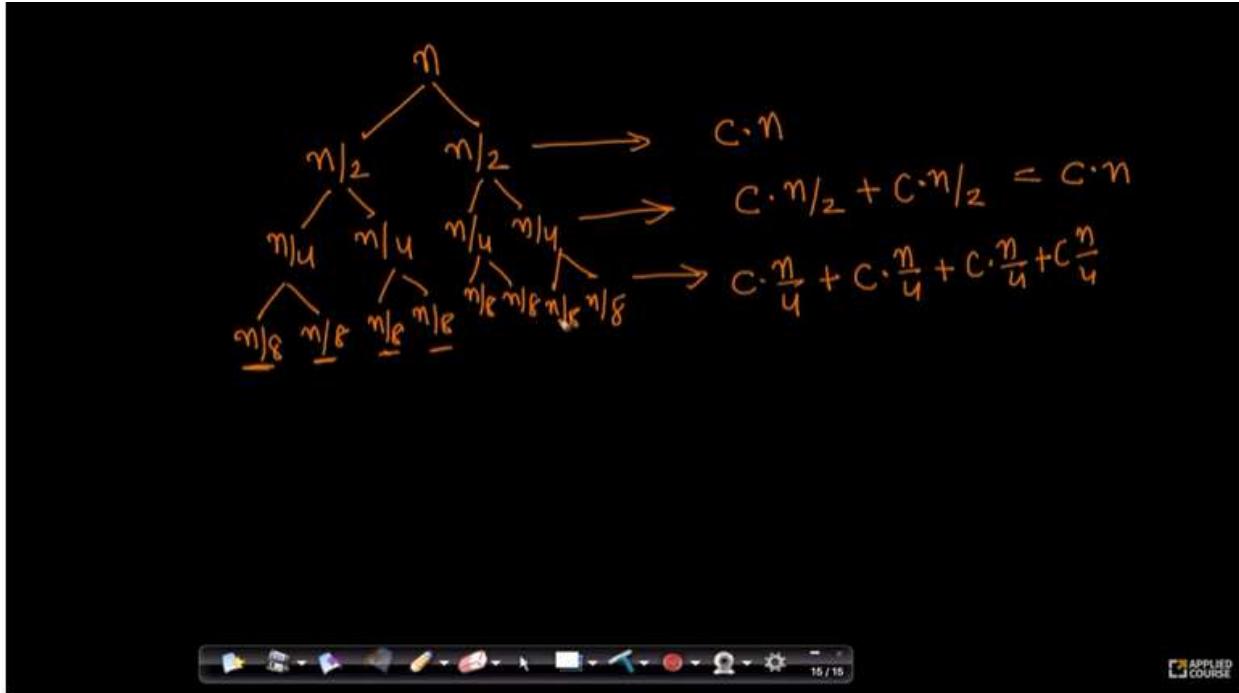
Timestamp: 16:10

So the total space complexity of merge sort is $O(1)+O(n)=O(n)$ ($O(1)$ for storing q in the pseudo code).

The total time complexity becomes

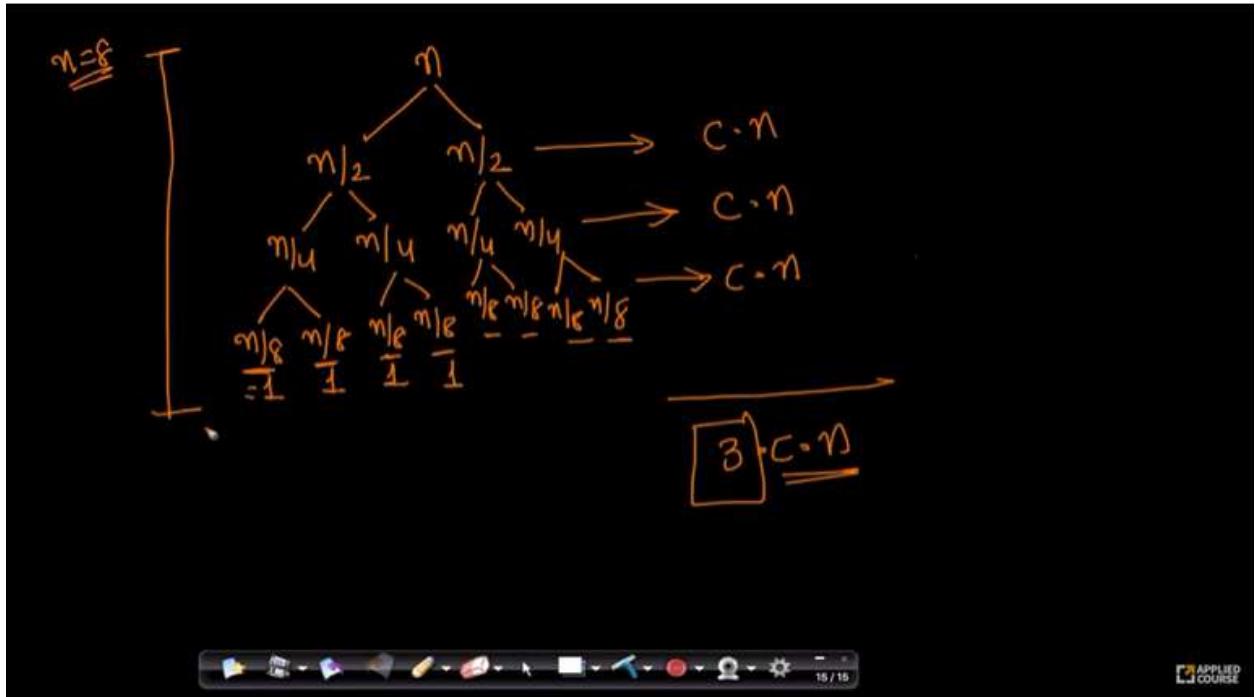
$$T(n)=2 \cdot T\left(\frac{n}{2}\right) + O(n)$$

11.10 Merge-Sort - Recursion Tree Method



Timestamp: 3:08

From the above figure we can see that at each stage for merge operation the time taken $c \cdot n$. For example at level 3 when merging 4 arrays of size $n/4$, we have to merge first two arrays in $c \cdot n/2$ time and the rest two arrays in $c \cdot n/2$ time. Hence the total time at that level = $c \cdot n/2 + c \cdot n/2 = c \cdot n$.



Timestamp: 4:47

We can see from the above figure that for 8 elements we have three levels, similarly if we can check for 16 elements we have 4 levels. If we can observe that if we have n elements then we have $\log_2 n$ levels. Hence the total time complexity for merge sort is $\lg n * O(n) = O(n \lg n)$ and space complexity is $O(n)$.

The diagram illustrates the time and space complexities of two sorting algorithms:

- Merge-Sort**:
 - Time complexity: $O(n \log_2 n)$
 - Space complexity: $O(n)$
- Insertion Sort**:
 - Time complexity:
 - Worst case: $O(n^2)$
 - Best case: $O(n)$
 - Space complexity: $O(1)$

Below the diagram, there is a timestamp of 11:00, a video control bar showing 11:00 / 12:28, and a YouTube interface with various icons.

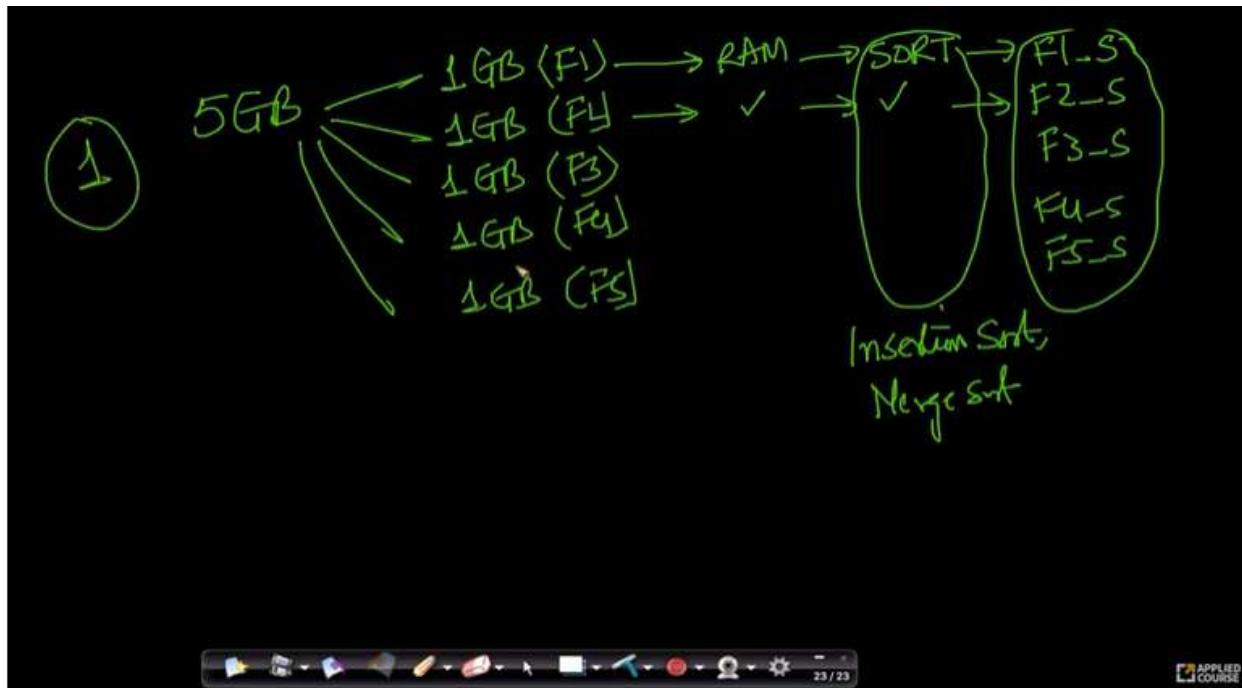
Timestamp: 11:00

Comparing insertion sort and merge sort, the best case time complexity of insertion sort is better than merge sort but the worst case time complexity of insertion sort is worse than that of merge sort.

The space complexity of insertion sort is $O(1)$ while the space complexity of merge sort is $O(n)$. We have to choose the appropriate sorting algorithms based on requirement.

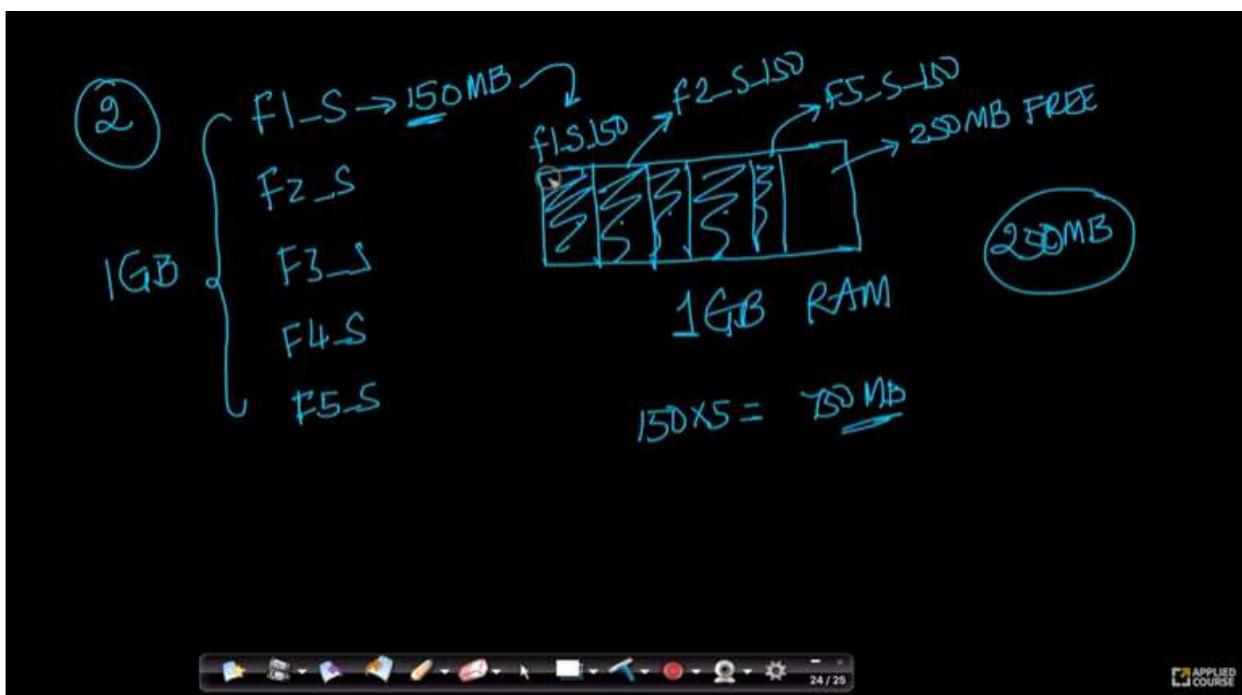
11.11 Merge-Sort - External Sorting

External Sorting is done when the entire data that needs to be sorted cannot be loaded in the ram. In this case we will use external storage such as hard disk hence the name External Sorting.



Timestamp: 7:50

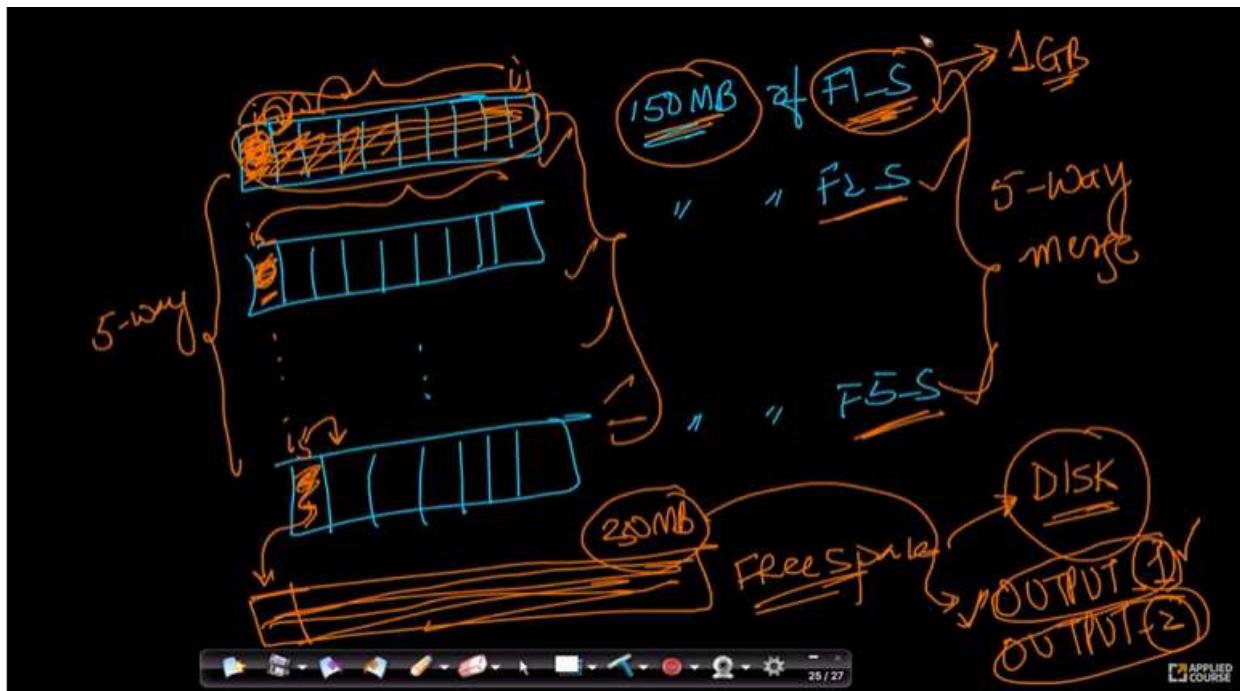
Step1) Suppose if you have a 5 gb file the first thing we have to do is to split the 5gb data into smaller arrays that can fit into the memory and apply any sorting algorithm (considering the space complexity) to sort them individually and store in files $F_{1_s}, F_{2_s}, F_{3_s}, F_{4_s}, F_{5_s}$.



Timestamp: 10:57

step2) Suppose we have 1gb of memory, then since we have already sorted parts of data and stored in files each of size 1gb in the disk. We can now pick the first 150mb of data from each file and load it in memory and use the remaining $1000 - 150 \times 5 = 250$ mb of memory to perform our operations. We now perform 5-way merging using these already sorted 5 arrays.

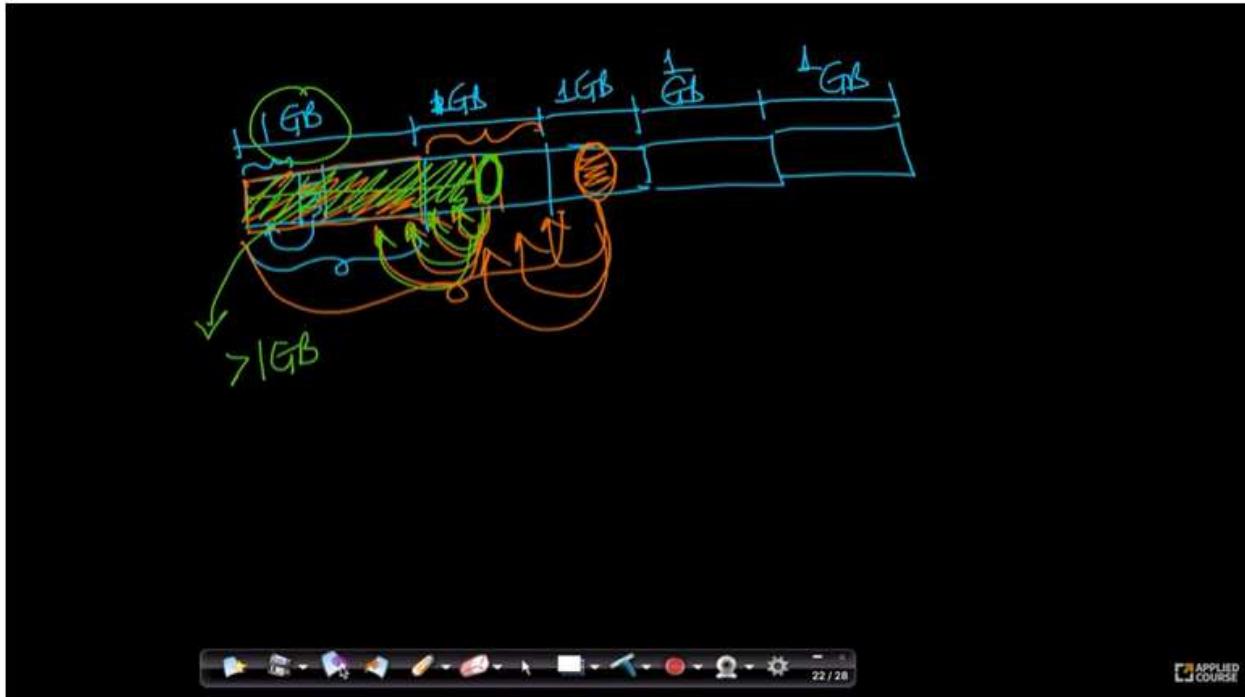
5-way merging is done similar to how merge is done in merge sort. But instead of combining 2 arrays we combine 5 arrays by combining first elements of each arrays and picking the smaller one and placing it in the output array which occupies the above said 250 mb of memory and updating the markers on the array.



Timestamp: 17:41

So once we find these 250 mb of sorted elements, this array is stored to disk and the 250 mb memory is freed. We then again start filling the next 250 mb and again store it in to disk.

Once all the elements in any of the 150mb array are done, we can load the next 150mb of data from the corresponding file. This continues until all the data of all the files is now finally merged and the output is stored in the disk.



Timestamp: 18:56

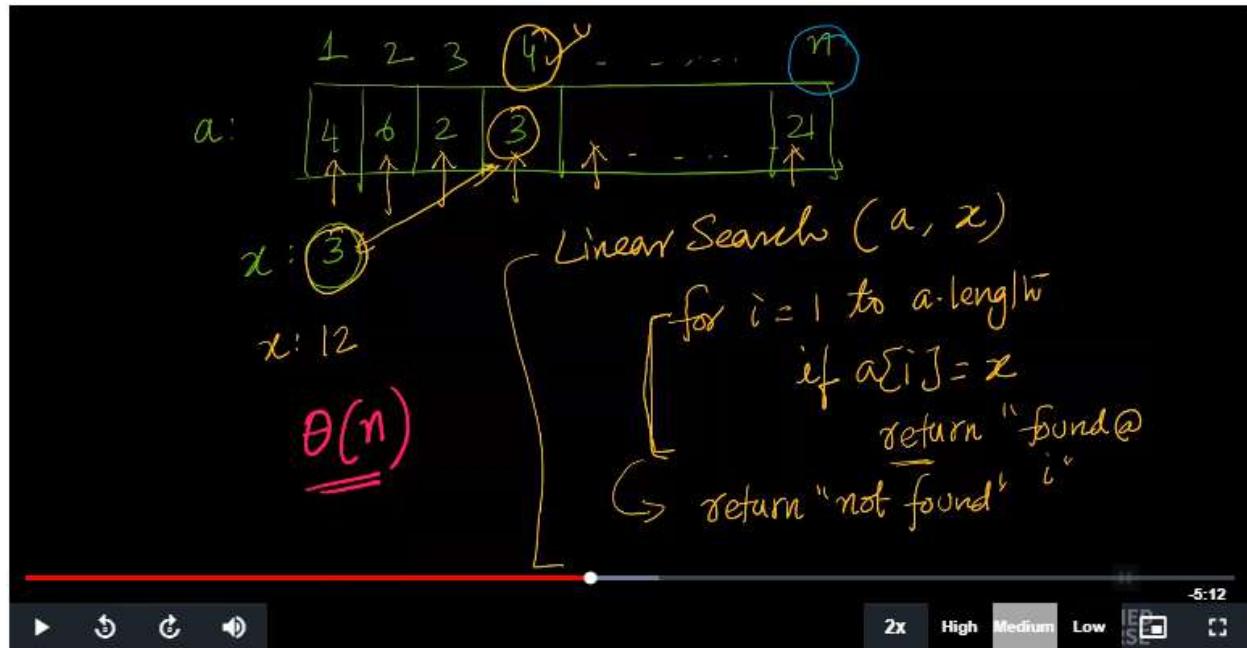
Insertion sort cannot perform external sorting easily because with insertion sort if the data is divided and sorted. Then each of the elements in the second array should again be compared with each element in the first array the way insertion sort works.

So insertion sort is not an external sort but merge sort using k-way merging is an external sorting algorithm because of its divide and conquer approach.

Please also refer to this link https://en.wikipedia.org/wiki/External_sorting#External_merge_sort.

External sort is helpful in real-world scenarios like sorting all of amazon products by price.

11.12 Linear Search



Timestamp: 4:34

Search is the problem of finding whether an element x is present in an array a . Linear search is a simple searching algorithm that compares each element of the array a with the search element x as shown in the above figure.

If the element is present in the array we will return the index at which the element x is present in the array a . If x is not present in the array a then it will return array is not found.

There is an another variation of the problem where we want to return all the indices where x is present. For that problem we may need to maintain another array to which we will add the indices at which x is present as we move along the array a comparing each element with x .

So the time complexity of linear search is $\Theta(n)$ where n is the size of the array since we loop through each element of the array and the space complexity of linear search is $\Theta(1)$ since we are only one additional looping variable i apart from the data given.

Linear search can work on linked list as well as we are comparing the elements with the search element x sequentially.

11.13 Binary Search -- intuition

intuition

Binary Search

$A: \begin{matrix} l=1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10=r \end{matrix}$ $n=r=10$

$\rightarrow \text{SORTED}$

$\begin{cases} x = A[m] \rightarrow \text{FOUND } x \text{ in } A \\ x > A[m] \rightarrow l = m+1 \\ x < A[m] \rightarrow r = m-1 \end{cases}$

① $x = 6$ found at $m = \lfloor \frac{r+l}{2} \rfloor = \lfloor \frac{1+10}{2} \rfloor = 5$

② $x = 6$ not found ($x < A[m]$)

l, r, m

7:14

Timestamp: 10:54

To perform binary search we need the array A to be sorted. To search an element x in an array A in binary search at each iteration

- we will check whether the middle element in the array is the search element x.
- If the middle element is not the search element, then we will compare whether the search element is greater than the middle element, if it is greater then we will search the right half of the array.
- If the search element is less than the middle element then we will search the left half of the array.

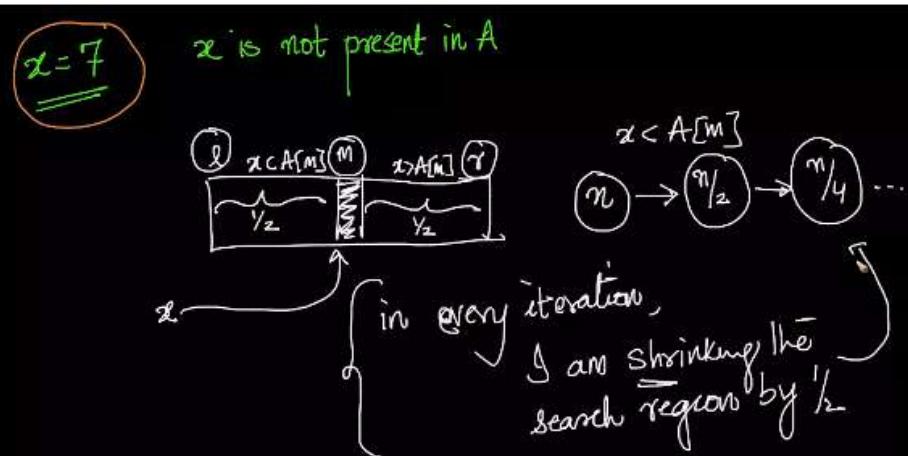
We repeat this process until we find the element or we are only left with one element array and the element is not the search element in which case the search element is not present in the array.

③ $l=3; r=4$
 $m = \lfloor \frac{r+l}{2} \rfloor = \lfloor \frac{7+3}{2} \rfloor = 5$
 $x=6 \quad A[m]=6$
 $x==A[m] \Rightarrow \text{FOUND } x \text{ in } A$



Timestamp: 11:45

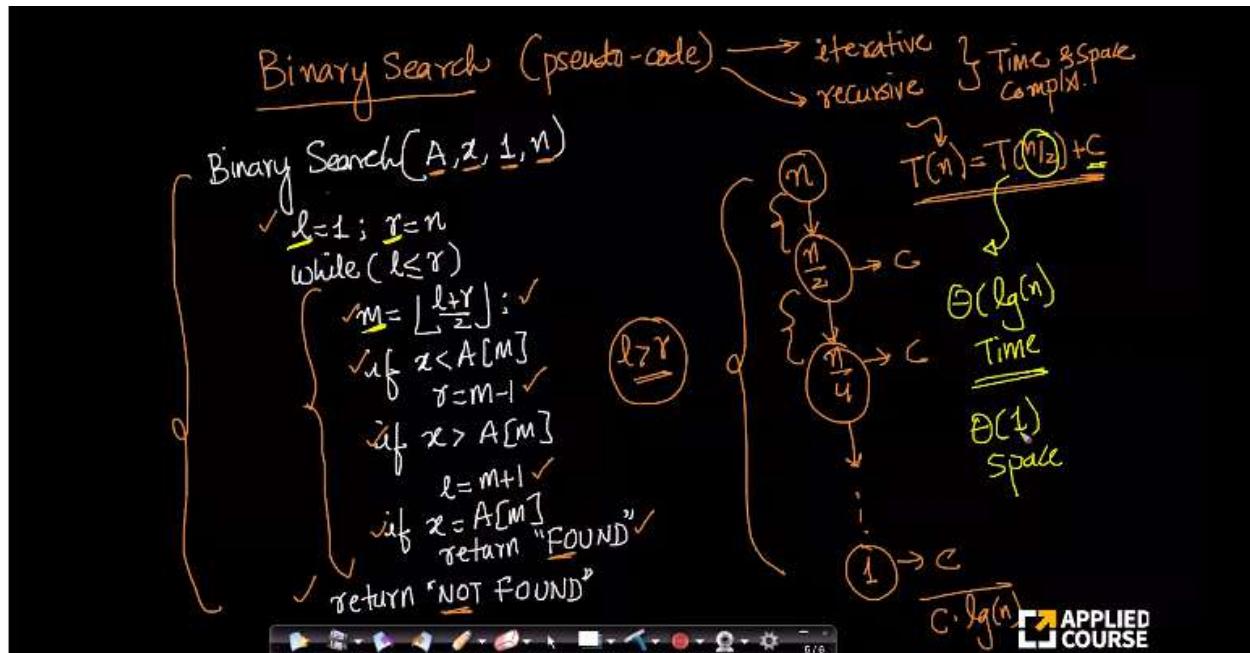
Please work through the example in the above figure where search element $x=6$.



Timestamp: 17:39

Notice that for every iteration we are shrinking our search region by half as shown in the figure.

11.14 Binary Search -- Pseudo Code



Timestamp: 4:27

In the above figure we can see that at iteration of the loop, we are performing constant amount of operations such as calculating the middle index m, comparing search element with the middle element,etc hence each iteration the time complexity is $\Theta(1)$. Notice that since at each iteration we are reducing the search space to half until we reach 1, hence the number of iterations for searching an array with n elements is $\lg n$. Hence the total time complexity is $\Theta(\lg n)$.

Notice that throughout the entire pseudo code we are only creating additional variables like l,r,etc which doesn't depend on n, hence the space complexity is $\Theta(1)$.

The screenshot shows a video player interface with a black background. At the top, there is handwritten pseudo code for a recursive binary search algorithm. The code is as follows:

```
BinSrchRecursive(A, x, l, r)
  if (l > r)
    return "NOT FOUND"
  m = ⌊(l+r)/2⌋
  if x == A[m]
    return "FOUND"
  if x < A[m]
    BinSrchRecursive(A, x, l, m-1)
  if x > A[m]
    BinSrchRecursive(A, x, m+1, r)
```

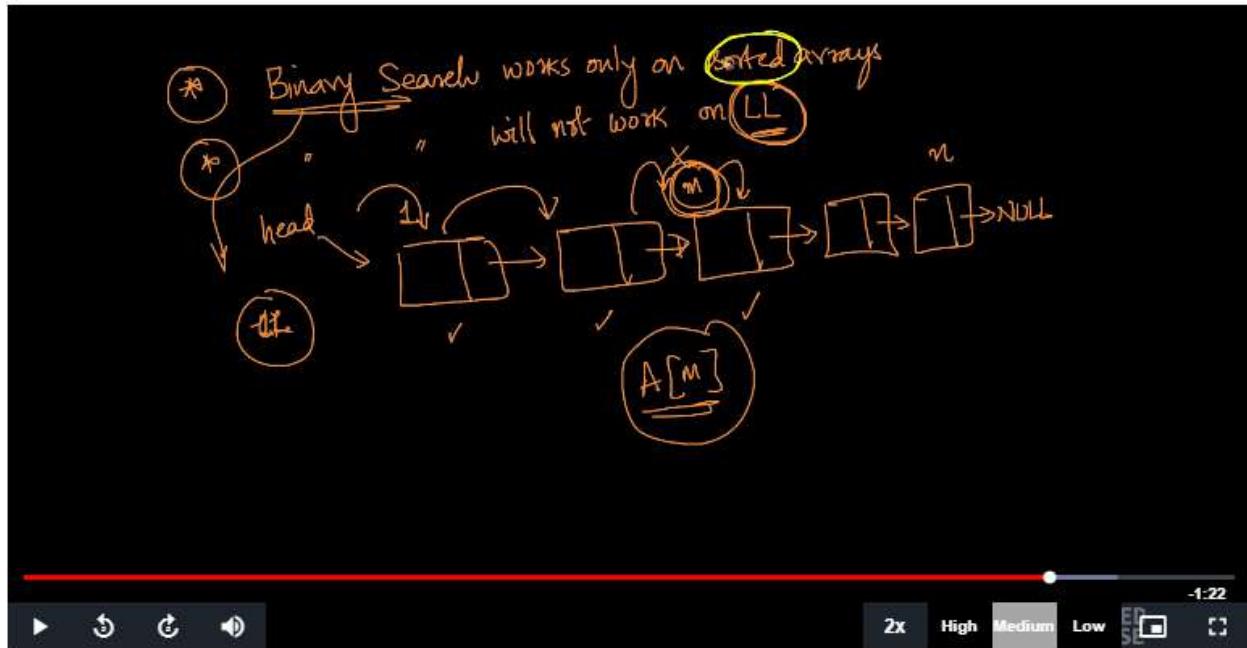
Below the video player, there is a control bar with icons for play, pause, and volume. To the right of the control bar, there are three buttons labeled "2x", "High", "Medium" (which is highlighted), and "Low". Further to the right are buttons for "SUBTITLE" and "SCREENDRAG". The timestamp "3:47" is located at the top right of the video player.

Timestamp: 5:16

The above figure shows the pseudo code for the recursive implementation of binary search. Note that at each recursive call the amount of time taken for a search space of n is the sum of constant time taken for checking the if conditions and the time taken for the next recursive call with search space $n/2$.

$$T(n) = T(n/2) + \Theta(1)$$

Solving this should once again give us the time complexity of binary search to be $\Theta(\lg n)$.

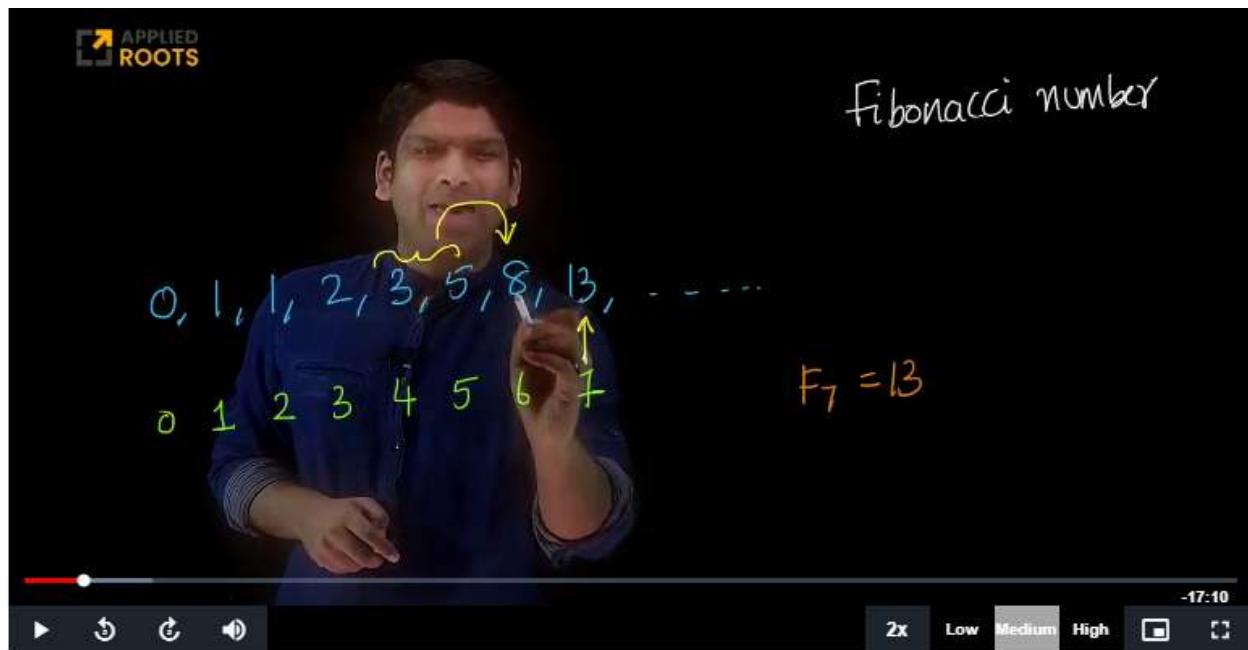


Timestamp: 7:41

Binary search will work only on sorted array since in binary search we reduce our search space by half after comparing with the middle element because if the array is sorted and the search element is less than the middle element(let's say) then the element can only be found in the first half. Since it is less than the middle element there is no way it can be less than the elements that are greater than the middle element.

Binary search doesn't work in the case of linked list since in binary search using arrays we are directly accessing the middle element of the array in constant time, we cannot do the same incase of linked list.

11.16 Fibonacci numbers using Recursion



Timestamp: 1:16

The above sequence of numbers is called Fibonacci series and the 0th fibonacci number is 0 and is indicated by F_0 , the 1st fibonacci number is 1 and is indicated by F_1 . If you notice from the 3rd Fibonacci number, the fibonacci number $F_i = F_{i-1} + F_{i-2}$.

The fibonacci numbers occurs in nature in many forms. Please use wikipedia for the same.

The video shows a man in a blue shirt explaining a recursive equation for the nth Fibonacci number. The equation is:

$$F_n = \begin{cases} F_{n-1} + F_{n-2} & \text{if } n > 2 \rightarrow \text{recursive case} \\ 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \end{cases} \rightarrow \text{base-cases}$$

The video player interface at the bottom includes a progress bar, control buttons (play, stop, etc.), and a settings bar with 2x, Low, Medium (selected), High, and other options.

Timestamp: 4:24

The nth Fibonacci number can be found using the above recursive equation. Notice for the base cases 0 and 1 we need not use any recursive call. We have two base cases and one recursive case.

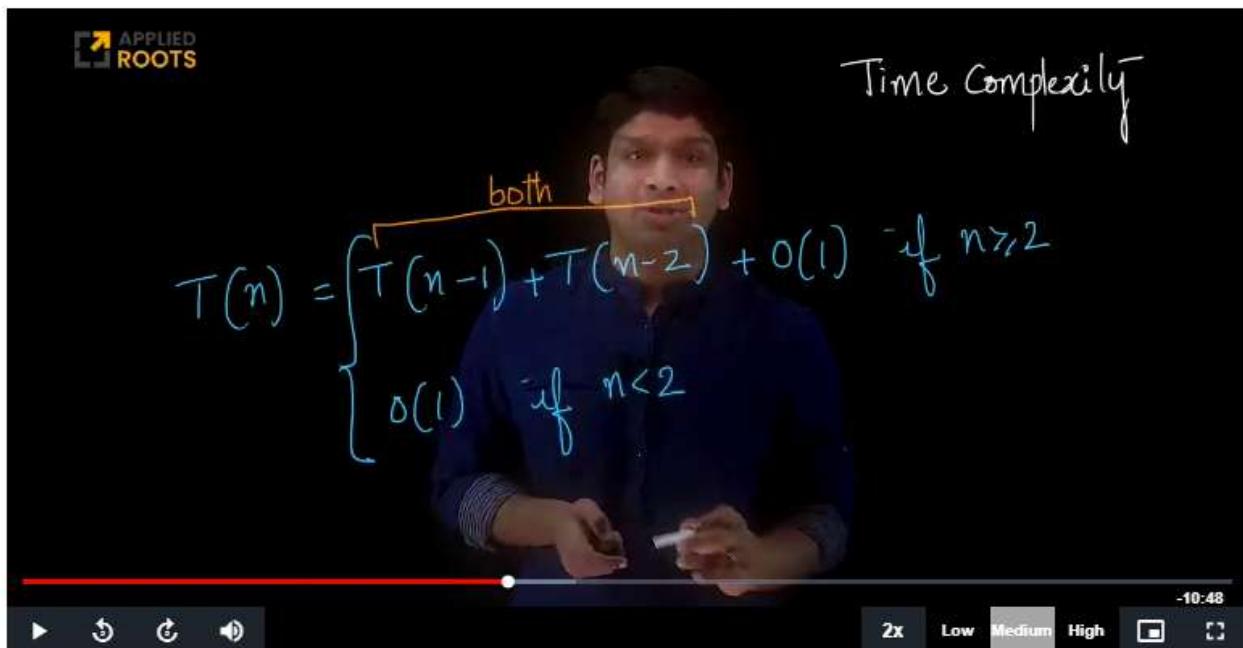
The video shows a man in a blue shirt explaining the time complexity of the recursive Fibonacci function. The code is:

```
1 def fibonacci(n):
2     if n==0:
3         return 0
4     if n==1:
5         return 1
6     fib_n = fibonacci(n-1) + fibonacci(n-2) → T(n-1) + T(n-2)
7     return fib_n
+ O(1)
```

The video player interface at the bottom includes a progress bar, control buttons, and a settings bar with 2x, Low, Medium (selected), High, and other options.

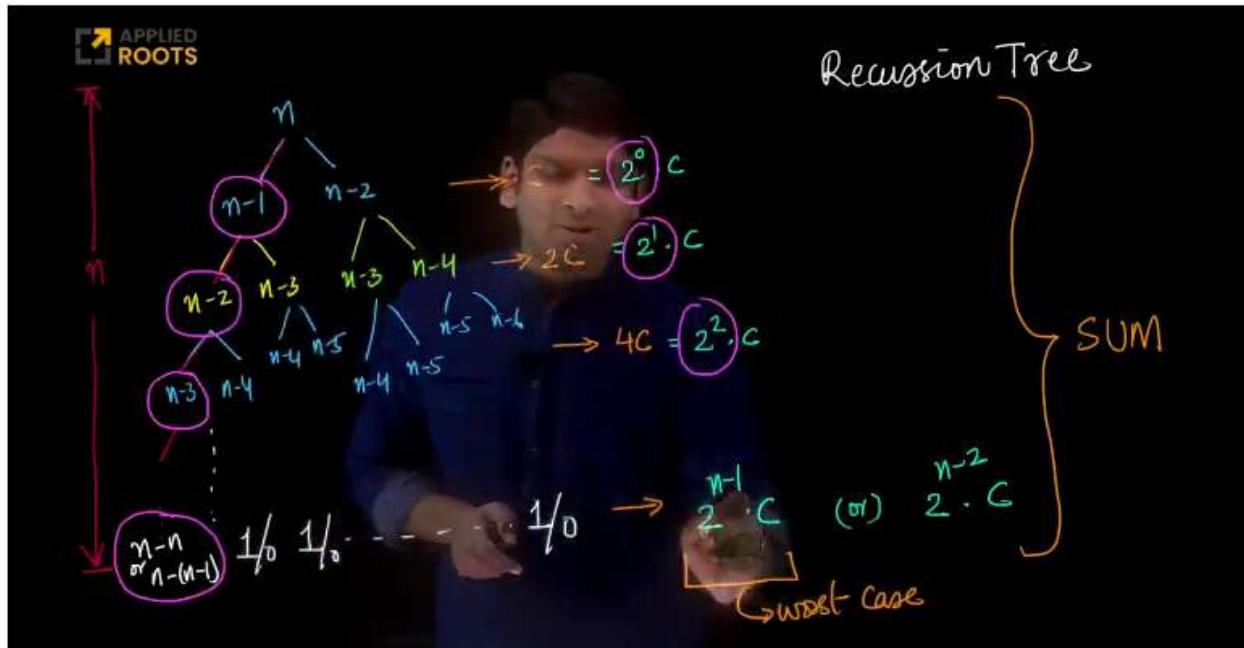
Timestamp: 6:22

The above figure shows to calculate the nth fibonacci number. Notice that for each recursive call we take O(1) time to check the base cases and the time taken for two recursive calls for calculating n-1 and n-2 fibonacci number respectively.



Timestamp: 7:15

The recursive equations looks as above. For the base case we take $T(n)= O(1)$ time else $T(n) = T(n-1) + T(n-2) + O(1)$, $O(1)$ time for addition of the previous two fibonacci numbers and for checking base conditions.



Timestamp: 13:08

Notice that using recursion tree as in the above picture we can find that the number of levels are n since we see that the leftmost element decreases by 1 each level until we reach base cases either 0 or 1.

Notice that each level i if the first element is $n-i$, we are need to do $2^{i-1} \cdot C$ number of operations. Summing the total time taken at all levels we can find that the sum follows geometric progression.

$$\sum_{i=0}^{n-1} 2^i \cdot C = C \cdot 2^0 + C \cdot 2^1 + C \cdot 2^2 + \dots + C \cdot 2^{n-1}$$

x 2 x 2

Geometric progression

Timestamp: 14:32

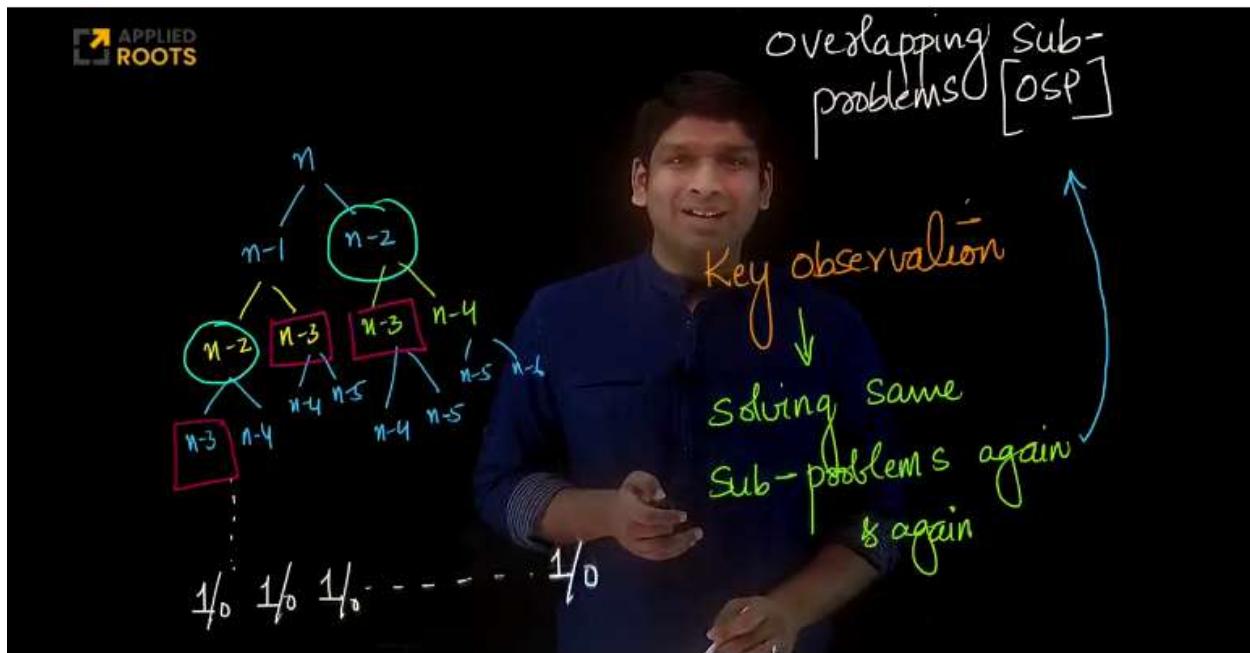
$$\Rightarrow \sum_{k=0}^{n-1} c \cdot 2^k = c \left(\frac{1-2^n}{1-2} \right) \quad r=2 \neq 1$$
$$= \frac{c (2^n - 1)}{2 - 1} = c (2^n) - c = \underline{\underline{O(2^n)}}$$

Very bad!!

Timestamp: 17:40

Solving the geometric progression, we can find that the time complexity of finding nth fibonacci number is $O(2^n)$ which is worse. In the next video, we will see how we can improve the time complexity.

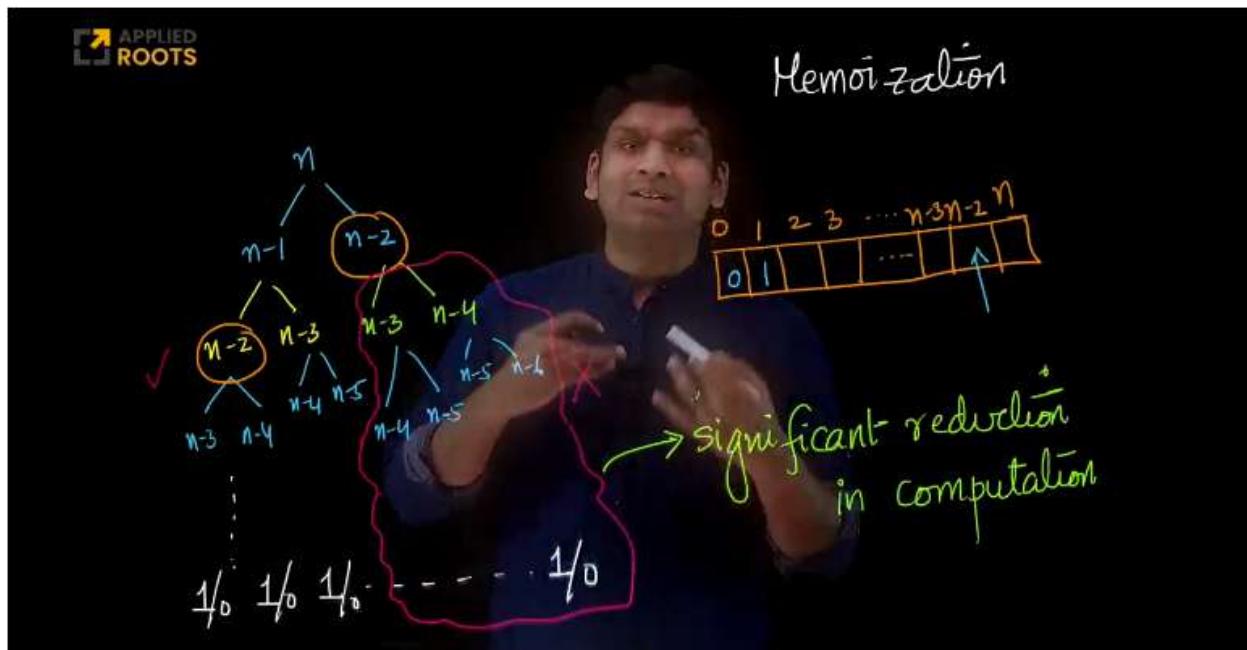
11.17 Dynamic Programming (For Fibonacci numbers)



Timestamp: 2:59

Notice in the recursion tree while calculating the n^{th} Fibonacci number, we are calculating fibonacci numbers such as $n-2$, $n-3$ many many times throughout the recursion tree. This issue of solving the same subproblems again and again to solve a larger subproblem is called overlapping sub-problems [osp]. One strategy to overcome this is to use a technique called memoization.

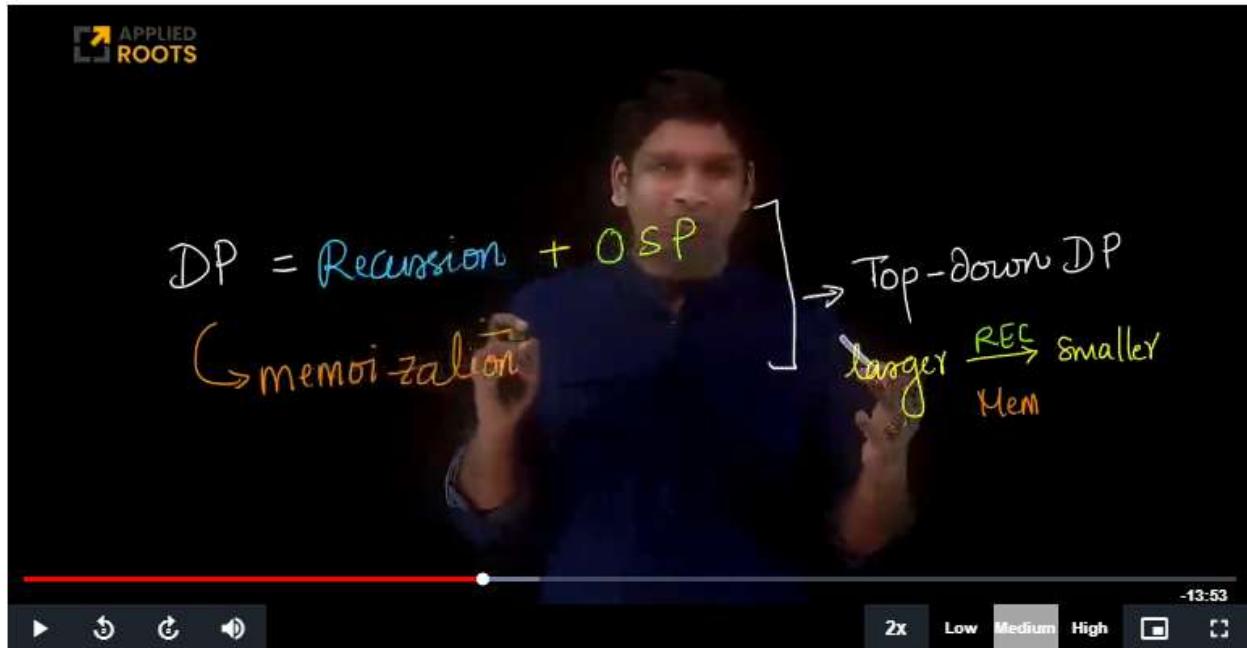
Memoization



Timestamp: 7:32

So the technique called memoization is to store the result of solving subproblem into a data structure such as array and using it directly instead of solving it again. For example in our problem of finding n th Fibonacci number we can store the $n-2$ th fibonacci number once computed in the left subtree in an array at index $n-2$ and directly use it when we need it in the right subtree.

So for calculating any subproblem we will first check whether it is already present in our array, if it is then we will directly use it else we will compute it and store in the array and we will use it from next time directly from the array.

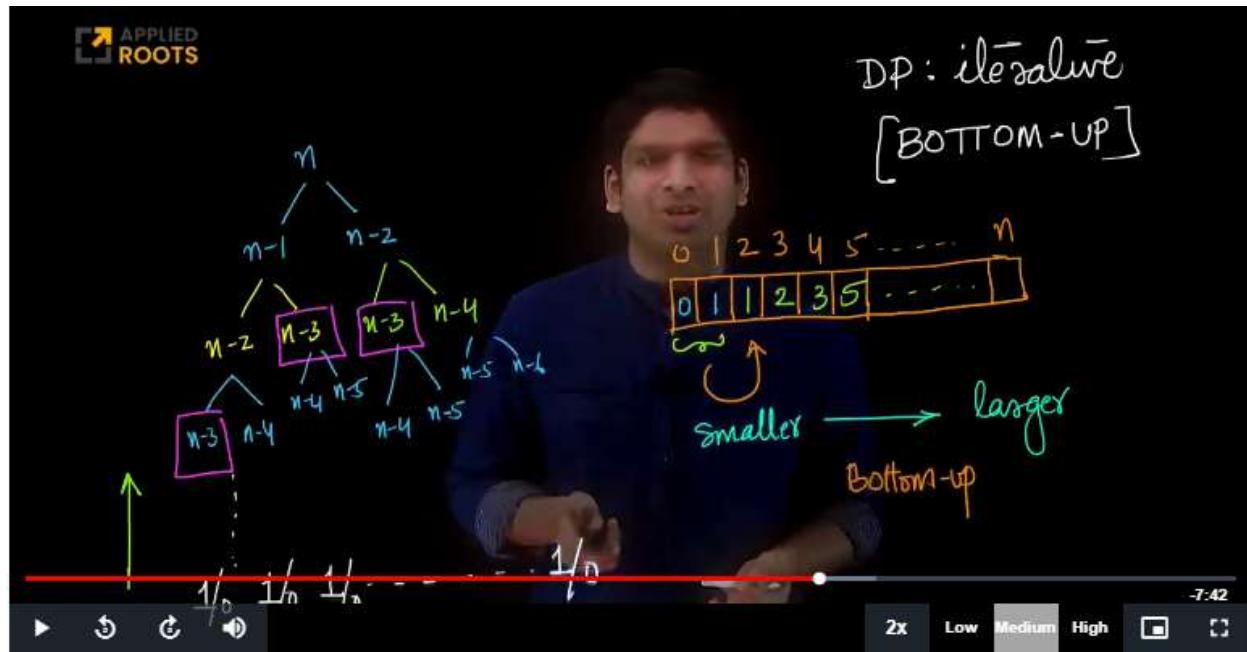


Timestamp: 8:29

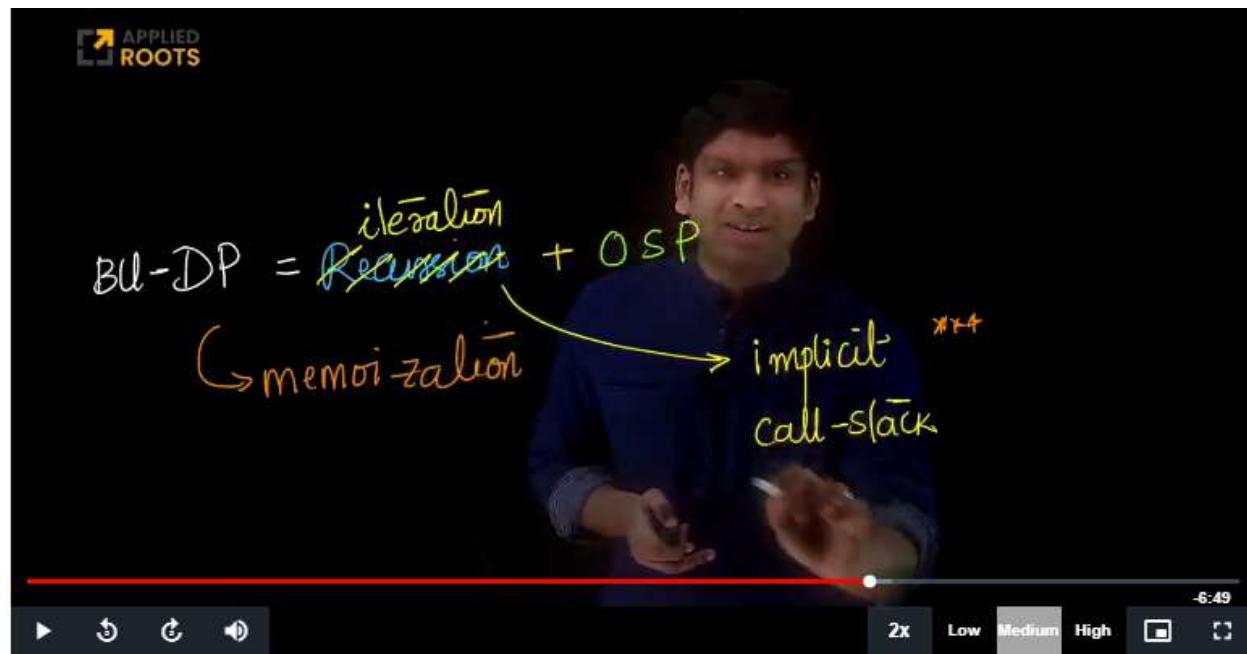
When we have a recursive subproblem and we noticed overlapping subproblems and if we use memoization to solve the problem then it is called dynamic programming (DP)

There are two types of dynamic programming

- a) Top down approach, in which we will solve larger problem by recursively solving smaller subproblems and use memoization. Recursion is often referred to as optimal substructure.
- b) Bottom up approach, where we will populate the datastructure (in this case array) by solving the subproblems and storing them and using them to solve the larger subproblems as shown in the below figure.



Timestamp: 14:40



Timestamp: 15:33

Memoization is often referred to as tabulation when we were are implementing bottom up DP. We often use bottom-up approach due to the implicit call stack caused by using recursion in the top-down approach.

Strategy

- Use examples → recursion equation
- observe OSP, if present
- Memoization + iterative (bottom up)

6:00

▶ ⏴ ⏵ 🔍 🔊 2x Low Medium High

Timestamp: 16:22

The strategy for solving a problem is to use examples and find its recursive equation. Check if it contains overlapping subproblem (OSP), if it is present then use memoization and iterative i.e bottom up DP.

Let us look at the code and time complexity for bottom-up DP approach.

Time complexity

```
1 def fibonacciIterative(n):
2     #memoization using a list
3     f = []
4
5     #base cases
6     f.append(0)
7     f.append(1)
8
9     # bottom-up DP
10    for i in range(2,n+1):
11        f.append(f[i-1] + f[i-2])
12
13    return f[n]
```

from $O(2^n)$

$\rightarrow O(1)$

$\rightarrow O(n)$ ✓

4:12

▶ ⏴ ⏵ 🔍 🔊 2x Low Medium High

Timestamp: 18:10

If we can observe from the above figure our iterative DP is taking $O(1)$ time for populating our array with base cases 0 and 1 and taking $O(n)$ time for calculating nth fibonacci number. So the total time complexity for our iterative DP for calculating the nth fibonacci number is $O(n)$ which is a significant drop from the time complexity $O(2^n)$ of our recursive algorithm with no memoization.

Let us look at the space complexity of iterative DP for our problem.

```
1 def fibonacciIterative(n):
2     #memoization using a list
3     f=[]
4     #base cases
5     f.append(0)
6     f.append(1)
7
8     # bottom-up DP
9     for i in range(2,n+1):
10        f.append(f[i-1] + f[i-2])
11
12    return f[n]
```

Space complexity
↓
additional space
[don't count input & output]

Timestamp: 19:16

Notice that our space complexity has become $O(n)$ due to memoization since we have to store the results of smaller subproblems to solve larger problems. Our space complexity increased from $O(1)$ time (although recursion uses call stack memory implicitly) in recursive approach with no memoization to $O(n)$ time with iterative approach using memoization.

Notice that by using iterative bottom up approach we are able to make time complexity come down from $O(2^n)$ to $O(n)$ with increase in explicit space complexity from $O(1)$ to $O(n)$. But this is completely fine because our ram sizes will increase considerably overtime and we are more concerned about time rather than memory.

11.18 Longest Common Sub-Sequence(LCS) using DP

The screenshot shows a YouTube video player interface. At the top, the title is "Longest common Subsequence (LCS) → Dyn. Prog." with a link to "Applicable". Below the title, there are "Watch later" and "Share" buttons. The video content includes handwritten notes:

- eg1**:
S1: A B C D G H
S2: A E D F H R
Note: $\text{LCS}(S1, S2) = \underline{\text{ADH}}$ of len(3) → maximal len
AD is a common Sub-seq of S1 & S2
(ADH) is a Sub-seq of S1 and S2
- eg2**:
S1: A G G T A B
S2: G X T X A Y B
 $\text{LCS}(S1, S2) = \underline{\text{GTAB}}$ of len 4

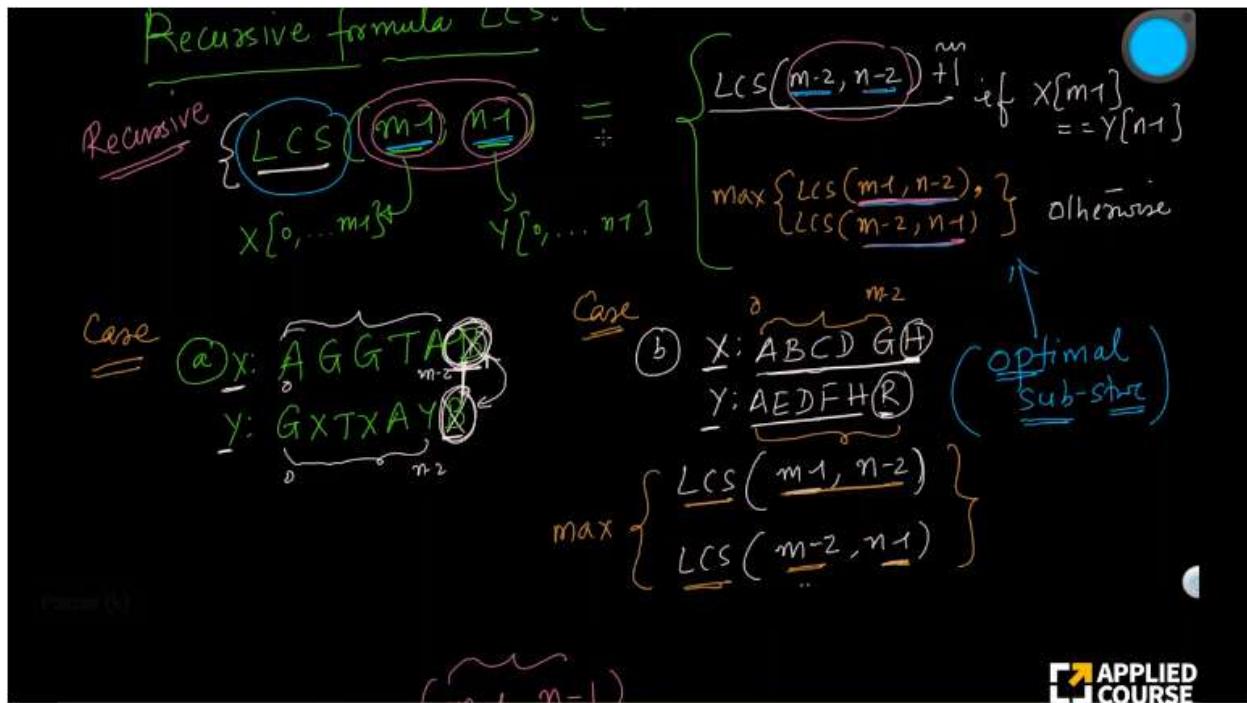
At the bottom of the video player, there are controls for "MORE VIDEOS", a progress bar showing "5:26 / 32:36", and a "YouTubeIE COURSE" logo.

Timestamp: 5:26

Longest common subsequence is the problem of finding the longest subsequence that is common between the two string. Please note that subsequence unlike substring doesn't have a condition that all the elements in the subsequence should be neighbours rather maintaining the same order as in the original sequence or string is enough to be a subsequence.

In the above figure, for eg1 ADH of length 3 is the longest common subsequence whereas for eg2 GTAB of length 4 is the longest common subsequence.

LCS has applications in genomics where we have to find the longest common subsequence between two DNA. It also has applications in finding difference between two files, identifying plagiarism, etc. That is why it is important to solve LCS problem.



Timestamp: 22:56

The recursive formulation of LCS is as above. The key intuition for this:

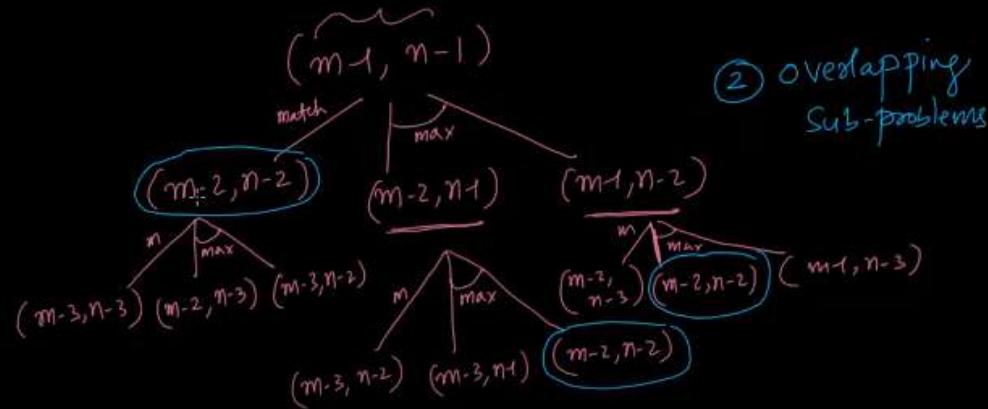
- As shown in case a in the above figure, if the last character in both the sequences match then the character is bound to be in the lcs of the sequences. Hence we will just add the length 1(length of the last character) and compute the lcs of the remaining part of both the sequences.
- As shown in case b in the above figure. If the last character in both sequences doesn't match then at least one of them is bound to be miss in the final lcs of the sequences, hence we will have two recursive calls excluding the last character of one sequence and including the last character of the other sequence in one call and vice-versa in the other.

Notice in the above figure there is an optimal substructure when we are trying to solve our bigger problem by dividing into smaller subproblems using recursion.



LCS (Dynamic Programming) | Application

Watch later Share



MORE VIDEOS



23:22 / 32:36



YouTube IE6 COURSE

Timestamp: 23:22

In the above figure we can notice in the recursion tree of our solution we have overlapping subproblems as highlighted by the blue boxes.

If we notice that in the worst case if our last characters of both the sequences doesn't match at each level we have to solve two subproblems and solve the max between them. So at level 1 we will be solving 2 subproblems, then at level 2 we will be solving 4 subproblem and so on and at last level h we will be performing 2^h sub problems and the no of levels h is $m+n$ since the recursion tree continues until both $(m-m, n-n=0)$ happens on all branches.

Hence at last level we have 2^{m+n} subproblems to solve and each takes $O(1)$ time. So the total time complexity of the recursive algorithm is $O(2^{m+n})$.

But since we have both optimal substructure and overlapping sub problems we can solve this problem using bottom-up dynamic programming as shown in the below figures.

$\ell(s(X, Y, \underline{m}, \underline{n}))$

$X: \{0, 1, 2, \dots, m\}$
 $Y: \{0, \dots, n\}$

$L[\underline{m+1}][\underline{n+1}]$: 2D array

for $i = 0$ to m
 for $j = 0$ to n
 if $i = 0$ or $j = 0$
 $L[i, j] = 0$



Timestamp: 28:49

bottom-up
Tabulated

$\ell(s(X, Y, \underline{m}, \underline{n}))$

for $i = 0$ to m
 for $j = 0$ to n
 if $i = 0$ or $j = 0$
 $L[i, j] = 0$
 else if $X[i-1] = Y[j-1]$
 $L[i, j] = L[i-1, j-1] + 1$ → last char match
 else
 $L[i, j] = \max\{L[i-1, j], L[i, j-1]\}$ ↑ otherwise

$LCS[\underline{0,0}] \downarrow$
 $\{0, 1\} \downarrow$
 $\{1, 0\}$
 $\{r, 1\}$
 $\{m, n\}$

$\text{Time: } O(m \cdot n)$
 $\text{Space: } O(m \cdot n)$

return $L[m, n]$



Timestamp: 31:59

Since we are following bottom-up approach when either of the sequences have length 0 then we are making LCS is 0 then we check whether the last character is same and call recursion accordingly.

If we notice the time complexity of the algorithm is $O(m*n)$ since we are using two for loops of length m and n and we are performing $O(1)$ operations for memoization, max etc within each iteration.

Hence using dynamic programming we are able to bring down an exponential time of complexity $O(2^{m+n})$ to $O(m*n)$ for solving our LCS problem.

11.19 LCS: Worked out example

$\checkmark S_1: QPQRR \quad \text{len } LCS = 4$

$\checkmark S_2: PQPRQR \quad S_1[m] = S_2[n]$

$LCS(m,n) = \begin{cases} LCS(m-1, n-1) + 1 & \text{if } \text{match} \\ \max\{LCS(m-1, n), \\ LCS(m, n-1)\} & \end{cases}$

$LCS(2,1) = S_1[2] = S_2[1] \quad \checkmark$

$LCS(1,0) + 1$

Table:

	0	Q	P	Q	R	R
0	0	0	0	0	0	0
1: P	0	0	1	1	1	1
2: Q	0	1	2	2	2	2
3: P	0	1	2	3	3	3
4: R	0	1	2	3	4	4
5: Q	0	1	2	3	4	4
6: R	0	1	2	3	4	4
7: P	0	1	2	3	4	4

bottom-up DP

(LCS(S1, S2) = 4) ✓

APPLIED COURSE

Timestamp: 11:02

Please workout the above example of finding the LCS between sequences S1: QPQRR and S2: PQPRQR.

Note that when there is a match between the column character and the row character at (m,n) we add 1 to the value at (m-1,n-1) and store in (m,n) otherwise we take the max of (m-1,n) and (m,n-1) and store in (m,n).

$\checkmark S_1: \underline{Q P Q R R}$ } $LCS = 4$
 $\checkmark S_2: P Q P R Q R P$
 $S_1[m] == S_2[n]$

$LCS(m,n) = \begin{cases} LCS(m-1, n-1) + 1 & \text{if } S_1[m] == S_2[n] \\ \max\{LCS(m-1, n), \\ \quad LCS(m, n-1)\} & \end{cases}$

$LCS(2,1) = S_1[2] == S_2[1] \checkmark$

actual LCS $LCS(1,0) + 1$ bottom-up DP Table

$(Q P Q R)$ ✓ $(P Q R R)$ $LCS(S_1, S_2) = 4$ ✓ $\underline{Q P R R}$

APPLIED COURSE

Timestamp: 18:11

From the above figure we can see that we can even trace back through our table to get the actual LCS along with length of LCS.

Starting from the end result of 4, we traceback following from which cell we got the value 4, notice that the end result 4 is calculated using the max of (7,5) or (6,6) hence we can take either paths. If we take path (6,6) which contains 4 we got 4 by incrementing the value at (5,5) by 1 since there is match at (6,6) between R and R. Likewise we can trace back to the start.

By taking multiple paths we can end up with different LCS which are all correct. Please work it out by following different color paths and match with the exact color LCS in the above figure.

11.20 Matrix Multiplication using DP

-27:13

$$\textcircled{2} \quad A \underset{10 \times 5}{\underbrace{B}} \underset{30 \times 5}{\underbrace{C}} = (\underline{AB})C = A(\underline{BC})$$

$A: 10 \times 30; B = 30 \times 5; C = 5 \times 60$

$\underline{10 \times 5}$

$R = \{A_1, A_2, A_3, \dots, A_n\}$

Q: What is the optimal order in which we can multiply them?

$\min \# \text{ops}$

$$(AB) \underset{10 \times 5}{\underbrace{C_i}} \underset{5 \times 60}{\Rightarrow} (10 \times 30 \times 5) + (10 \times 5 \times 60) = 4500$$

$$A(\underline{BC}) \underset{10 \times 60}{\Rightarrow} (30 \times 5 \times 60) + (10 \times 30 \times 60) = 27000$$

$P_{i-1} \ P_i \ P_i \ P_{i+1} \ P_{i+2} \ \dots \ P_n$

MATRIX-CHAIN-ORDER(p)

2x n-1 High Medium Low

Timestamp: 08:26

Matrix multiplication is the problem of finding which matrices to multiply first in the multiplication sequence of matrices $A_1 * A_2 * A_3 * A_4 \dots * A_n$. Given a matrix like A_i to be $P_{i-1} \times P_i$. For example, multiplying matrix sequence $A * B * C$ can be multiplied in two ways i) By first computing $A * B$ and then $(A * B) * C$ ii) By first computing $B * C$ and then $A * (B * C)$. The way which reduces the number of multiplications has to be chosen to reduce the number of operations and save time.

Matrix Chain multiplication : Dynamic prog

① $A_{10 \times 30} B_{30 \times 5} = T_{10 \times 5} \Rightarrow \# \text{operations} = 10 \times 30 \times 5$

$\left\{ \begin{array}{c} 10 \\ i \end{array} \right\} \left[\begin{array}{c} 30 \\ \boxed{0001111} \end{array} \right] \times \left[\begin{array}{c} 5 \\ \boxed{\square} \end{array} \right] = \left[\begin{array}{c} \square \\ T \\ 10 \times 5 \end{array} \right] ; A_{m \times n} B_{n \times r} \rightarrow m \times n \times r$

② $A B C = (AB)C = A(BC)$

$A: 10 \times 30; B: 30 \times 5; C: 5 \times 60$

$(AB)C: (10 \times 30 \times 5) + (10 \times 5 \times 60) = 4500$

$A(BC): (30 \times 5 \times 60) + (10 \times 30 \times 60) = 27000$

Timestamp: 29:59

Timestamp: 5:40

As shown in the above figure choosing to perform A^*B then $(A^*B)^*C$ reduced the time taken 6 times for the given matrix sizes. Notice that to multiply matrices of size $m \times n$ and $n \times r$ the number of multiplications to perform is $m \times n \times r$.

① optimal Sub-Strc.

$\{A_i A_{i+1} \dots A_j\}$ Let's say $m[i, j]$: cost of $A_i \dots A_j$

$m[i, j] = \begin{cases} 0 & \text{if } i=j \\ \min_{i \leq k < j} m[i, k] + m[k+1, j] + p_i p_k p_j & \text{if } i \neq j \end{cases}$

$(A_i A_{i+1} \dots A_k)(A_{k+1} \dots A_j)$

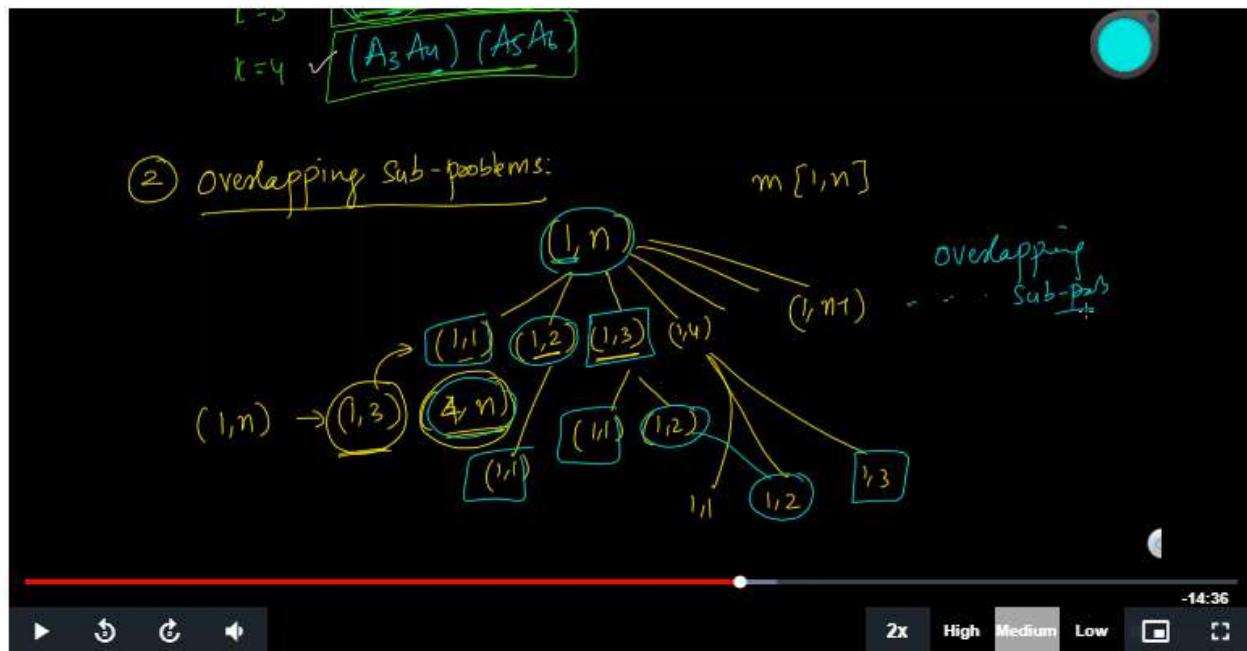
$\checkmark A_3 A_4 A_5 A_6 \rightarrow (A_3 A_4 A_5) A_6$

Timestamp: 21:30

Timestamp: 14:09

Given an array sequence $A_i A_{i+1} \dots A_j$. Let's say $m[i,j]$ is the cost of multiplying the sequence of array A_i to A_j . Then we can solve the problem to find $m[i,j]$ by first addressing the base case.

When $i=j$ it means that there is only one array hence no multiplications are required hence cost $m[i,j]=0$ otherwise for all possible values of k such that $i \leq k < j$ we find which value of k minimizes the combined cost of multiplying first k arrays and the next $n-k$ arrays and the number of multiplications to multiply these resultant two arrays.



Timestamp: 21:03

Notice that as in the above figure the recursion tree for the problem contains the overlapping subproblems as shown in the figure. Since we have optimal substructure and overlapping subproblem we can solve this problem using DP.

In normal recursive implementation without DP it takes $\Omega(2^n)$ time the proof of which will be explained in discrete maths. Note that the time complexity to solve the matrix multiplication problem for $n+1$ arrays is given by the catalan number n .

(Q) Simple recursion \rightarrow Time Complx
 NO DP \rightarrow generalizing functions \rightarrow Discrete Maths ✓
 $(1, n) \rightarrow$ Catalan Number $C_n = \frac{(2n)!}{n!(n+1)!}$
 $C_n \sim \frac{4^n}{n^{3/2}} = \Omega(2^n)$

Timestamp: 24:00

Let us see the below pseudo code to solve matrix multiplication problem using bottom-up DP.

min #ops \leftarrow

$A_1 \dots A_j \quad \left\{ \begin{array}{l} i \geq 1 \\ j \geq i, j \leq n \end{array} \right.$

A_1, \dots, A_n

$d=2$

$i = 1, 2, 3, \dots, n-1$

$j = 2, 3, 4, \dots, n$

$A_1 A_2$ $A_2 A_3$ $A_3 A_4$

bottom-up DP CLRS

MATRIX-CHAIN-ORDER(p)

```

1  $n = p.length - 1$ 
2 let  $m[1..n, 1..n]$  and  $s[1..n-1, 2..n]$  be new tables
3 for  $i = 1$  to  $n$ 
4    $m[i, i] = 0$   $\rightarrow$  base-case
5 for  $l = 2$  to  $n$ 
6   for  $i = 1$  to  $n-l+1$ 
7      $j = i + l - 1$ 
8      $m[i, j] = \infty$ 
9     for  $k = i$  to  $j-1$ 
10     $q = m[i, k] + m[k+1, j] + p_{i-1} p_k p_j$ 
11    if  $q < m[i, j]$ 
12       $m[i, j] = q$ 
13       $s[i, j] = k$ 
14 return  $m$  and  $s$ 

```

$O(n^3)$

① optimal sub-strc:

$m[i, j]$ let's say $m[i, j]$: cost of mul

Timestamp: 33:42

n stores the length of the array since p (the array containing matrix sizes) has $n+1$ elements.

We create matrix m whose element $m[i,j]$ store the minimum cost for multiplying matrices from A_i to A_j . We also create matrix s whose elements $s[i,j]$ stores the value of k that splits the sequence of matrices into $i...k$ and $k+1...j$.

We store 0 in all elements of m with same i and j since the cost of multiplication given only a single array is 0 since no multiplications are required.

We first start with trying to multiply array sequences of length 2 and figure out which two arrays to multiply first. Suppose we were given array A_1, A_2, A_3, A_4 then we have to find out whether multiplying (A_1, A_2) or (A_2, A_3) or (A_3, A_4) is optimal. Then as L increases we will look at combining larger length sequences.

So when $l=2$, i loops from 1 to $n-1$ and j takes value 2 when $i=1$ and when $i=2$ j takes value 3 and so on. So when $i=n-1$ $j=n$. k takes values 1 when $i=1$ and $j=2$, k takes only value 2 when $i=2$ and $j=3$.

So when $i=1$ and $j=2$, k takes value 1, so q becomes $m[1,1] + m[2,2] + \text{pop1p2} = \text{pop1p2}$ since we know that $m[i,i]=0$ so in this case the total cost is the cost to multiply matrices A_1 and A_2 .

For different values of l, for different combinations of i and j (based on l) $m[i,j]$ stores the minimum cost to multiply to array sequence $A_i..A_j$ by first calculating $A_i..A_k$ and $A_{k+1}..A_j$ and s stores the value k by which to split the array sequence $A_i..A_j$.

Please workout using the table similar to how we solve our lcs problem with sample sequence of arrays A_1, A_2, A_3, A_4, A_5 with some examples sizes.

Notice that three loops each for l, i and k each of them runs for at most n times and inside each iteration we do $O(1)$ operations of comparing and storing and adding. Hence the total time complexity is $O(n^3)$.

Notice that we came down from $O(2^n)$ time complexity using normal recursive approach to $O(n^3)$ time complexity using bottom-up dynamic programming approach. Please go to 11.26 for a solved example.

11.21 Subset- Sum problem using DP

The image shows a video player interface with a dark background. Handwritten notes are overlaid on the screen:

Subset-sum problem: (DP)

i/p { $\begin{matrix} 1 \\ 3 \end{matrix}$, $\begin{matrix} 2 \\ 34 \end{matrix}$, $\begin{matrix} 3 \\ 4 \end{matrix}$, $\begin{matrix} 4 \\ 12 \end{matrix}$, $\begin{matrix} 5 \\ 5 \end{matrix}$, $\begin{matrix} 6 \\ 2 \end{matrix}$ } non-ve integers

Q) are there elements in the set s.t their sum = SUM = 9

↪ T/F

$\checkmark \underline{\{4,5\}} \Rightarrow \underline{4+5} = \underline{\text{SUM}} = 9$

Below the video player are standard controls: back, forward, volume, and a progress bar. To the right are buttons for 2x, High, Medium (which is selected), and Low.

Timestamp: 2:31

Subset sum problem is the problem of finding whether a sum of subset of elements from a given set of non-negative integers will be equal to the sum given. In the example in the figure, given a set of elements of non-negative integers set = {3, 34, 4, 12, 5, 2} we want to find whether it contains any subset whose sum of elements is 9. {4, 5} is a subset whose sum is 9 = sum hence there exists a subset.

$\hookrightarrow T/F$
 $\checkmark \{4, 5\} \Rightarrow 4+5 = \underline{\text{SUM}} = 9$
Brute-force. set of n numbers & $\underline{\text{SUM}}$
 ↳ exponential in n # Subsets $\Rightarrow 2^n$
 $O(2^n)$
 Let's generate all subsets & find out if \exists a subset whose sum = $\underline{\text{SUM}}$

Timestamp: 4:50

The brute force approach of solving this problem is to take each subset and check whether the sum of the elements of each subset is equal to the given sum. Since there $O(2^n)$ subsets the time complexity will be exponential in terms of n .

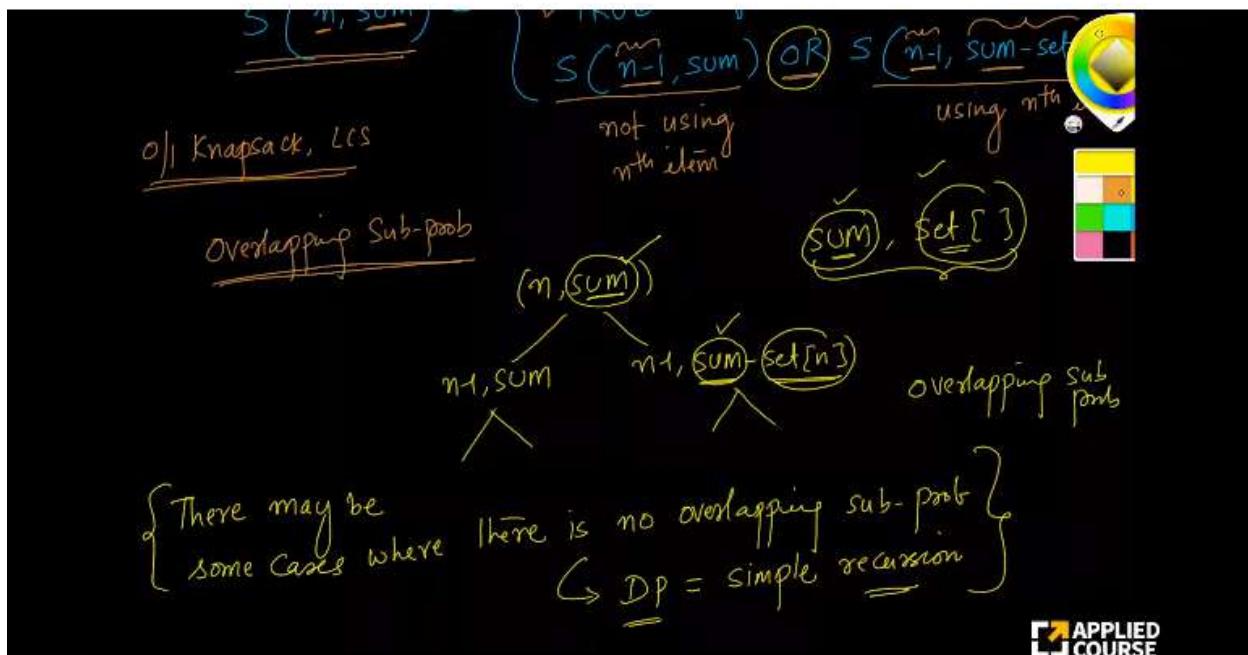
↳ exponential in n # Subsets $\Rightarrow O(2^n)$ & find out if \exists where sum = $\underline{\text{SUM}}$

Recursion: (optimal sub-sols)
 $S(\underline{n}, \underline{\text{sum}}) = \begin{cases} \checkmark \text{FALSE} & \text{if } n=0 \text{ & } \text{sum} > 0 \\ \checkmark \text{TRUE} & \text{if } \text{sum} = 0 \\ S(\underline{n-1}, \underline{\text{sum}}) \text{ OR } S(\underline{n-1}, \underline{\text{sum-set}[n]}) & \text{not using } n^{\text{th}} \text{ item} \end{cases}$

Timestamp: 9:40

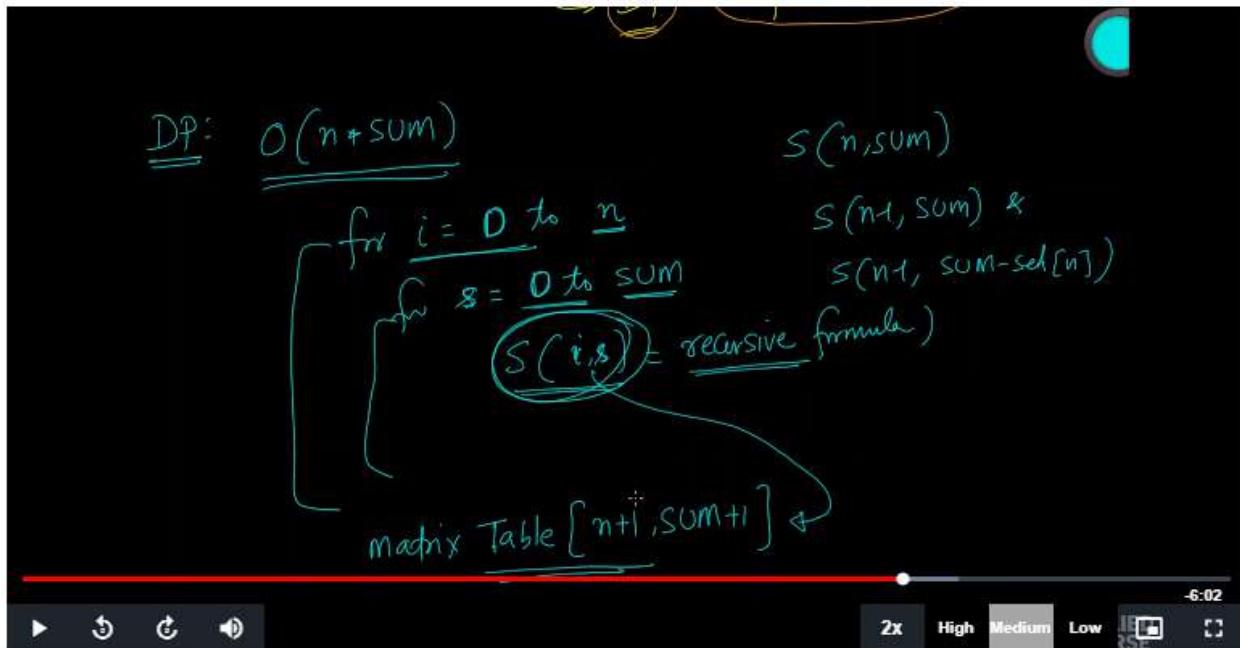
The problem can be solved using recursion as above. Notice that in the base case if we have no elements in the array then we can't find a subset that matches the sum if it greater than 0. If the sum is zero then any subset can satisfy the condition hence we return true.

For the recursive case, there are two cases i) if the last element is not present in the result subset then the problem can be solved by taking the set without the last element ii) If the last element is present in the subset then the remaining sum can be found using all elements in the set without this element. Notice that this recursion equation has optimal substructure.



Timestamp: 12:59

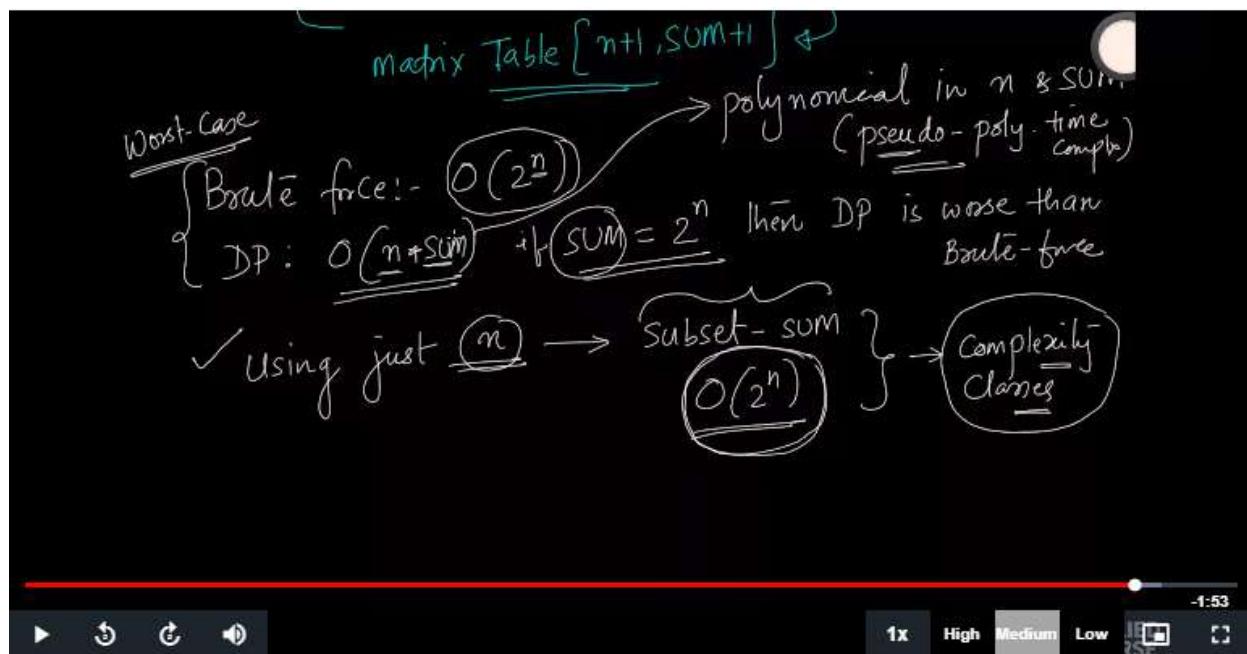
We can find from the recursion tree that there exist overlapping subproblems. But in some cases there might not be a overlapping subproblem then it becomes simple recursion but if it exists then we can use DP.



Timestamp: 16:14

As shown in the above figure, for implementing DP we create a matrix with number of rows to be number of elements and number of columns as sum and we loop through n and sum filling our matrix using the recursion equation.

Since we are using two loops then performing $O(1)$ computation inside them our time complexity of subset sum problem using DP is $O(n*sum)$.



Timestamp: 20:23

Although $O(n^*sum)$ is polynomial in n and sum , sum in worst case can still be a very large number such in $O(2^n)$ hence our DP time complexity can become $O(n2^n)$ which is worse than brute force. Hence no matter what the algorithm we use the time complexity still remains worse. Hence the subset problem belongs to a special case of problems called complexity classes.

11.23 Solved Problem - 1

Consider the following recursive C function that takes two arguments.

```
unsigned int foo(unsigned int n, unsigned int r) {
    if (n>0) return ((n%r) + foo(n/r, r)); // recursive
    else return 0;
}
```

What is the return value of the function foo when it is called as foo(345, 10)?

A. 345
B. 12
C. 5
D. 3

Timestamp: 3:10

You can solve the above problem by going through each recursion call.

First when foo is called on 345, $n \% 10$ [here $r=10$] gives 5 and foo is then called on 34, $n \% 10$ gives 4 and foo is then called on 3, $n \% 10$ gives 3 and then foo is called on 0 since the condition is not satisfied foo returns 0 to its previous call, the previous calls adds its value of $n \% r=3$ to this 0 and returns 3 to the previous call and so on to get the final result 12.

11.24 Solved Problem - 2

The screenshot shows a YouTube video player with the following details:

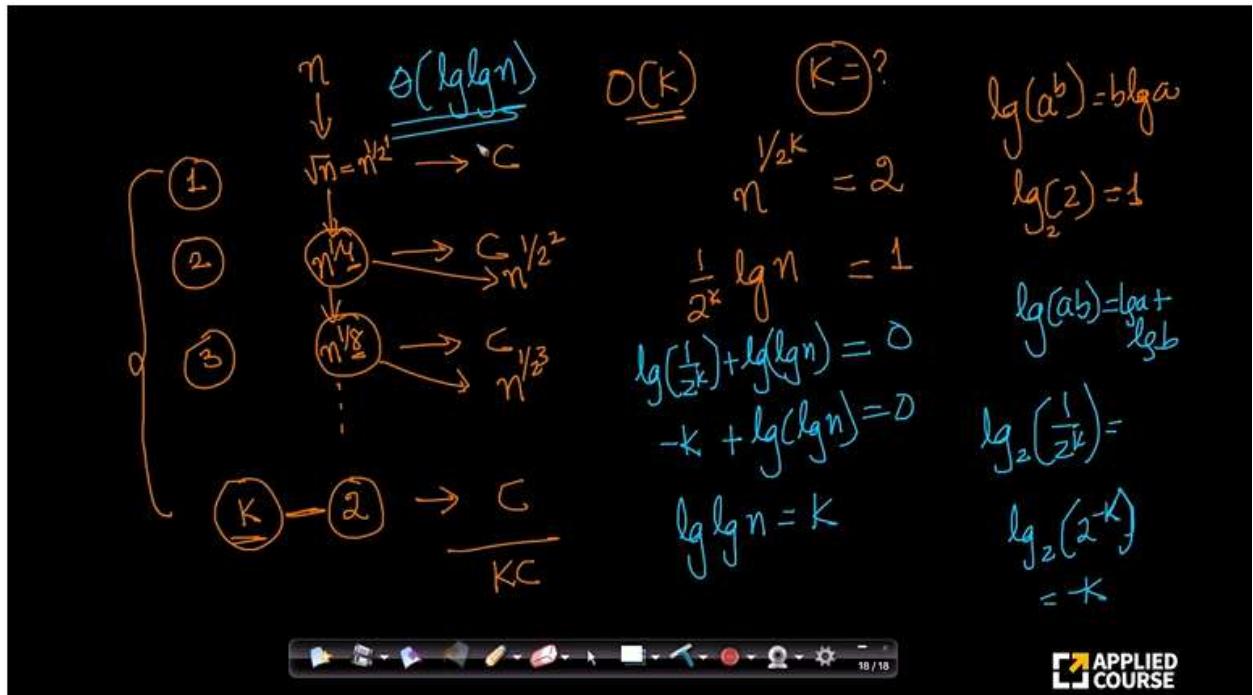
- Title:** GATE 2007 | Solved Problems | Gate Appliedcourse
- Time:** 1:15 / 6:20
- Content:**
 - Handwritten note: $T(n) = T(\sqrt{n}) + C$, $T(\leq 2) = C$
 - Text: What is the time complexity of the following recursive function?
 - C code:

```
int DoSomething (int n) {
    if (n <= 2)
        return 1; // base case
    else
        return (DoSomething (floor (sqrt(n))) + n);
}
```
 - Multiple-choice question:
 - A. $\Theta(n^2)$
 - B. $\Theta(n \log_2 n)$
 - C. $\Theta(\log_2 n)$
 - D. $\Theta(\log_2 \log_2 n)$
- Controls:** MORE VIDEOS, Settings, YouTube APPLIED COURSE

Timestamp: 1:15

We can observe from the given question in the above figure that when $n \leq 2$ we have the base case and we just simply return 1 but if n is greater than 2 we have the recursion case of calling the function on floor of sqrt of n and adding n . Since we are doing constant $\Theta(1)$ work in if and in adding n at each recursive call, the recursion equation is as mentioned in the above figure.

We will solve the recurrence equation using the recursion tree method as below.



Timestamp: 5:39

We can notice in the recursion tree, for the 1st recursive is called on $n^{1/2}$, the 2nd recursive is called on $n^{1/4}$ ($\frac{1}{4}$ is 2^2) so if k is the last level then the k th recursive call is called on $n^{1/(2 \text{ power } k)}$. Since the termination condition is n at that level should be less than 2, we have the below equation.

$$n^{1/(2 \text{ power } k)} = 2$$

Solving the eqn by taking log as required as in the above figure, we will get $K = \lg \lg n$. Hence the total time complexity of the program is $\lg \lg n * \Theta(1)$ which is $\Theta(\lg \lg n)$ option D.

11.25 Solved Problem - 3

GATE 2006 | Big O, Theta, Omega notation | Data Structure & Algorithms

Watch later Share

Consider the following recurrence:

$$T(n) = 2T(\sqrt{n}) + 1, T(1) = 1$$

Which one of the following is true?

A. $T(n) = \Theta(\log \log n)$
B. $T(n) = \Theta(\log n)$
C. $T(n) = \Theta(\sqrt{n})$
D. $T(n) = \Theta(n)$

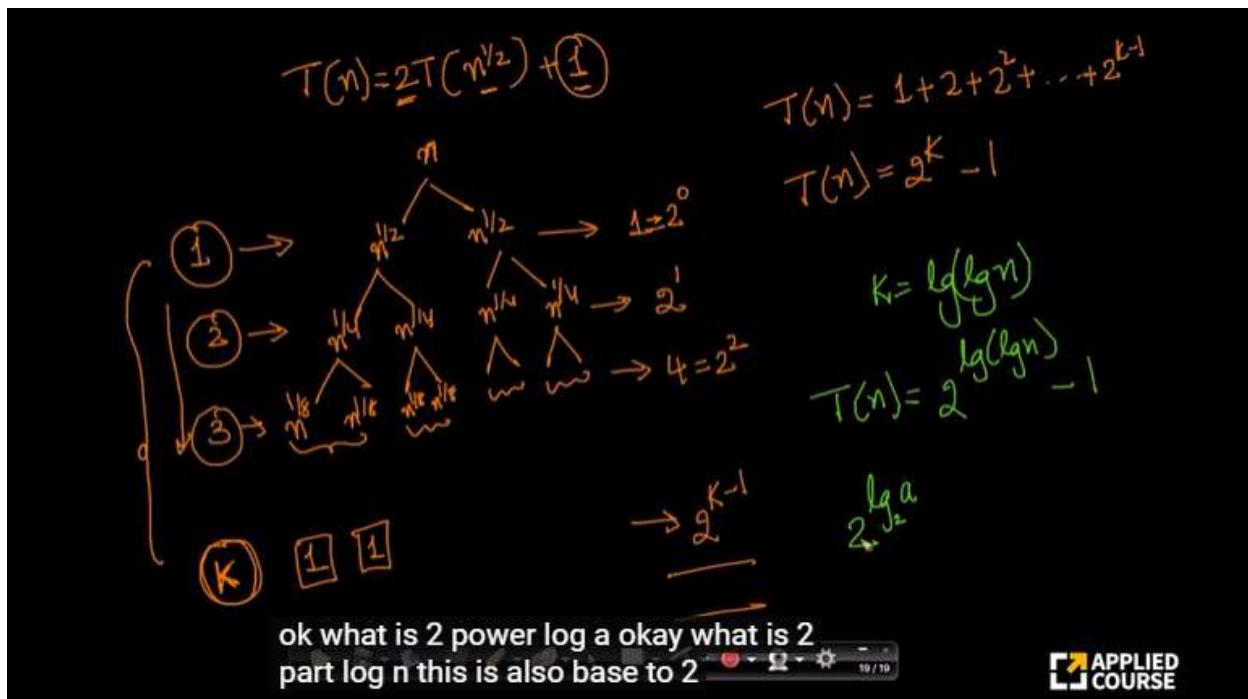
MORF VIDEOS
Play (k)

0:12 / 6:24

CC APPLIED COURSE

Timestamp: 0:12

In the question in the figure above notice that the base condition is when $n=1$. Using the given recursive equation we can come up with the recursion tree as below.



Timestamp: 5:28

As shown in the above figure at each level l we perform 2^{l-1} , due to 2^{l-1} recursive calls at each level l . Having solved the previous problem in 11.24 we know that number of levels k in such a recursion tree is $\lg \lg n$. Hence we have the time complexity $T(n) = 1+2+4+\dots 2^{k-1}$ as shown in the above figure.

We know the sum of such geometric series is $2^k - 1$. Here since $k = \lg \lg n$. The total time taken will be $\Theta(\lg \lg n)$ since $2^{\lg \lg n} = \lg n$ because $2^{\lg a} = a$.

Hence the correct option is b.

11.26 Solved Problem - 4

3. Four matrices M_1, M_2, M_3 and M_4 of dimensions $p \times q, q \times r, r \times s$ and $s \times t$ respectively can be multiplied in several ways with different number of total scalar multiplications. For example, when multiplied as $((M_1 \times M_2) \times (M_3 \times M_4))$, the total number of multiplications is $pqr + rst + prt$. When multiplied as $((M_1 \times M_2) \times M_3) \times M_4$, the total number of scalar multiplications is $pqr + prs + pst$.

If $p = 10, q = 100, r = 20, s = 5$ and $t = 80$, then the number of scalar multiplications needed is:

- (A) 248000
- (B) 44000
- (C) 19000
- (D) 25000

$M_1 M_2 M_3 M_4$

$$m[i,j] = \begin{cases} 0 & \text{if } i=j \\ \min_{1 \leq k \leq j-1} (m[i,k] + m[k+1,j] + p_{i,k} * p_{k,j} * p_{j,i}) & \text{if } i < j \end{cases}$$

Given:

$$p_0 = 10, p_1 = 100, p_2 = 20, p_3 = 5, p_4 = 80$$

$$m[1,1] = m[2,2] = m[3,3] = m[4,4] = 0 \quad (\text{base-case})$$

$$m[1,2] = p_0 + p_1 + p_2 = 10 \times 100 \times 20 = 20,000$$

$$m[2,3] = p_1 + p_2 + p_3 = 100 \times 20 \times 5 = 10,000$$

$$m[3,4] = p_2 + p_3 + p_4 = 20 \times 5 \times 80 = 8,000$$

 APPLIED COURSE

Timestamp: 0:19

This problem is an example of the matrix multiplication problem discussed in 11.20 hence it has the same recurrence equation as shown in the above figure.

As discussed in the matrix multiplication problem, we start with lesser chain lengths and increase the chain length to match the entire sequence.

So start with length 1 sequences, since we know that $m[i,i]=0$ since they are no multiplications required if we have only one matrix, $m[1,1]=m[2,2]=m[3,3]=m[4,4]=0$ which is the base case.

Next we will be dealing with length 2 sequences and compute $m[1,2]$, $m[2,3]$, $m[3,4]$ as below:

Given: $p_0 = 10, p_1 = 100, p_2 = 20, p_3 = 5, p_4 = 80$
 $m[i,j] = \begin{cases} 0 & \text{if } i=j \\ \min_{1 \leq k \leq j} (m[i,k] + m[k+1,j] + p_{i,k} + p_{k+1,j}) & \text{if } i < j \end{cases}$

(A) 24800
 (B) 44000
 (C) 19000
 (D) 25000

$m[1,1] = m[2,2] = m[3,3] = m[4,4] = 0$ (base case)

$m[1,2] = p_0 + p_1 + p_2 = 10 + 100 + 20 = 20,000$

$m[2,3] = p_1 + p_2 + p_3 = 100 + 20 + 5 = 10,000$

$m[3,4] = p_2 + p_3 + p_4 = 20 + 5 + 80 = 8,000$

$m[1,3] = \min \{ m[1,1] + m[2,3] + (10 + 100 + 5), m[1,2] + m[3,3] + (100 + 20 + 5) \}$

Timestamp: 3:05

Next we will be dealing with matrix chain of length 3, from here we have to choose the way of multiplying the sequences which reduces the number of total multiplications in that chain. Please take a look at the below figure.

$$m[3,4] = p_2 + p_3 + p_4 = \underline{20 \times 5 \times 80} = 8000$$

len(3)

$$m[1,3] = \min \left\{ \begin{array}{l} m[1,1] + m[2,3] + (10 + 100 \times 5), \\ m[1,2] + m[3,3] + (10 + 20 + 5) \end{array} \right\}$$

A₁A₂A₃

K=1,2

$$m[2,4] = \underline{15,000}$$

$$m[2,4] = \min \left\{ \begin{array}{l} m[2,2] + m[3,4] + (100 \times 20 + 80), \\ m[2,3] + m[4,4] + (100 + 5 \times 80) \end{array} \right\}$$

K=2,3

$$= \underline{50,000}$$

$$m[1,4] = \min \left\{ \begin{array}{l} m[1,1] + m[2,4] + (10 + 100 + 80), \\ m[1,2] + m[3,4] + (10 + 20 + 80), \\ \dots + (10 + 50 + 80) \end{array} \right\}$$

Notice in the above figure that we took the number of multiplications for computing $m[1,3]$ as the minimum of finding $m[1,3]$ by splitting into $m[1,2]$ and $m[2,3]$ and combining them or by splitting $m[1,3]$ by splitting into $m[1,1]$ and $m[1,3]$ and combining them.

$m[2,4] = \min \left\{ \frac{m[1,2] * m[2,3]}{m[2,3] + m[3,4] + (100 * 5 * 80)} \right\}$

$m[1,4] = \min \left\{ \begin{array}{l} m[1,1] * m[2,4] + (10 * 100 * 80), \\ m[1,2] * m[3,4] + (10 * 20 * 80), \\ m[1,3] * m[4,4] + (10 * 50 * 80) \end{array} \right\}$

$= 19,000$

Timestamp: 6:20

Now we finally compute chain of length 4, which will cover the entire sequence.

We choose $m[1,4]$ as the minimum number of multiplications required among splitting $m[1,4]$ into $m[1,1]$ and $m[2,4]$ and the cost of combining them or splitting $m[1,4]$ into $m[1,2]$ and $m[3,4]$ and the cost of combining them or splitting $m[1,4]$ into $m[1,3]$ and $m[4,4]$ and the cost of combining them.

We found it to be 19000 which is the minimum number of multiplications required to compute $M1 * M2 * M3 * M4$ hence the answer is option C.

11.27 Solved Problem - 5

4. The subset-sum problem is defined as follows. Given a set of n positive integers, $S = \{a_1, a_2, a_3, \dots, a_n\}$, and positive integer W , is there a subset of S whose elements sum to W ? A dynamic program for solving this problem uses a 2-dimensional Boolean array X , with n rows and $W+1$ columns. $X[i, j], 1 \leq i \leq n, 0 \leq j \leq W$, is TRUE if and only if there is a subset of $\{a_1, a_2, \dots, a_i\}$ whose elements sum to j . Which of the following is valid for $2 \leq i \leq n$ and $a_i \leq j \leq W$?

(A) $X[i, j] = X[i-1, j] \vee X[i, j-a_i]$

(B) $X[i, j] = X[i-1, j] \vee X[i-1, j-a_i]$

(C) $X[i, j] = X[i-1, j] \vee X[i, j-a_i]$

(D) $X[i, j] = X[i-1, j] \vee X[i-1, j-a_i]$

W: SUM
 $X(i, j)$
DISCRETE MATHS ✓ (OR)
 \wedge (AND)
SUBSET SUM PROBLEM (DP)

Timestamp: 4:38

The question in crisp is asking the recurrence relation of our subset sub problem with given sum as w .

The correct answer is option b in the below options.

A. $X[i, j] = X[i-1, j] \vee X[i, j-a_i]$

B. $X[i, j] = X[i-1, j] \vee X[i-1, j-a_i]$

C. $X[i, j] = X[i-1, j] \wedge X[i, j-a_i]$

D. $X[i, j] = X[i-1, j] \wedge X[i-1, j-a_i]$

Option B is correct since for computing $X[i, j]$ we check it as the or condition of $X[i-1, j]$ which signifies that in the end subset, element i is not present and $X[i-1, j-a_i]$ which signifies that element i is present in the end subset so we can just check whether we can get sum $j-a_i$ from the remaining $i-1$ elements.

11.28 Solved Problem - 6

(Q) $\begin{bmatrix} \text{int } j, n; \\ j=1; \\ \text{while } (j \leq n) \\ \quad j=j+2; \end{bmatrix}$

$n > 0$
How many comparisons are made in the while loop?

(a) $\lceil \log_2 n \rceil + 2 = \cancel{\infty}$ (b) $\cancel{n=4}$ (c) $\lceil \log_2 n \rceil = \cancel{\infty}$
 (d) $\lfloor \log_2 n \rfloor + 2 = 4$ (e) $n=5$ ✓
 $j=1, 2, 4, 8 \rightarrow \cancel{4}$

$\left\{ \begin{array}{ll} \lceil x \rceil : \text{ceil} & \lceil 2 \cdot 1 \rceil = 3 \\ \lfloor x \rfloor : \text{floor} & \lfloor 2 \cdot 1 \rfloor = 2 \end{array} \right.$
 $\frac{\log_2 4 = 2; \log_2 8 = 3}{\lceil \log_2 5 \rceil = \lceil 2 \dots \rceil = 3}$
 $\lfloor \log_2 5 \rfloor = \lfloor 2 \dots \rfloor = 2$



Timestamp: 4:20

The question is to find the number of comparison made in the while loop given in the above figure. Since the options are not asking for time complexity but for the exact answer. We will have to try it for an example n and try it out.

Suppose if we take n=5 then for j=1 it checks makes one comparison, for j=2 it makes the second comparison, for j=4 it makes the third comparison, for j=8 we make the fourth comparison and the condition fails hence there are no more comparison.s

Checking 4 against each options only option d satisfies, hence no of comparisons are as mentioned in the option d.

11.29 Solved Problem - 7

The image shows a Google Slides slide titled "Data Structures & Algorithm". The slide contains handwritten mathematical notes and a digital calculator.

Handwritten notes:

- $m = 64 \rightarrow 30 \text{ sec}$
- $c \cdot n \lg n = 30 \text{ if } n=64$
- $c + 64 \lg 64 = 30 \Rightarrow c + 64 \cdot 6 = 30 \Rightarrow c = 30/(64 \cdot 6)$
- $c + m \lg m = 360$
- $\frac{30}{64 \cdot 6} + m \lg m = 360$
- $m \lg m = \frac{360}{30} = 12$
- $12 = \frac{m \lg m}{m \lg m} = \frac{m \lg m}{m \lg m} = 4608$

Text box:

Assume that a mergesort algorithm in the worst case takes 30 seconds for an input of size 64. Which of the following most closely approximates the maximum input size of a problem that can be solved in 6 minutes?

Calculator screen:

4608

C	%	%	/
7	8	9	x
4	5	6	-
1	2	3	+

APPLIED COURSE

Timestamp: 4:01

In the question in the above figure we have to find out how much is the input size that can be sorted in 6 minutes using merge sort, given that we were able to solve an input size of 64 using merge sort in 30 seconds.

We know that merge sort takes $c \cdot n \lg n$ time for some constant c , since we were given that for input size 64 we took that 30 seconds, so solving $c \cdot 64 \lg 64 = 30$ should give us $c = 30/(64 \cdot 6)$.

Now let us say the required max input size is m , then given it takes time 6 minutes = $6 \cdot 60 = 360$ seconds. We know that $c \cdot m \lg m = 360 \rightarrow 30/(64 \cdot 6) \cdot m \lg m = 360$, solving this by substituting the given options should give the required size of 512. Hence the answer is option B.

12.1 What are Modules in Python

- Modules refer to a file containing Python statements and definitions.
- A file containing Python code, for e.g.: abc.py, is called a module and its module name would be "abc".
- We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code.
- We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

Examples:

```
[ ] import math  
print(math.pi)
```

```
3.141592653589793
```

```
[ ] import datetime  
datetime.datetime.now()
```

```
datetime.datetime(2017, 10, 18, 20, 47, 20, 606228)
```

import with renaming



```
import math as m  
print(m.pi)
```



```
3.141592653589793
```

from...import statement

We can import specific names from a module without importing the module as a whole.

```
[ ] from datetime import datetime  
datetime.now()  
  
datetime.datetime(2017, 10, 18, 20, 47, 38, 17242)
```

▼ import all names

```
[ ] from math import *  
print("Value of PI is " + str(pi))  
  
Value of PI is 3.141592653589793
```

- math and datetime are examples of modules in python.
- We can also import a function by renaming as shown above.
- Instead of importing whole module we can import functions which are needed as shown below

- **dir() built in function**

We can use the dir() function to find out names that are defined inside a module.

```
[ ] dir(example)
```

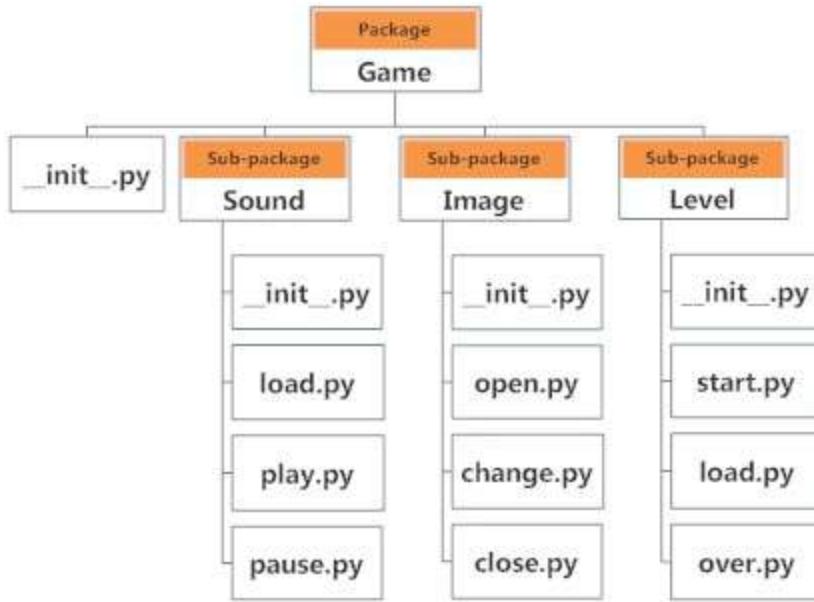
```
['__builtins__',  
 '__cached__',  
 '__doc__',  
 '__file__',  
 '__loader__',  
 '__name__',  
 '__package__',  
 '__spec__',  
 'add']
```

```
[ ] print(example.add.__doc__)
```

This program adds two numbers and return the result

- The dir() function returns all properties and methods of the specified object, without the values.
- This function will return all the properties and methods, even built-in properties which are default for all objects.

- Packages are a way of structuring Python's module namespace by using "dotted module names".
- A directory must contain a file named **init.py** in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.



- Intuitively we can think of packages as folders in python and modules are .py files. For python to consider a folder as a package it needs to have `__init__.py` file in it.

importing module from a package

We can import modules from packages using the dot (.) operator.

```
[3] #import Gate.Image.open
```

- We can import a module from the package using . notation.

12.4 Introduction to NumPy

NumPy Arrays

python objects:

1. high-level number objects: integers, floating point

2. containers: lists (costless insertion and append), dictionaries (fast lookup)

Numpy provides:

1. extension package to Python for multi-dimensional arrays
2. closer to hardware (efficiency)
3. designed for scientific computation (convenience)
4. Also known as array oriented computing

1. Creating arrays

```
[ ] 1 import numpy as np  
2 a = np.array([0, 1, 2, 3])  
3 print(a)  
4  
5 print(np.arange(10))
```

```
[0 1 2 3]  
[0 1 2 3 4 5 6 7 8 9]
```

Why it is useful: Memory-efficient container that provides fast numerical operations.

```
[ ] 1 #python lists  
2 L = range(1000)  
3 %timeit [i**2 for i in L]
```

```
307 µs ± 17.6 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

```
[ ] 1 a = np.arange(1000)  
2 %timeit a**2
```

```
1.35 µs ± 126 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

```
1 #1-D  
2  
3 a = np.array([0, 1, 2, 3])  
4  
5 a
```

```
array([0, 1, 2, 3])
```

```
1 # 2-D, 3-D....  
2  
3 b = np.array([[0, 1, 2], [3, 4, 5]])  
4  
5 b
```

```
array([[0, 1, 2],  
       [3, 4, 5]])
```

```
] 1 c = np.array([[[], [0, 1], [2, 3]], [[4, 5], [6, 7]]])  
2  
3 c
```

```
array([[[0, 1],  
       [2, 3]],
```

```
      [[4, 5],  
       [6, 7]]])
```

- We can create arrays in numpy as shown above.

1.2 Functions for creating arrays

```
] 1 #using arange function
2
3 # arange is an array-valued version of the built-in Python range function
4
5 a = np.arange(10) # 0.... n-1
6 a
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
] 1 b = np.arange(1, 10, 2) #start, end (exclusive), step
2
3 b
```

```
array([1, 3, 5, 7, 9])
```

```
1 #using linspace
2
3 a = np.linspace(0, 1, 6) #start, end, number of points
4
5 a
```

```
array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ])
```

```
1 #common arrays
2
3 a = np.ones((3, 3))
4
5 a
```

```
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

```
1 b = np.zeros((3, 3))
2
3 b
```

```
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

```
1 #create array using diag function
2
3 a = np.diag([1, 2, 3, 4]) #construct a diagonal array.
4
5 a

array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])

1 np.diag(a)  #Extract diagonal

array([1, 2, 3, 4])

1 #create array using random
2
3 #Create an array of the given shape and populate it with random samples from a uniform distribution over [0, 1].
4 a = np.random.rand(4)
5
6 a

array([ 0.85434586,  0.05106692,  0.37337949,  0.32093548])
```

- We can use functions for creating numpy arrays as shown above.

2. Basic DataTypes

You may have noticed that, in some instances, array elements are displayed with a **trailing dot (e.g. 2. vs 2)**. This is due to a difference in the **data-type** used:

```
1 a = np.arange(10)
2 a.dtype
```

```
dtype('int64')
```

```
1 #You can explicitly specify which data-type you want:
2 a = np.arange(10, dtype='float64')
3 a
```

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.])
```

```
1 #The default data type is float for zeros and ones function
2 a = np.zeros((3, 3))
3 print(a)
4 a.dtype
```

```
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]
dtype('float64')
```

Each built-in data type has a character code that uniquely identifies it.

'b' – boolean

'i' – (signed) integer

'u' – unsigned integer

'f' – floating-point

'c' – complex-floating point

'm' – timedelta

'M' – datetime

'O' – (Python) objects

'S', 'a' – (byte-)string

'U' – Unicode

'V' – raw data (void)

3. Indexing and Slicing

The items of an array can be accessed and assigned to the same way as other **Python sequences (e.g. lists)**:

```
[ ] 1 a = np.arange(10)
2 print(a[5]) #indices begin at 0, like other Python sequences (and C/C++)
```

5

```
[ ] 1 # For multidimensional arrays, indexes are tuples of integers:
2 a = np.diag([1, 2, 3])
3 print(a[2, 2])
```

3

```
▶ 1 a[2, 1] = 5 #assigning value
2 a
```

```
array([[1, 0, 0],
       [0, 2, 0],
       [0, 5, 3]])
```

```
] 1 a = np.arange(10)
2 a
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
] 1 a[1:8:2] # [startindex: endindex(exclusive) : step]
```

```
array([1, 3, 5, 7])
```

```
] 1 #we can also combine assignment and slicing:
2 a = np.arange(10)
3 a[5:] = 10
4 a
```

```
array([ 0,  1,  2,  3,  4, 10, 10, 10, 10, 10])
```

```
▶ 1 b = np.arange(5)
2 a[5:] = b[::-1] #assigning
3 a
```

```
array([0, 1, 2, 3, 4, 4, 3, 2, 1, 0])
```

- A slicing operation creates a view on the original array, which is just a way of accessing array data. Thus the original array is not copied in memory. You can use `np.may_share_memory()` to check if two arrays share the same memory block.

4. Copies and Views

When modifying the view, the original array is modified as well:

```
[ ] 1 a = np.arange(10)
2 a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

[ ] 1 b = a[::2]
2 b
array([0, 2, 4, 6, 8])

[ ] 1 np.shares_memory(a, b)
2
True

[ ] 1 b[0] = 10
2 b
array([10, 2, 4, 6, 8])

[ ] 1 a #eventhough we modified b, it updated 'a' because both shares same memory
2
array([10, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

- When we copy a new array gets created ,so changes made to the copied array don't affect the original array.

```
2
3 a = np.arange(10)
4
5 c = a[::2].copy()      #force a copy
6 c

array([0, 2, 4, 6, 8])

] 1 np.shares_memory(a, c)
2
False

] 1 c[0] = 10
2
3 a

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

5. Fancy Indexing

NumPy arrays can be indexed with slices, but also with boolean or integer arrays (**masks**). This method is called **fancy indexing**. It creates copies not views.

Using Boolean Mask

```
[ ] 1 a = np.random.randint(0, 20, 15)
2 a
array([18, 17,  1, 18,  5, 17,  0, 14, 12, 11,  4, 15, 16,  8,  7])
```

```
[ ] 1 mask = (a % 2 == 0)
```

```
[ ] 1 extract_from_a = a[mask]
2
3 extract_from_a
```

```
array([18, 18,  0, 14, 12,  4, 16,  8])
```

Indexing with a mask can be very useful to assign a new value to a sub-array:

```
[ ] 1 a[mask] = -1
2 a
array([-1, 17,  1, -1,  5, 17, -1, -1, -1, 11, -1, 15, -1, -1,  7])
```

Indexing with an array of integers

```
[ ] 1 a = np.arange(0, 100, 10)
2 a
array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90])
```

```
[ ] 1 #Indexing can be done with an array of integers, where the same index is repeated several time:
2 a[[2, 3, 2, 4, 2]]
array([20, 30, 20, 40, 20])
```

```
▶ 1 # New values can be assigned
2 a[[9, 7]] = -200
3 a
array([ 0, 10, 20, 30, 40, 50, 60, -200, 80, -200])
```

12.5 Common Operations in NumPy

1. Elementwise Operations

1. Basic Operations

with scalars

```
[ ] 1 a = np.array([1, 2, 3, 4]) #create an array  
2  
3 a + 1
```

```
array([2, 3, 4, 5])
```

```
[ ] 1 a ** 2
```

```
array([ 1,  4,  9, 16])
```

All arithmetic operates elementwise

```
[ ] 1 b = np.ones(4) + 1  
2  
3 a - b
```

```
array([-1.,  0.,  1.,  2.])
```

```
[ ] 1 a * b
```

```
array([ 2.,  4.,  6.,  8.])
```

```
▶ 1 # Matrix multiplication  
2  
3 c = np.diag([1, 2, 3, 4])  
4  
5 print(c * c)  
6 print("*****")  
7 print(c.dot(c))
```

👤 [[1 0 0 0]
 [0 4 0 0]
 [0 0 9 0]
 [0 0 0 16]]

[[1 0 0 0]
 [0 4 0 0]
 [0 0 9 0]
 [0 0 0 16]]

- Below are the comparison operations that we can perform using numpy

comparisons

```
[ ] 1 a = np.array([1, 2, 3, 4])
2 b = np.array([5, 2, 2, 4])
3 a == b
```

array([False, True, False, True], dtype=bool)

```
[ ] 1 a > b
```

array([False, False, True, False], dtype=bool)

```
[ ] 1 #array-wise comparisons
2 a = np.array([1, 2, 3, 4])
3 b = np.array([5, 2, 2, 4])
4 c = np.array([1, 2, 3, 4])
5
6 np.array_equal(a, b)
```

False

```
[ ] 1 np.array_equal(a, c)
```

True

Below are the logical operations that we can perform using numpy

Logical Operations

```
[ ] 1 a = np.array([1, 1, 0, 0], dtype=bool)
2 b = np.array([1, 0, 1, 0], dtype=bool)
3
4 np.logical_or(a, b)
```

array([True, True, True, False], dtype=bool)

```
[ ] 1 np.logical_and(a, b)
```

array([True, False, False, False], dtype=bool)

Transcendental functions:

```
1 a = np.arange(5)
2
3 np.sin(a)

[ ] array([ 0.           ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ])

[ ] 1 np.log(a)

/Users/satishatcha/.virtualenvs/course/lib/python2.7/site-packages/ipykernel_launcher.py
    """Entry point for launching an IPython kernel.
array([-inf,  0.       ,  0.69314718,  1.09861229,  1.38629436])

[ ] 1 np.exp(a)  #evaluates e^x for each element in a given input

array([ 1.           ,  2.71828183,  7.3890561 ,  20.08553692,  54.59815003])
```

2.Basic Reductions

Below are the basic reductions we can do using numpy.

computing sums

```
[ ] 1 x = np.array([1, 2, 3, 4])
2 np.sum(x)
```

10

```
[ ] 1 #sum by rows and by columns
2
3 x = np.array([[1, 1], [2, 2]])
4 x
```

array([[1, 1],
 [2, 2]])

```
[ ] 1 x.sum(axis=0)  #columns first dimension
array([3, 3])
```

```
[ ] 1 x.sum(axis=1)  #rows (second dimension)
array([2, 4])
```

Other reductions

```
[ ] 1 x = np.array([1, 3, 2])
2 x.min()
```

1

```
[ ] 1 x.max()
```

3

```
[ ] 1 x.argmin()# index of minimum element
```

0

```
[ ] 1 x.argmax()# index of maximum element
```

1

Statistics

```
[ ] 1 x = np.array([1, 2, 3, 1])
2 y = np.array([[1, 2, 3], [5, 6, 1]])
3 x.mean()
```

```
1.75
```

```
[ ] 1 np.median(x)
```

```
1.5
```

```
[ ] 1 np.median(y, axis=-1) # last axis
```

```
array([ 2.,  5.])
```

```
[ ] 1 x.std()           # full population standard dev.
```

```
0.82915619758884995
```

3.Broadcasting

- Basic operations on numpy arrays (addition, etc.) are elementwise
- This works on arrays of the same size. Nevertheless, it's also possible to do operations on arrays of different sizes if NumPy can transform these arrays so that they all have the same size: this conversion is called broadcasting.
- The image below gives an example of broadcasting:

$$\begin{array}{c}
 \begin{array}{ccccc}
 \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 10 & 10 & 10 \\ \hline 20 & 20 & 20 \\ \hline 30 & 30 & 30 \\ \hline \end{array} & + & \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 10 & 10 & 10 \\ \hline 20 & 20 & 20 \\ \hline 30 & 30 & 30 \\ \hline \end{array} & + & \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \end{array} & = & \\
 \end{array} \\
 \begin{array}{ccccc}
 \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 10 & 10 & 10 \\ \hline 20 & 20 & 20 \\ \hline 30 & 30 & 30 \\ \hline \end{array} & + & \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 10 & 10 & 10 \\ \hline 20 & 20 & 20 \\ \hline 30 & 30 & 30 \\ \hline \end{array} & + & \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline 10 & 11 & 12 \\ \hline 20 & 21 & 22 \\ \hline 30 & 31 & 32 \\ \hline \end{array} \\
 \end{array} \\
 \begin{array}{ccccc}
 \begin{array}{|c|} \hline 0 \\ \hline 10 \\ \hline 20 \\ \hline 30 \\ \hline \end{array} & + & \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \end{array} & = & \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 10 & 10 & 10 \\ \hline 20 & 20 & 20 \\ \hline 30 & 30 & 30 \\ \hline \end{array} & + & \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline \end{array} & = & \\
 \end{array}
 \end{array}$$

```
1 a = np.tile(np.arange(0, 40, 10), (3,1))
2 print(a)
3
4 print("*****")
5 a=a.T
6 print(a)
```

```
[[ 0 10 20 30]
 [ 0 10 20 30]
 [ 0 10 20 30]]
```

```
*****  
[[ 0  0  0]
 [10 10 10]
 [20 20 20]
 [30 30 30]]
```

+ Co

```
1
2 b = np.array([0, 1, 2])
3 b
```

```
array([0, 1, 2])
```

```
1
2 a + b
```

```
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22],
       [30, 31, 32]])
```

```
[1] 1 a = np.arange(0, 40, 10)
[2] 2 a.shape
[3]
```

```
(4,)
```

```
[1] 1 a = a[:, np.newaxis] # adds a new axis -> 2D array
[2] 2 a.shape
```

```
(4, 1)
```

```
[1]
```

```
1 a
array([[ 0],
       [10],
       [20],
       [30]])
```

```
[1]
```

```
1 a + b
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22],
       [30, 31, 32]])
```

4.Array Shape Manipulation

1.Flattening

```
[ ] 1 a = np.array([[1, 2, 3], [4, 5, 6]])  
2 a.ravel() #Return a contiguous flattened array.
```

```
array([1, 2, 3, 4, 5, 6])
```

```
1 a.T #Transpose
```

```
array([[1, 4],  
       [2, 5],  
       [3, 6]])
```

```
[ ] 1 a.T.ravel()
```

```
array([1, 4, 2, 5, 3, 6])
```

2.Reshaping

The inverse operation to flattening:

```
[ ] 1 print(a.shape)
2 print(a)
```

```
(2, 3)
[[1 2 3]
 [4 5 6]]
```

```
[ ] 1 b = a.ravel()
2 print(b)
```

```
[1 2 3 4 5 6]
```

```
[ ] 1 b = b.reshape((2, 3))
2 b
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

3.Adding a Dimension

- Indexing with the np.newaxis object allows us to add an axis to an array
- newaxis is used to increase the dimension of the existing array by one more dimension, when used once. Thus,

1D array will become 2D array

2D array will become 3D array

3D array will become 4D array and so on

```
1 z = np.array([1, 2, 3])
2 z
```

```
array([1, 2, 3])
```

```
1 z[:, np.newaxis]
```

```
array([[1],
       [2],
       [3]])
```

4.Dimension Shuffling

```
[ ] 1 a = np.arange(4*3*2).reshape(4, 3, 2)
2 a.shape
```

```
(4, 3, 2)
```

```
[ ] 1 a
```

```
array([[[ 0,  1],
       [ 2,  3],
       [ 4,  5]],

      [[[ 6,  7],
        [ 8,  9],
        [10, 11]],

       [[12, 13],
        [14, 15],
        [16, 17]]],

      [[[18, 19],
        [20, 21],
        [22, 23]]])
```



```
1 a[0, 2, 1]
```



```
5
```

5.Resizing

```
[ ] 1 a = np.arange(4)
2 a.resize((8,))
3 a

array([0, 1, 2, 3, 0, 0, 0, 0])
```

However, it must not be referred to somewhere else:

```
[ ] 1 b = a
2 a.resize((4,))

-----
ValueError                                     Traceback (most recent call last)
<ipython-input-68-702766c88583> in <module>()
      1 b = a
----> 2 a.resize((4,))

ValueError: cannot resize an array that references or is referenced
by another array in this way.  Use the resize function
```

[SEARCH STACK OVERFLOW](#)

6.Sorting Data

```
] 1 #Sorting along an axis:  
2 a = np.array([[5, 4, 6], [2, 3, 2]])  
3 b = np.sort(a, axis=1)  
4 b
```

```
array([[4, 5, 6],  
       [2, 2, 3]])
```

```
] 1 #in-place sort  
2 a.sort(axis=1)  
3 a
```

```
array([[4, 5, 6],  
       [2, 2, 3]])
```

```
] 1 #sorting with fancy indexing  
2 a = np.array([4, 3, 1, 2])  
3 j = np.argsort(a)  
4 j
```

```
array([2, 3, 1, 0])
```

```
] 1 a[j]
```

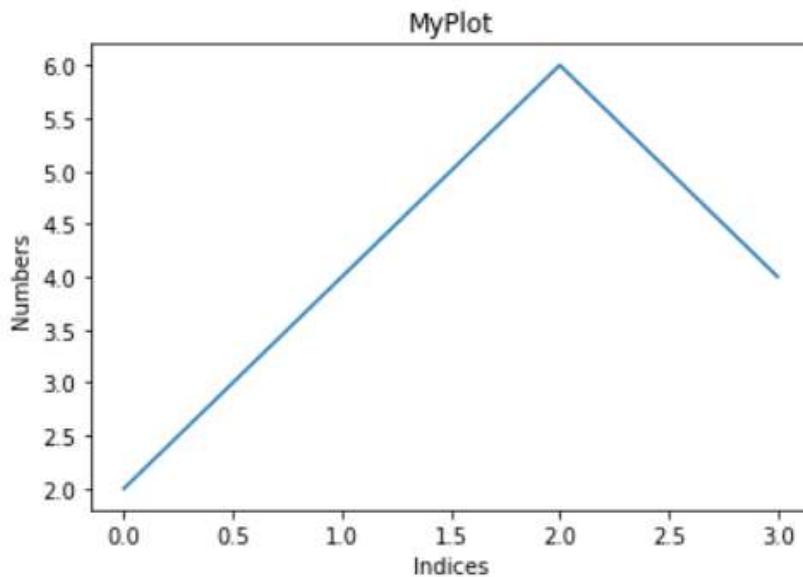
```
array([1, 2, 3, 4])
```

12.6 Getting started with Matplotlib

1.Plotting

- **matplotlib.pyplot** is a collection of command style functions that make matplotlib work like MATLAB.
- Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

```
plt.plot([2,4, 6, 4])
plt.ylabel("Numbers")
plt.xlabel('Indices')
plt.title('MyPlot')
plt.show()
```

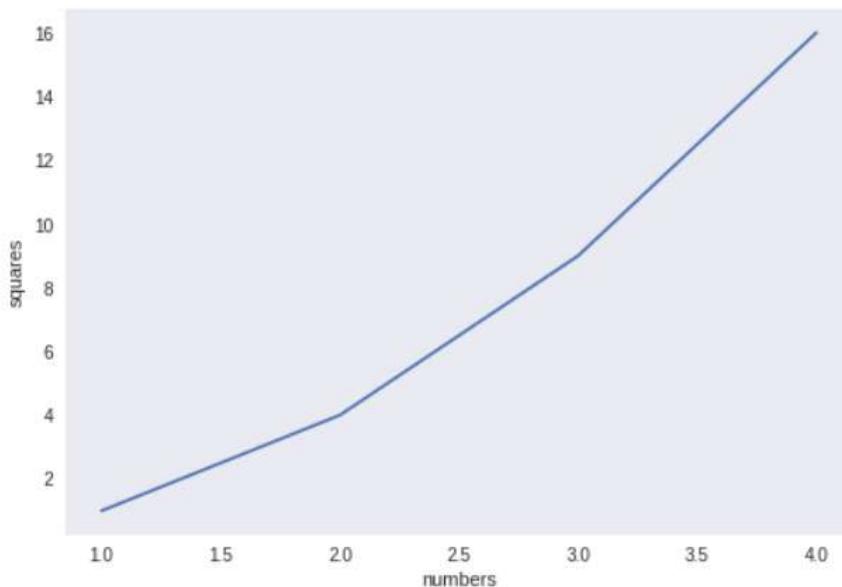


- If you provide a single list or array to the `plot()` command, matplotlib assumes it is a sequence of y values, and automatically generates the x values for you. Since python ranges start with 0, the default x vector has the same length as y but starts with 0. Hence the x data are [0,1,2,3].

plot x versus y

```
[ ] plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.ylabel('squares')
plt.xlabel('numbers')
plt.grid() # grid on

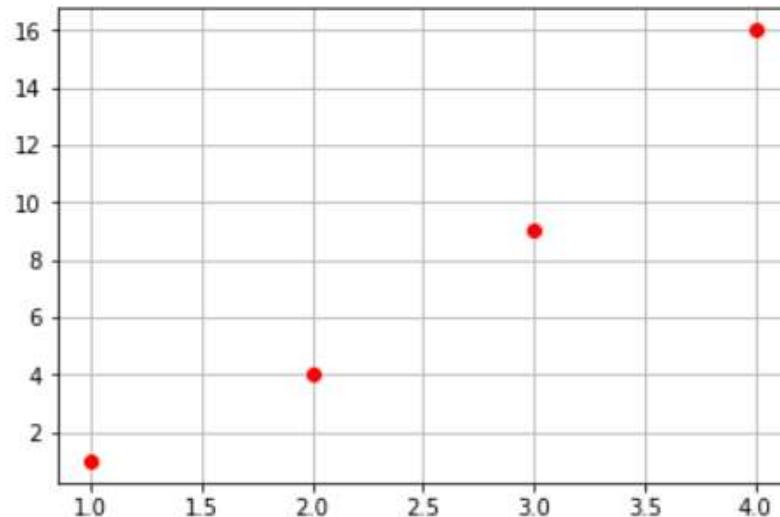
plt.show()
```



- For every x, y pair of arguments, there is an optional third argument as shown below which is the **format string** that indicates the color and line type of the plot.

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.grid()

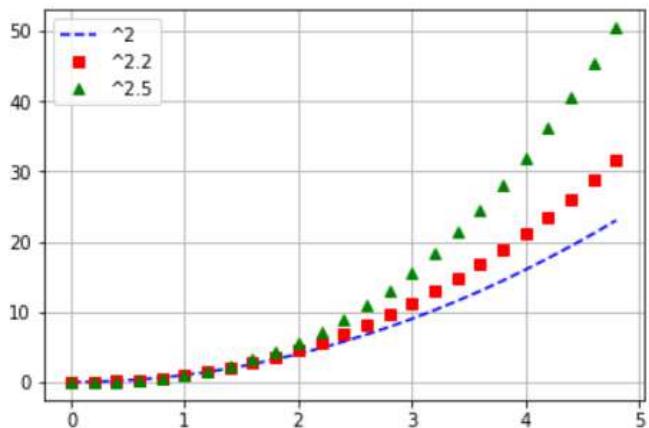
plt.show()
```



- If matplotlib were limited to working with lists, it would be fairly useless for numeric processing. Generally, you will use **numpy arrays**. In fact, all sequences are converted to numpy arrays internally.

```
import numpy as np
t = np.arange(0., 5., 0.2)

#blue dashes, red squares and green triangles
plt.plot(t, t**2, 'b--', label='^2')#    'rs',    'g^')
plt.plot(t,t**2.2, 'rs', label='^2.2')
plt.plot(t, t**2.5, 'g^', label='^2.5')
plt.grid()
plt.legend() # add legend based on line labels
plt.show()
```



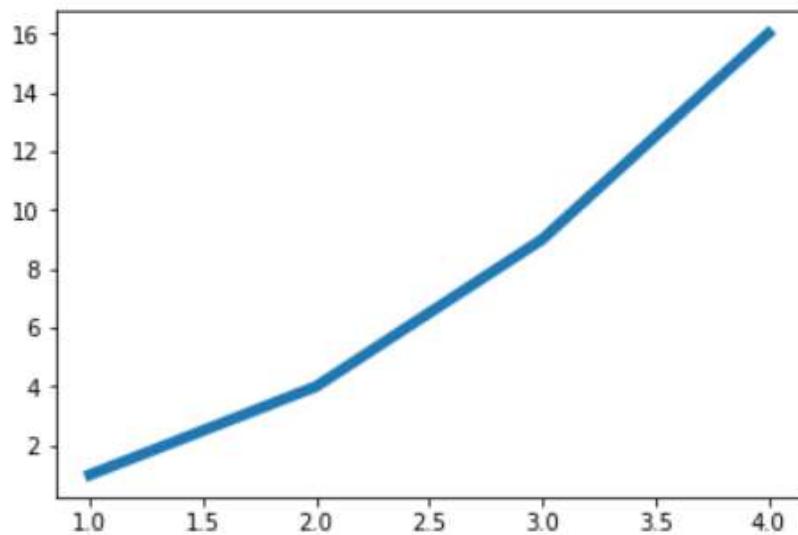
2. Controlling line properties

Lines have many attributes that you can set: linewidth, dash style etc

use keyword args



```
x = [1, 2, 3, 4]  
y = [1, 4, 9, 16]  
plt.plot(x, y, linewidth=5.0)  
plt.show()
```



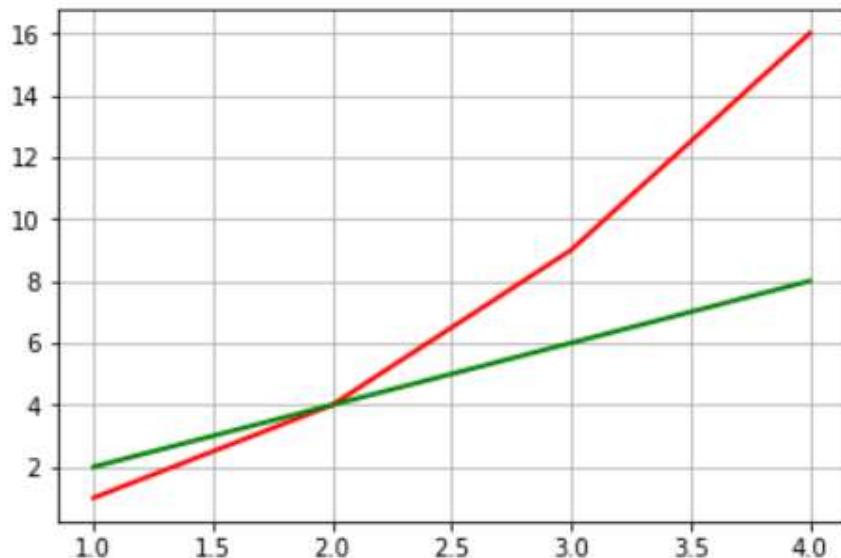
use the setp()

```
▶ x1 = [1, 2, 3, 4]
y1 = [1, 4, 9, 16]
x2 = [1, 2, 3, 4]
y2 = [2, 4, 6, 8]
lines = plt.plot(x1, y1, x2, y2)

# use keyword args
plt.setp(lines[0], color='r', linewidth=2.0)

# or MATLAB style string value pairs
plt.setp(lines[1], 'color', 'g', 'linewidth', 2.0)

plt.grid()
```



3 .Working with multiple figures and axes

```

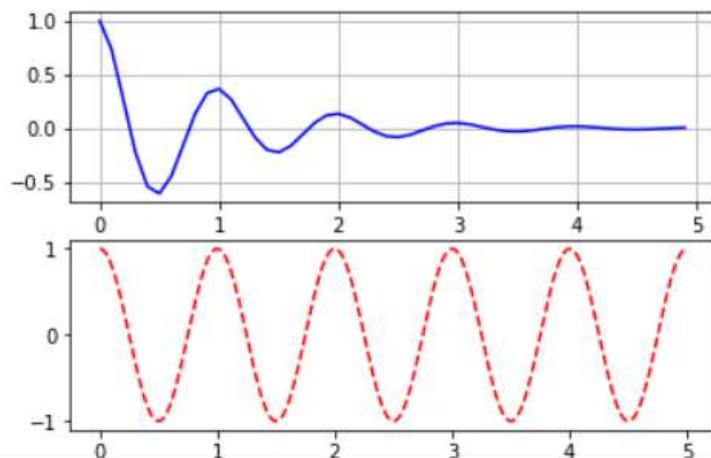
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
# The subplot() command specifies numrows, numcols,
# fignum where fignum ranges from 1 to numrows*numcols.
plt.subplot(211)
plt.grid()
plt.plot(t1, f(t1), 'b-')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()

```



`matplotlib.pyplot .subplots` creates a figure and a grid of subplots with a single call, while providing reasonable control over how the individual plots are created.

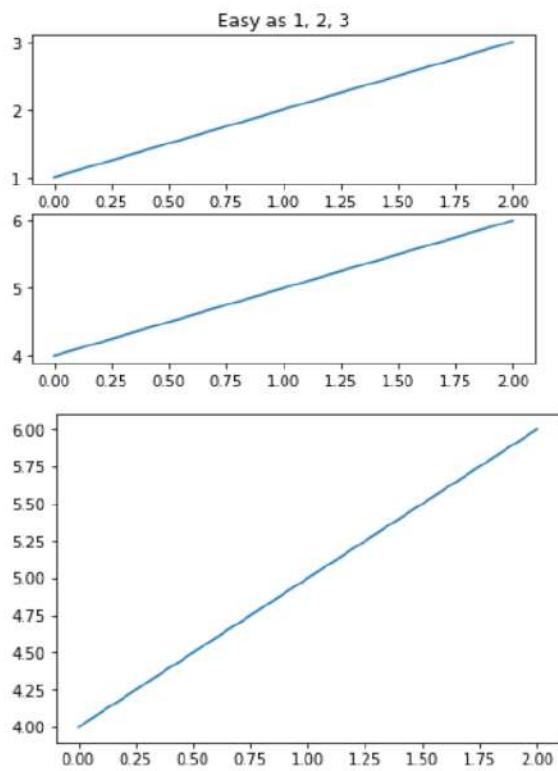
```

plt.figure(1)                      # the first figure
plt.subplot(211)                   # the first subplot in the first figure
plt.plot([1, 2, 3])
plt.subplot(212)                   # the second subplot in the first figure
plt.plot([4, 5, 6])

plt.figure(2)                      # a second figure
plt.plot([4, 5, 6])                # creates a subplot(111) by default

plt.figure(1)                      # figure 1 current; subplot(212) still current
plt.subplot(211)                   # make subplot(211) in figure1 current
plt.title('Easy as 1, 2, 3')        # subplot 211 title
plt.show()

```



12.7 Getting started with Pandas

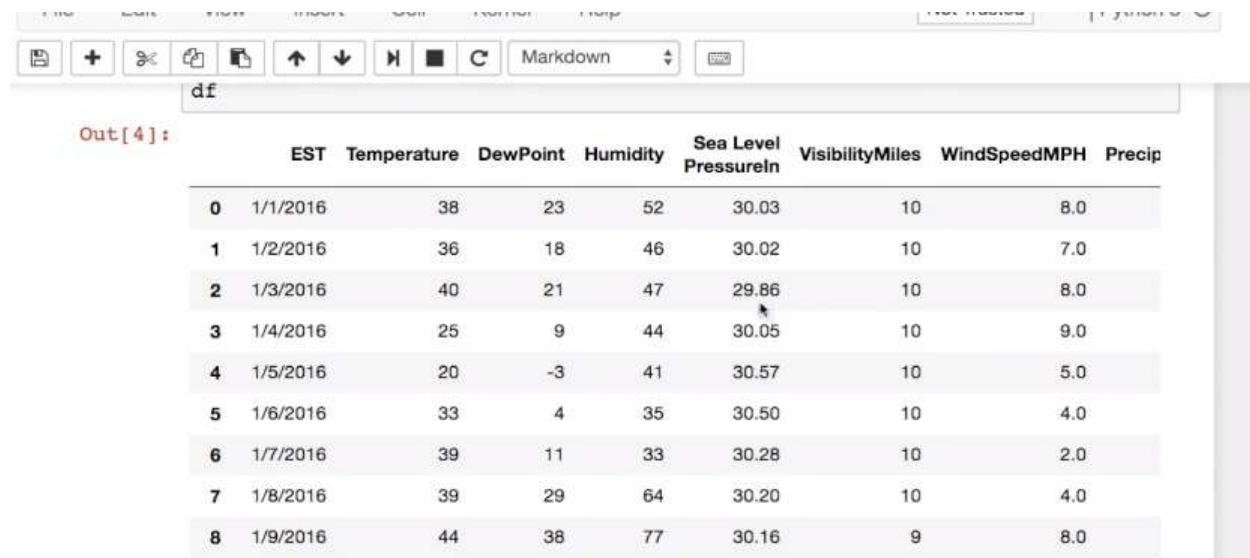
What is pandas-python? Introduction and Installation

- Pandas is python module that makes data science easy and effective
- Weather dataset

Questions?

1. What was the maximum temperature in new york in the month of january?
2. On which days did it rain?
3. What was the average speed of wind during the month?

The data frame shown below is the data at hand and we try to answer the above using the data.



The screenshot shows a Jupyter Notebook interface with a toolbar at the top containing various icons for file operations, cell selection, and cell type. Below the toolbar, the code cell is labeled "Out[4]:" and contains the variable name "df". The data is presented as a Pandas DataFrame with the following structure:

	EST	Temperature	DewPoint	Humidity	Sea Level Pressurein	VisibilityMiles	WindSpeedMPH	Precip
0	1/1/2016	38	23	52	30.03	10	8.0	
1	1/2/2016	36	18	46	30.02	10	7.0	
2	1/3/2016	40	21	47	29.86	10	8.0	
3	1/4/2016	25	9	44	30.05	10	9.0	
4	1/5/2016	20	-3	41	30.57	10	5.0	
5	1/6/2016	33	4	35	30.50	10	4.0	
6	1/7/2016	39	11	33	30.28	10	2.0	
7	1/8/2016	39	29	64	30.20	10	4.0	
8	1/9/2016	44	38	77	30.16	9	8.0	

```
In [4]: # now we will see in pandas  
  
import pandas as pd  
df = pd.read_csv('nyc_weather.csv')  
df
```

We read csv files using pandas as shown.

```
In [6]: #get the maximum temparature  
df['Temperature'].max()  
  
Out[6]: 50
```

Using max function we can obtain maximum value from a column in pandas dataframe

```
In [7]: #to know which day it rains  
df['EST'][df['Events'] == 'Rain']  
  
Out[7]: 8      1/9/2016  
9      1/10/2016  
15     1/16/2016  
26     1/27/2016  
Name: EST, dtype: object
```

We used a boolean expression which evaluates to be true only for those rows where the Events column has Rain. Finally the EST column will be returned.

```
In [8]: #3. average wind speed  
df['WindSpeedMPH'].mean()  
  
Out[8]: 6.8928571428571432
```

We can use the mean() function on the WindSpeedMPH column in the dataframe and get average speed.

Installation

```
pip3 install pandas
```

For installing pandas we use above command.

12.8 Understanding DataFrames in Pandas

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Introduction to Pandas dataframe

Data frame is a main object in pandas. It is used to represent data with rows and columns

Data frame is a data structure represent the data in tabular or excel spreadsheet like data)

creating dataframe:



```
1 import pandas as pd  
2 df = pd.read_csv("weather_data.csv")    #read weather.csv data  
3 df
```

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain
5	1/6/2017	31	2	Sunny

We use pandas to load our data into a dataframe.

```
1 #list of tuples
2
3 weather_data = [('1/1/2017', 32, 6, 'Rain'),
4                  ('1/2/2017', 35, 7, 'Sunny'),
5                  ('1/3/2017', 28, 2, 'Snow'),
6                  ('1/4/2017', 24, 7, 'Snow'),
7                  ('1/5/2017', 32, 4, 'Rain'),
8                  ('1/6/2017', 31, 2, 'Sunny')
9                 ]
10 df = pd.DataFrame(weather_data, columns=['day', 'temperature', 'windspeed', 'event'])
11 df
```

	day	temp	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain
5	1/6/2017	31	2	Sunny

Using list of tuples we can create pandas dataframe as shown above

```
1 #get dimensions of the table  
2  
3 df.shape #total number of rows and columns  
  
(6, 4)
```

```
1 #if you want to see initial some rows then use head command (default 5 rows)  
2 df.head()
```

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain

```
1 #if you want to see last few rows then use tail command (default last 5 rows will print)  
2 df.tail()
```

	day	temperature	windspeed	event
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow

- Using the shape attribute we can find numbers of rows and columns in the dataframe.
- head() gives top five rows in a dataframe, similarly tail() gives bottom 5 rows of the dataframe

```
] 1 #slicing  
2 df[2:5]
```

	day	temperature	windspeed	event
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain

```
] 1 df.columns #print columns in a table
```

```
Index(['day', 'temperature', 'windspeed', 'event'], dtype='object')
```

- We can use slicing and obtain required rows in the dataframe.
- Columns attribute gives the column names of the dataframe

```
1 df.day      #print particular column data

0    1/1/2017
1    1/2/2017
2    1/3/2017
3    1/4/2017
4    1/5/2017
5    1/6/2017
Name: day, dtype: object
```

```
1 #another way of accessing column
2 df['day'] #df.day (both are same)

0    1/1/2017
1    1/2/2017
2    1/3/2017
3    1/4/2017
4    1/5/2017
5    1/6/2017
Name: day, dtype: object
```

```
1 #get 2 or more columns
2 df[['day', 'event']]
```

	day	event
0	1/1/2017	Rain
1	1/2/2017	Sunny
2	1/3/2017	Snow

- We can get particular column data or 2 or more columns as shown above.

```
1 #print max temperature  
2 df['temperature'].max()
```

35

```
1 #print max temperature  
2 df['temperature'].min()
```



24

```
[ ] 1 #print max temperature  
2 df['temperature'].describe()
```

```
count      6.000000  
mean      30.333333  
std       3.829708  
min      24.000000  
25%      28.750000  
50%      31.500000  
75%      32.000000  
max      35.000000  
Name: temperature, dtype: float64
```

We can use functions like min(),max() on a particular column of dataframe to obtain min or max value.Alternatively we can use describe() function which gives us overall statistics of a particular column.

```
1 # select rows which has maximum temperature
2 df[df.temperature == df.temperature.max()]
3
```

	day	temperature	windspeed	event
1	1/2/2017	35	7	Sunny

```
1 #select only day column which has maximum temperature
2 df.day[df.temperature == df.temperature.max()]
3
```

```
1    1/2/2017
Name: day, dtype: object
```

We can also select particular rows based on the conditions we specify as shown above.

12.9 Common Operations on Data Frames

1. Read and write CSV and XLS files

```
In [3]: import pandas as pd
df = pd.read_csv('weather_data.csv')
df
```

```
Out[3]:      day  temperature  windspeed  event
0   1/1/2017        32          6   Rain
1   1/2/2017        35          7  Sunny
2   1/3/2017        28          2   Snow
3   1/4/2017        24          7   Snow
4   1/5/2017        32          4   Rain
```

```
#read excel file
df = pd.read_excel('weather_data.xlsx')
df
```

```
Out[3]:
```

	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain
5	1/6/2017	31	2	Sunny

We can read csv and excel files into pandas dataframes as shown above.

```
1 #write DF to csv
2 df.to_csv('new.csv')
3 df.to_csv('new_noIndex.csv', index=False)
```

```
1 # INSTALL: pip3 install openpyxl
2
3 #write DF to Excel
4 df.to_excel('new.xlsx', sheet_name='weather_data')
```

Similarly we can store data into csv or excel files as shown above.

2.GROUP-BY

```
] 1 g = df.groupby('city')
2 g
<pandas.core.groupby.DataFrameGroupBy object at 0x106d495f8>
```

```
1 for city, city_df in g:
2     print(city)
3     print(city_df)

mumbai
      day   city  temperature  windspeed  event
4 1/1/2017  mumbai        90          5  Sunny
5 1/2/2017  mumbai        85         12    Fog
6 1/3/2017  mumbai        87         15    Fog
7 1/4/2017  mumbai        92          5   Rain

new york
      day   city  temperature  windspeed  event
0 1/1/2017  new york       32          6   Rain
1 1/2/2017  new york       36          7  Sunny
2 1/3/2017  new york       28         12   Snow
3 1/4/2017  new york       33          7  Sunny

paris
      day   city  temperature  windspeed  event
8 1/1/2017    paris        45         20  Sunny
9 1/2/2017    paris        50         13 Cloudy
10 1/3/2017   paris        54          8 Cloudy
11 1/4/2017   paris        42         10 Cloudy
```

When we used groupby() on city column ,it groups all the rows where city name is same and group them together.Its similar to groupby command in SQL

```
1 #or to get specific group  
2 g.get_group('new york')  
3
```

	day	city	temperature	windspeed	event
0	1/1/2017	new york	32	6	Rain
1	1/2/2017	new york	36	7	Sunny
2	1/3/2017	new york	28	12	Snow
3	1/4/2017	new york	33	7	Sunny

```
1 #Find maximum temperature in each of the cities  
2 print(g.max())
```

city	day	temperature	windspeed	event
mumbai	1/4/2017	92	15	Sunny
new york	1/4/2017	36	12	Sunny
paris	1/4/2017	54	20	Sunny

```

1 print(g.mean())
2

      temperature  windspeed
city
mumbai          88.50       9.25
new york        32.25       8.00
paris           47.75      12.75

1 print(g.describe())
\

      temperature
      count   mean       std    min    25%    50%    75%    max
city
mumbai          4.0  88.50  3.109126  85.0  86.50  88.5  90.50  92.0
new york        4.0  32.25  3.304038  28.0  31.00  32.5  33.75  36.0
paris           4.0  47.75  5.315073  42.0  44.25  47.5  51.00  54.0

      windspeed
      count   mean       std    min    25%    50%    75%    max
city
mumbai          4.0   9.25  5.057997  5.0   5.00    8.5  12.75  15.0
new york        4.0   8.00  2.708013  6.0   6.75    7.0  8.25  12.0
paris           4.0  12.75  5.251984  8.0   9.50   11.5  14.75  20.0

```

Using the groupby object that we obtained we can obtain data of each group,mean etc as shown above.

3.Concatenate Data Frames

```
1 import pandas as pd
2 india_weather = pd.DataFrame({
3     "city": ["mumbai", "delhi", "banglore"],
4     "temperature": [32, 45, 30],
5     "humidity": [80, 60, 78]
6 })
7
8 india_weather
```

	city	humidity	temperature
0	mumbai	80	32
1	delhi	60	45
2	banglore	78	30

```
1 us_weather = pd.DataFrame({
2     "city": ["new york", "chicago", "orlando"],
3     "temperature": [21, 14, 35],
4     "humidity": [68, 65, 75]
5 })
6 us_weather
```

	city	humidity	temperature
0	new york	68	21
1	chicago	65	14
2	orlando	75	35

- Let's create two data frames as shown above

```
1 #concat two dataframes
2 df = pd.concat([india_weather, us_weather])
3 df
```

	city	humidity	temperature
0	mumbai	80	32
1	delhi	60	45
2	banglore	78	30
0	new york	68	21
1	chicago	65	14
2	orlando	75	35

```
1 #if you want continuous index
2 df = pd.concat([india_weather, us_weather], ignore_index=True)
3 df
```

	city	humidity	temperature
0	mumbai	80	32
1	delhi	60	45
2	banglore	78	30
3	new york	68	21
4	chicago	65	14
5	orlando	75	35

- We can concatenate two dataframes using concat() as shown above ,for getting a unique index while concatenating we set ignore_index parameter to true.
- We can Combine dataframe objects horizontally along the x axis by passing in axis=1as shown below.

```
] 1 df = pd.concat([india_weather, us_weather],axis=1)
2 df
```

	city	humidity	temperature		city	humidity	temperature
0	mumbai	80	32	new york	68	21	
1	delhi	60	45	chicago	65	14	
2	banglore	78	30	orlando	75	35	

4. Merge DataFrames

Consider two data frames as shown below

```
1 temperature_df = pd.DataFrame({  
2     "city": ["mumbai", "delhi", "banglore", 'hyderabad'],  
3     "temperature": [32, 45, 30, 40]})  
4 temperature_df
```

city temperature

	city	temperature
0	mumbai	32
1	delhi	45
2	banglore	30
3	hyderabad	40

```
[ ] 1 humidity_df = pd.DataFrame({  
2     "city": ["delhi", "mumbai", "banglore"],  
3     "humidity": [68, 65, 75]})  
4 humidity_df
```

city humidity

	city	humidity
0	delhi	68
1	mumbai	65
2	banglore	75

```
| 1 #merge two dataframes without explicitly mention index  
2 df = pd.merge(temperature_df, humidity_df, on='city')  
3 df
```

```
|      city  temperature  humidity  
0    mumbai        32        65  
1     delhi         45        68  
2   banglore        30        75
```

```
| 1 #OUTER-JOIN  
2 df = pd.merge(temperature_df, humidity_df, on='city', how='outer')  
3 df
```

```
|      city  temperature  humidity  
0    mumbai        32        65.0  
1     delhi         45        68.0  
2   banglore        30        75.0  
3  hyderabad        40        NaN
```

- We can merge two dataframes as shown above ,it is similar to join in SQL.

5.Numerical Indexing (.loc vs iloc)

```
| 1 df = pd.DataFrame([1,2,3,4,5,6,7,8,9,19], index=[49,48,47,46,45, 1, 2, 3, 4, 5])  
2 df
```

```
0  
49 1  
48 2  
47 3  
46 4  
45 5  
1 6  
2 7  
3 8  
4 9  
5 19
```

We can index our data frame numerically as shown above ,it creates an index column and by default the index will be the row number.

```
46 4  
45 5  
1 6  
2 7  
3 8  
4 9  
5 19
```

index

```
In [23]: s.loc[46] get rid  
Out[23]: 4
```

```
In [24]: s.iloc[3]  
Out[24]: 4
```

- We can use loc() and by specifying the index we can get the value as shown above.
- We can use iloc() and by specifying row number we can get the value.

13.1 Lambda Functions

- Lambda or Anonymous functions are the functions defined without a name.
- The normal functions are defined with the 'def' keyword whereas the lambda functions are defined with the 'lambda' keyword.
- Lambda functions are used extensively along with the built-in functions.

The below example was discussed at the timestamp 1:00.

```
def double(x):
    return x * 2

print(double(5))

10
```

In this example, we are defining the function `double()` which multiplies the given input number with '2'.

If we want to define the same function using the 'lambda' keyword, then it would be written as

```
double = lambda x: x*2

print(double(5))

10
```

In this way, we are defining the whole function in a single line.

Here '**double**' is the function name, the 'x' present to the left side of colon(:) is an argument and **x*2** is the return value.

For example, if we want to compute the doubled value for number 10, then the syntax for it would be **double(10)**.

Below are the examples that were discussed starting from the timestamp 2:52 illustrating the usage of lambda functions along with the `map()`, `filter()` and `reduce()` functions.

Example 1

```
#Example use with filter()
lst = [1, 2, 3, 4, 5]
even_lst = list(filter(lambda x: (x%2 == 0), lst))
print(even_lst)
```

```
[2, 4]
```

In the above example, we are passing a list as an input and want only the even numbers out of it. So we are defining a lambda function **x:x%2==0** which means for the element ‘x’ it returns **True**, only if it is divisible by 2. If $x \% 2 == 0$, then it will be taken into the new list ‘even_lst’. Otherwise the element ‘x’ will be ignored.

Example 2

```
#Example use with map()
lst = [1, 2, 3, 4, 5]
new_lst = list(map(lambda x: x ** 2, lst))
print(new_lst)
```

```
[1, 4, 9, 16, 25]
```

In this example, we are passing a list as an input and want to apply square operation on all the elements in the list and get the result. So here we are defining a lambda function **x:x**2** which means for the element ‘x’, the square operation is performed. The **map()** function applied this lambda function to every element in the list ‘lst’ and returns the result in the form of a ‘map’ object. If we want the result in a list format, then we have to apply **list()** function on the ‘map’ object.

Example 3

```
#Example use with reduce()
from functools import reduce

lst = [1, 2, 3, 4, 5]
product_lst = reduce(lambda x, y: x*y, lst)
print(product_lst)
```

In this example, we are defining a lambda function that performs the product of two numbers. So here we are passing a list 'lst' and the reduce() picks the first 2 elements in the list and computes the product and then computes the product of the obtained result with the third element. This result is multiplied with the 4th element and this continues till the end.

13.2 File Handling in Python

File I/O

- File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk).
- Since, random access memory (RAM) is volatile which loses its data when the computer is turned off, we use files for future use of the data.
- When we want to read from or write to a file we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed.

File Operations

- 1) Opening a file
- 2) Read or Write
- 3) Closing a file

Opening a file

Python has a built-in function `open()` to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

Syntax

`f = open(file_name, mode)`

Here ‘file_name’ is the name of the file that has to be opened. ‘Mode’ indicates whether to read from the file or write the data to the file or append the data to the existing data in the file.

‘F’ is a file object used to handle the operations on the file. It is also called ‘Handle’.

Example

`f = open('example.txt','r')`

Here it first checks if the file ‘example.txt’ is present in the current directory. If yes, then this file will be opened in read mode. If the file is not present, then it throws an error.

We also have to specify the mode while opening a file. In mode, we specify whether we want to read ‘r’, write ‘w’ or append ‘a’ to the file. We also can specify if we want to open the file in text mode or binary mode.

If we do not specify the mode, then the default mode will be the reading mode.

File Modes

- 'r' → Opens a file for reading. (default)
- 'w' → Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
- 'x' → Opens a file for exclusive creation. If the file already exists, the operation fails.
- 'a' → Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
- 't' → Open in text mode. (default)
- 'b' → Open in binary mode.
- '+' → Open a file for updating (reading and writing)

- The default encoding is platform dependent. In windows, it is '**cp1252**' but '**utf-8**' in Linux.
- So, we must not also rely on the default encoding or else our code will behave differently in different platforms.
- Hence, when working with files in text mode, it is highly recommended to specify the encoding type.

Below is an example that shows how to open a file

```
: f = open('example.txt') #equivalent to 'r'  
f = open('example.txt', 'r')  
  
f = open('test.txt', 'w')
```

In the first statement, we are opening the file directly without specifying the mode. But still the default mode is read('r'). So the file opens in read mode.

In the second statement, the file opens in read mode, as we have specified the mode as 'r'.

In both these statements, it first checks if the specified file is present in the current directory or not. If the file is present, then it will go ahead with opening the file. If the file is not present, then it will throw an error (FileNotFoundException).

In the third statement, as we want to open the file in write mode, it first checks if the file is present in the current directory. If the file is present, then all the contents in the file will be deleted and this file will be opened for writing

purpose. If the file is not present, then it creates a file and then opens it for writing purpose.

Closing a file

- Closing a file will free up the resources that were tied with the file and is done using the `close()` method.
- Python has a garbage collector to clean up unreferenced objects but, we must not rely on it to close the file.

Below is an example discussed at the timestamp 5:45

```
f = open('example.txt')
f.close()
```

This method of closing a file is not at all safe because if any exception occurs while performing any operations, then the code exits without closing the file.

Hence the best and the safe way is given below

```
try:
    f = open("example.txt")
    # perform file operations

finally:
    f.close()
```

In this example, we are opening the file and performing file operations. This block of code is enclosed in the 'try' block. In case, if any exception occurs, then the control comes out of it and executes the 'finally' block in which we are closing the file.

This way, we are guaranteed that the file is properly closed even if an exception is raised, causing the program flow to stop.

The best way to do this is using the **with** statement. This ensures that the file is closed when the '**with**' block is exited. We don't need to explicitly call the `close()` method. It is done internally.

The syntax to open a file 'example.txt' without using the 'with' keyword is

```
f = open('example.txt')
```

The syntax to open the same file using the ‘with’ statement is

```
with open("example.txt",encoding = 'utf-8') as f:
```

Writing to a file

- In order to write the data into a file, we have to open a file in write ('w'), append ('a') or exclusive creation ('x') mode.
- Whenever we want to write the data into a file, if there is any data already present in it, then all the data will be erased.
- Writing a string or sequence of bytes (for binary files) is done using **write()** method. This method returns the number of characters written to the file.

Below is an example of writing data to a file and it has been discussed at the timestamp 8:30

```
f = open("test.txt", "w")
f.write("This is a First File\n")
f.write("Contains two lines\n")
f.close()
```

In this example, in the first statement, we are opening the file ‘test.txt’ in write mode. If this file is already present, then it would be opened in write mode. If it is not present in the current directory, then a new file is created with the name ‘test.txt’ and is opened in the write mode.

In the 2nd and the 3rd statements, we are writing the data to the file. In case, if any data is already existing in the file, then all that data will be erased and this new data gets added into the file.

In the 4th statement, we are closing the file.

Reading from a file

There are various methods available to read the data from the files.

- | | | |
|------------|---|--|
| read(size) | → | reads the specified ‘size’ number of characters. |
| readline() | → | reads only one line of the data |
| read() | → | reads the entire file |

In `read()` method if a value is not specified for the 'size' parameter, then it reads the entire file.

The below examples were discussed starting from the timestamp 9:35

```
: f = open("test.txt", "r")
f.read()

: 'This is a First File\nContains two lines\n'
```

```
: f = open("test.txt", "r")
f.read(4)

: 'This'
```

```
: #f = open("test.txt", "r")
f.read(10)

: ' is a Firs'
```

In the first example, we are opening the file and are reading the entire data from the file.

In the second example, we are opening the file and as we have specified the size as 4 in `read()` method, only the first 4 characters will be read from the file.

In the third cell, as we are not opening the file again and also we haven't closed the file in the previous example, the first 4 characters were already read. So the cursor is now at position 4. When we pass the size as 10 in the `read()` method, from the 5th position, till the 10th position all the characters will be read.

We can change the file cursor position using the `seek()` method. We can find out the current position of our file cursor using `tell()` method. The below examples were discussed in the video starting from the timestamp 12:00.

```
f.tell()
```

```
14
```

```
f.seek(0) #bring the file cursor to initial position
```

```
0
```

```
print(f.read()) #read the entire file
```

```
This is a First File  
Contains two lines
```

Here after executing the previously discussed examples, the file cursor has moved to the 14th position. So when we run the **f.tell()** method, it is showing the current cursor position as 14.

As we wanted the cursor to move back to the 0th position, we are calling the **f.seek(0)** method. The value in the parentheses of **seek()** method indicates the position to which the cursor has to be moved.

Again from here we are running **f.read()** to read the entire file.

Apart from reading the files character wise, we also can read them line by line. Using a loop, we can read a file line by line and this is the fastest and efficient approach. Below are the examples illustrating the file reading line by line.

```
: f.seek(0)  
for line in f:  
    print(line)
```

```
This is a First File
```

```
Contains two lines
```

In this example, we are running a loop, starting from the position 0 till the end. Here the contents are read until it encounters the next line character (ie., \n). This way all the lines are read.

```
f = open("test.txt", "r")
f.readline()
```

```
'This is a First File\n'
```

```
f.readline()
```

```
'Contains two lines\n'
```

```
f.readline()
```

```
..
```

In this above example, we are reading line by line using the readline() method. Here while reading the characters from the cursor position, if it encounters a newline character(ie., \n), then it the data till there would be read and is returned. Again when we execute the readline() method, then from the position of the new line character that was encountered, till the next new line character, the data would be read and is returned. This way, using the readline() method, we can read all the contents present in the file.

```
f.seek(0)
f.readlines()
```

```
['This is a First File\n', 'Contains two lines\n']
```

In this above code, all the lines are read at once starting from the cursor position(here it is 0) till the end of the file and are returned as a list of strings.

All the above discussed methods will stop reading/fetching the data and return empty values once if the end of the file is reached.

Renaming and Deleting Files in Python

Along with read/write operations, we also can perform rename/delete operations on the files. But in order to perform these operations, we have to import the 'os' module.

Below examples were discussed starting from the timestamp 14:50

```
import os

#Rename a file from test.txt to sample.txt
os.rename("test.txt", "sample.txt")

f = open("sample.txt", "r")
f.readline()

'This is a First File\n'
```

In the above example, we are renaming the ‘test.txt’ file to ‘sample.txt’ and then we are reading the contents from the ‘sample.txt’ file and it all went successfully.

```
#Delete a file sample.txt
os.remove("sample.txt")

f = open("sample.txt", "r")
f.readline()

-----
FileNotFoundException                                     Traceback (most recent call last)
<ipython-input-25-6eeb6c12a935> in <module>()
----> 1 f = open("sample.txt", "r")
      2 f.readline()

FileNotFoundException: [Errno 2] No such file or directory: 'sample.txt'
```

In this example, we are deleting the ‘sample.txt’ file and are trying to open and read the file contents. But we are unsuccessful in opening and reading the file, as it was already deleted.

Python Directory and File Management

- If there are a large number of files to handle in your Python program, you can arrange your code within different directories to make things more manageable.
- A directory or folder is a collection of files and subdirectories. Python has the `os` module, which provides us with many useful methods to work with directories (and files as well).

Getting the Current Directory

We can get the name of the current directory using the `getcwd()` method of the '`os`' module. This method returns the directory name in the form of a string.

The below example was discussed at the timestamp 16:10

```
import os  
os.getcwd()  
  
'/Users/varma/Google Drive/OnlineVideos/2/python-basics/File Operation'
```

Changing the Directory

We can change the current working directory using the `chdir()` method. The new path that we want to change to must be supplied as a string to this method. We can use both forward slash (/) or the backward slash (\) to separate path elements.

The below example was discussed at the timestamp 16:40

```
: os.chdir("/Users/varma/")  
  
: os.getcwd()  
: '/Users/varma'
```

List Directories and Files

We can get a list of all the directories and files using the `listdir()` method. We have to specify the path of the directory as an argument for `listdir()` method and it returns a list of all directories and files in that directory.

If we do not specify any path, then by default it gives a list of all directories and files in the current directory.

Below is the syntax to find out the list of all directories and files in the current directory.

```
os.listdir(os.getcwd())
```

Making New Directory

- We can make a new directory using the **mkdir()** method.
- This method takes in the path of the new directory. If the full path is not specified, the new directory is created in the current working directory.
- If we want to remove an empty directory, then we have to use the **rmdir()** method. This method works only on empty directories.
- If we want to remove non-empty directories, then we have to use **rmtree()** method in the '**shutil**' module.

The below example was discussed at the timestamp 19:20 illustrating all the above discussed operations.

```
: import shutil  
  
os.mkdir('test')  
os.chdir('./test')  
f = open("testfile.txt",'w')  
f.write("Hello World")  
os.chdir("../")  
os.rmdir('test')
```

```
-----  
OSError Traceback (most recent call last)  
<ipython-input-39-a990945f349d> in <module>()  
      6 f.write("Hello World")  
      7 os.chdir("../")  
----> 8 os.rmdir('test')  
      9  
     10  
  
OSError: [Errno 66] Directory not empty: 'test'
```

```
: # remove an non-empty directory  
shutil.rmtree('test')
```

```
: os.getcwd()  
'/Users/varma'
```

13.3 Exception Handling

Python Errors and Built-in Exceptions

- We sometimes do encounter errors while writing the programs. Error caused by not following the proper structure (syntax) of the language is called syntax error or parsing error.
- Errors can also occur at runtime and these are called **exceptions**.
- Few examples are when a file we try to open does not exist (FileNotFoundException), dividing a number by zero (ZeroDivisionError), module we try to import is not found (ImportError) etc.
- Whenever these types of runtime errors occur, Python creates an exception object. If not handled properly, it prints a traceback to that error along with some details about why that error occurred.

Below are a few examples of errors that were discussed starting from the timestamp 0:15

```
if a < 3
File "<ipython-input-2-8625009197cc>", line 1
    if a < 3
    ^
SyntaxError: invalid syntax
```

Here we haven't used the colon symbol after the condition, hence it has raised a syntax error.

```
1 / 0
-----
ZeroDivisionError                                Traceback (most recent call last)
<ipython-input-3-b710d87c980c> in <module>()
----> 1 1 / 0

ZeroDivisionError: integer division or modulo by zero
```

Here we are trying to divide a number and hence it has raised the ZeroDivisionError.

```
: open('test.txt')
-----
IOError                                     Traceback (most recent call last)
<ipython-input-4-46a2b0c9e87f> in <module>()
----> 1 open('test.txt')

IOError: [Errno 2] No such file or directory: 'test.txt'
```

As there is no file called ‘test.txt’, it has raised the IOError.

Python Built-in Exceptions

The `dir()` gives a list of all the built-in exceptions that are handled by Python.

The syntax of the `dir()` has been explained at the timestamp 2:00

```
dir(__builtins__)

['ArithmeticError',
 'AssertionError',
 'AttributeError',
 'BaseException',
 'BlockingIOError',
 'BrokenPipeError',
 'BufferError',
 'BytesWarning',
 'ChildProcessError',
 'ConnectionAbortedError',
 'ConnectionError',
 'ConnectionRefusedError',
 'ConnectionResetError',
 'DeprecationWarning',
 'EOFError',
 'Ellipsis',
 'EnvironmentError',
 'Exception',
 'False',
 'FileNotFoundError',
 'FloatingPointError',
 'GeneratorExit',
 'ImportError',
 'ImportWarning',
 'IndexError',
 'KeyError',
 'KeyboardInterrupt',
 'MemoryError',
 'NameError',
 'NotImplementedError',
 'NotImplementedWarning',
 'OverflowError',
 'PendingDeprecationWarning',
 'RuntimeError',
 'StopAsyncIteration',
 'StopIteration',
 'SyntaxError',
 'SyntaxWarning',
 'TimeoutError',
 'ValueError',
 'Warning']
```

Python Exception Handling - Try, Except and Finally blocks

- Python has many built-in exceptions which forces the program to output an error when something in it goes wrong.
 - When these exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled. If not handled, our program will crash.

- For example, if function A calls function B which in turn calls function C and an exception occurs in function C. If it is not handled in C, the exception passes to B and then to A.
- If never handled, an error message is spit out and the program comes to a sudden, unexpected halt.

Catching Exceptions in Python

- In Python, the exceptions are handled using the ‘try’ statement.
- A critical operation which can raise an exception is placed inside the try clause and the code that handles exception is written in the except clause.

Below is an example discussed at the timestamp 2:35 in the video.

```
: # import module sys to get the type of exception
import sys

lst = ['b', 0, 2]

for entry in lst:
    try:
        print("The entry is", entry)
        r = 1 / int(entry)
    except:
        print("Oops!", sys.exc_info()[0], "occurred.")
        print("Next entry.")
        print("*****")
    print("The reciprocal of", entry, "is", r)

The entry is b
Oops! <class 'ValueError'> occurred.
Next entry.
*****
The entry is 0
Oops! <class 'ZeroDivisionError'> occurred.
Next entry.
*****
The entry is 2
The reciprocal of 2 is 0.5
```

Here in this example, we are looping through the list ‘lst’ and are trying to divide the number 1 by each of the elements in the list ‘lst’.

When we divide the number 1 by character ‘b’, it throws ValueError.

When we divide the number 1 by the number ‘0’, it throws ZeroDivisionError.

When we divide the number 1 by the number 2, then it gives the result as 0.5

Note: We can have only one ‘try’ block and one or more ‘except’ clauses. But the except clause(s) execute only if an exception occurs in the ‘try’ block. Otherwise the ‘except’ block doesn’t execute at all.

The name of the exception occurred is obtained by the function `sys.exc_info()`

Catching Specific Exceptions in Python

In the above example, we did not mention any exception in the except clause.

This is not a good programming practice as it will catch all exceptions and handle every case in the same way. We can specify which exceptions an except clause will catch.

A try clause can have any number of except clauses to handle them differently but only one will be executed in case an exception occurs.

The below example was discussed at the timestamp 7:45

```

: import sys

lst = ['b', 0, 2]

for entry in lst:
    try:
        print("*****")
        print("The entry is", entry)
        r = 1 / int(entry)
    except(ValueError):
        print("This is a ValueError.")
    except(ZeroDivisionError):
        print("This is a ZeroError.")
    except:
        print("Some other error")
    print("The reciprocal of", entry, "is", r)

*****
The entry is b
This is a ValueError.
*****
The entry is 0
This is a ZeroError.
*****
The entry is 2
The reciprocal of 2 is 0.5

```

In this example, we are handling the ZeroDivisionError and ValueError exceptions separately by writing separate 'except' clauses for them. After this we have again defined an 'except' clause that handles other exceptions.

Raising Exceptions

- In Python, exceptions are raised whenever the corresponding errors occur at the runtime. But we also can forcefully raise exceptions using the 'raise' keyword.
- We can also optionally pass in value to the exception to clarify why that exception was raised.

The below examples illustrate how to manually raise exceptions. These examples were discussed starting from the timestamp 9:30.

```
: raise KeyboardInterrupt
```

```
-----  
KeyboardInterrupt                                Traceback (most recent call last)  
<ipython-input-15-7d145351408f> in <module>()  
----> 1 raise KeyboardInterrupt
```

```
KeyboardInterrupt:
```

```
: raise MemoryError("This is memory Error")
```

```
-----  
MemoryError                                 Traceback (most recent call last)  
<ipython-input-16-0ce8140e6e3c> in <module>()  
----> 1 raise MemoryError("This is memory Error")
```

```
MemoryError: This is memory Error
```

```
: try:  
    num = int(input("Enter a positive integer:"))  
    if num <= 0:  
        raise ValueError("Error:Entered negative number")  
except ValueError as e:  
    print(e)
```

```
Enter a positive integer:-10  
Error:Entered negative number
```

In this above example, we are checking if the given input number is less than or equal to 0. If yes, then we are raising the ‘ValueError’ exception and in the ‘except’ clause we are printing the value we passed to the exception to clearly indicate why the error has occurred.

As the value entered is less than 0, the ‘ValueError’ got raised and the message is getting printed.

Try-finally

- The ‘try’ statement in Python can have an optional ‘finally’ clause.
- Irrespective of whether an exception occurs or not, the ‘finally’ block gets executed.
- In case, if any exception occurs in the ‘try’ block, then the corresponding ‘except’ block gets executed and then the ‘finally’ block gets executed.

- In case, if there are no exceptions occurred in the ‘try’ block, then immediately the ‘finally’ block gets executed.
- This clause is executed no matter what, and is generally used to release external resources. The ‘finally’ block is always present after the ‘try’ and all the ‘except’ blocks.
- It is always a good programming practice to specify the ‘finally’ block whenever we are using the ‘try’ and ‘except’ blocks in our code.

The below example was discussed at the timestamp 12:45 that illustrates the usage of ‘finally’ block.

```
try:  
    f = open('sample.txt')  
    #perform file operations  
  
finally:  
    f.close()
```

In this example, we are performing the file closing operation in the ‘finally’ block because sometimes whenever we open a file and are working on it and if any exception occurs, then the code execution stops without closing the file. In order to release the resources occupied, we can use ‘finally’ block.

13.4 Debugging in Python

Debugging is about understanding where the bugs are in our program. **pdb** implements an interactive debugging environment for Python programs. It includes features to let you pause your program, look at the values of variables, and watch program execution step-by-step, so you can understand what your program actually does and find bugs in the logic.

Starting the Debugger

From the Command Line

The below examples were discussed starting from the timestamp 0:52

```
def seq(n):
    for i in range(n):
        print(i)
    return

seq(5)

0
1
2
3
4
```

From within the program

```
: import pdb

#interactive debugging
def seq(n):
    for i in range(n):
        pdb.set_trace() # breakpoint
        print(i)
    return

seq(5)

# c : continue
# q: quit
# h: help
# llist
# p: print
# p locals()
# p globals()
```

Debugger Commands

1) h(elp)

Without argument, print the list of available commands. With a command as argument, print help about that command. help pdb displays the full documentation (the docstring of the pdb module). Since the command argument must be an identifier, help exec must be entered to get help on the ! command.

2) w(here)

Print a stack trace, with the most recent frame at the bottom. An arrow indicates the current frame, which determines the context of most commands.

3) d(own)

Move the current frame count (default one) levels down in the stack trace (to a newer frame).

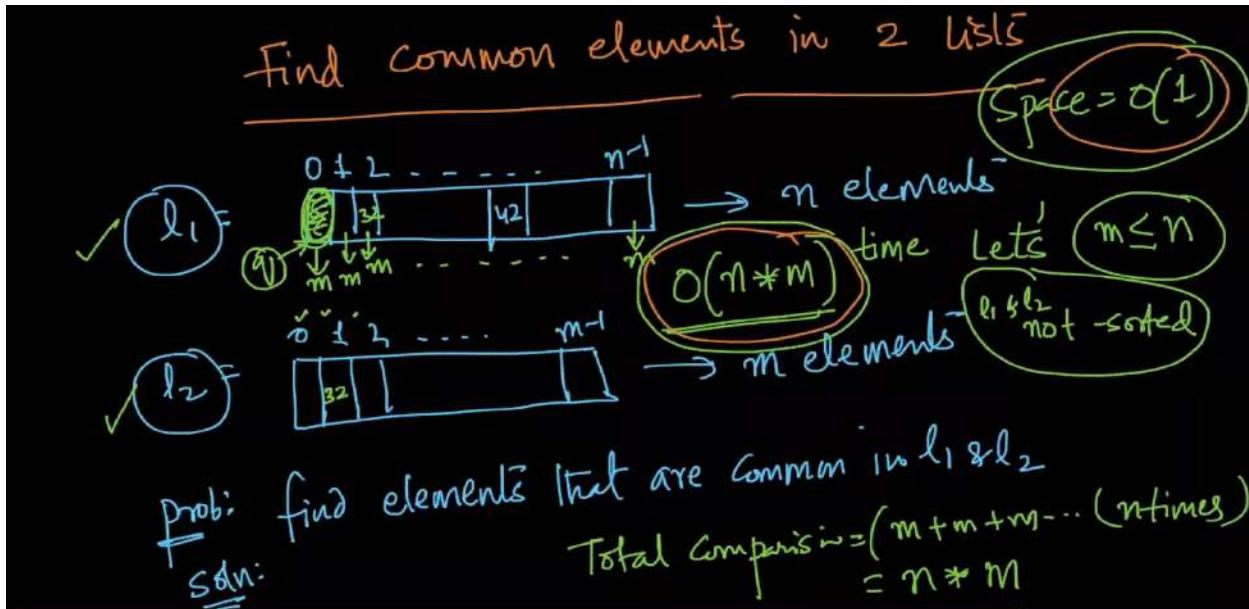
4) c(ont(inue))

Continue execution, only stop when a breakpoint is encountered.

5) q(uit)

Quit from the debugger. The program being executed is aborted.

13.5 Solved Problem: Finding the elements common in two lists



Below is the code snippet that was discussed at the timestamp 4:20

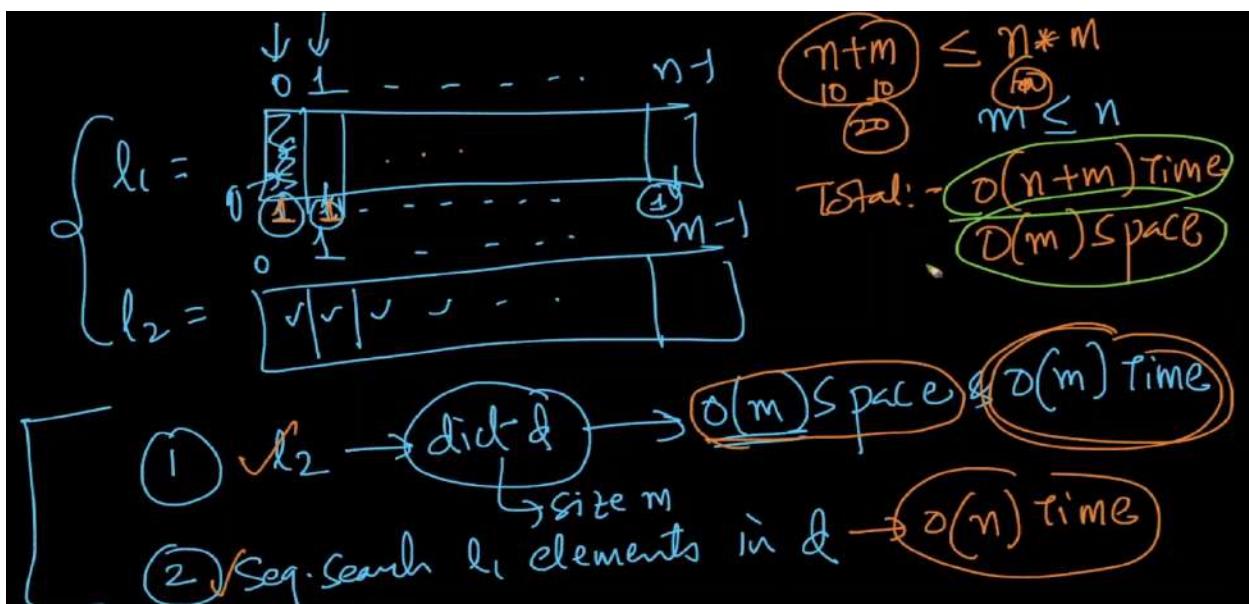
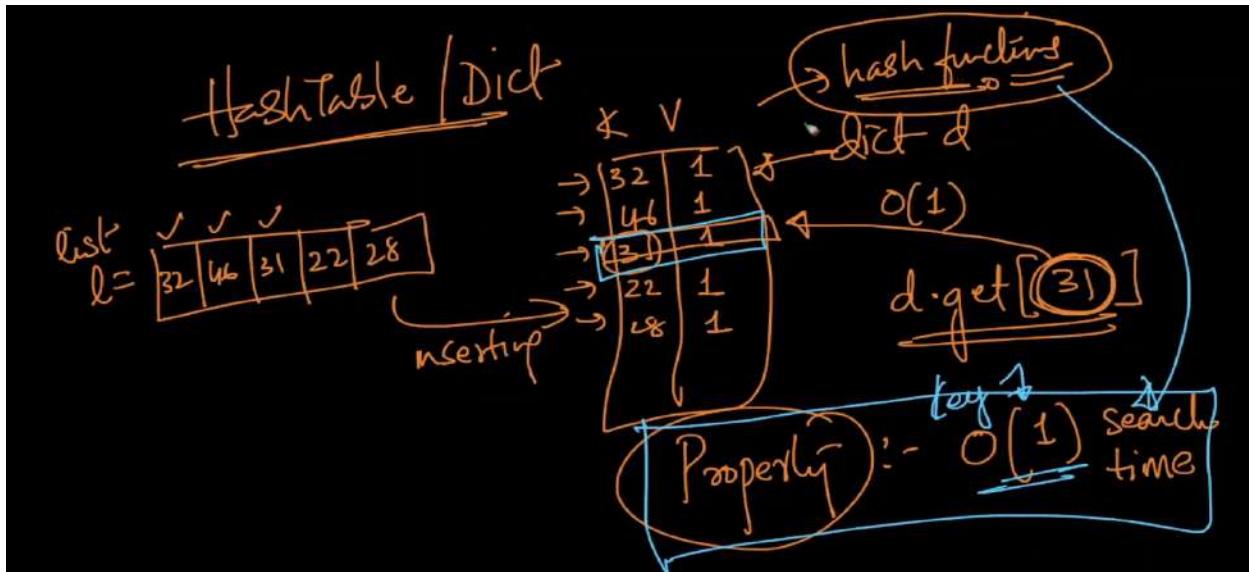
```
# Find elements common in two lists:
l1 = list(range(100))
random.shuffle(l1)

l2 = list(range(50))
random.shuffle(l2)

# find common elements : O(n*m)
cnt=0;
for i in l1:
    for j in l2:
        if i==j:
            print(i)
            cnt += 1;
print("Number of common elements:", cnt)
```

- In the above example, we are generating two lists 'l1' and 'l2'.
- We are initializing a variable 'cnt' to 0.
- We are looping through each of the lists using two 'for' loops and comparing each element in 'l1' with each element in 'l2'.
- If we find any element present in both the lists, then we have to increment 'cnt' by 1.
- If 'n' is the number of elements in 'l1' and 'm' is the number of elements in 'l2', then the time complexity would be $O(n*m)$. As we are using only one variable (ie., 'cnt'), the space complexity is $O(1)$.

13.6 Solved Problem: Finding the elements common in two lists using Hashtable/Dict



$n \rightarrow$ number of elements in list ' l_1 '

$m \rightarrow$ number of elements in list ' l_2 '

In the previous scenario, we have seen the time complexity was $O(n \cdot m)$ and the space complexity was $O(1)$. In order to improve the time complexity, we go with another approach. In this approach, we are using a dictionary/hashtable to store the elements of the list ' l_2 '. It takes $O(m)$ time complexity to store all the elements of ' l_2 ' in the dictionary.

After storing the elements of 'l2' in the dictionary, we are checking if each element of 'l1' is present in the dictionary as a key. For this comparison purpose, the time complexity is $O(n)$. The space complexity used to store the 'm' elements in the dictionary is $O(m)$.

So for the addition of 'm' elements into the dictionary, the time complexity is $O(n)$ and the time complexity for comparison of 'n' elements with the 'm' keys is $O(m)$. So the total time complexity is $O(m+n)$ and the total space complexity is $O(m)$.

The below example was discussed at the timestamp 9:30 in the video.

```
# Find elements common in two lists:
l1 = list(range(100))
random.shuffle(l1)

l2 = list(range(50))
random.shuffle(l2)

# find common elements in lists in O(n) time and O(m) space if m < n

## add all elements in the smallest list into a hashtable/Dict: O(m) space
smallList = {}
for ele in l2:
    smallList[ele] = 1; # any value is OK. Key is important

# Now find common element
cnt=0;
for i in l1:
    if smallList.get(i) != None: # search happens in constant time.
        print(i);
        cnt += 1;
print("Number of common elements:", cnt)
```

Time Complexity for storing the 'm' elements in the dictionary = $O(m)$

Time Complexity for checking if all the 'n' elements are present in the dictionary = $O(n)$

The total time complexity of this program = $O(n+m)$

Space Complexity for this program = $O(m)$ (as we are storing the 'm' values in the dictionary)

14.1 Procedural vs Object Oriented Programming

- Large software can be built in C. e.g: Linux Kernel
- OOP: Easier to design, reuse components and build large software
- SDE: should know OOP as they will be working in teams building large software
- ML & DS roles: Should know the basics to use various modules and extend them when needed.
- The true power of OOP and OOD can be understood when designing a large piece of software like a library for plotting

14.2 Classes and Objects (real-world example: the TinyURL service)

```
3 import random
4 import string
5 d = dict();
6 # given a long URL, get a short URL
7 def getShortURL(longURL):
8     # length = random value in 6-10
9     l = random.randint(6,10);
10
11    # generate random characters into a string of length l
12    chars = string.ascii_lowercase
13    shortURL = ''.join(random.choice(chars) for i in range(l))
14
15    # check if this string is already present in dict d
16    if shortURL in d:
17        return getShortURL(longURL);
18    else:
19        d[shortURL] = longURL;
20
21
22    r = "https://www.shortURL.com/" + shortURL
23    return r;
24
25 def getLongURL(shortURL):
26
27    # extract key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
28    k = shortURL[25:];
29
30    if k in d:
31        return d[k];
32    else:
33        return None;
```

- We have defined a dictionary two functions getLongURL() and getShortURL() as shown above.
- A class is basically a logical grouping of all the related functions and members together, we convert the above procedural implementation into object oriented implementation as shown below.

```

1 # Class: group all variables/attributes and functions/methods into a single logical unit
2
3 class ShortURL:
4
5     def __init__(self): # constructor; not must, but, good to have; initialize all attributes here
6         self.d=dict();
7
8     # given a long URL, get a short URL
9     def getShortURL(self, longURL): # first argument to all methods is "self" => this object
10        # length = random value in 6-10
11        l = random.randint(6,10);
12
13        # generate random characters into a string of length l
14        chars = string.ascii_lowercase
15        shortURL = ''.join(random.choice(chars) for i in range(l))
16
17
18        # check if this string is already present in dict d
19        if shortURL in self.d:
20            return getShortURL(longURL);
21        else:
22            self.d[shortURL] = longURL;
23
24
25
26        r = "https://www.shortURL.com/" + shortURL
27        return r;
28
29    def getLongURL(self, shortURL):
30
31        # print(self.d); # print statement for debugging
32
33        # extract key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
34        k = shortURL[25:];
35
36
37        if k in self.d:
38            return self.d[k];
39        else:
40            return None;
41

```

Timestamp 2.39

Class: datatype

Object: variable

self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

```
1 # Class: datatype/DS & Object: variable of a class
2 s = ShortURL() # constructor being called; memory allocated.
3
4 print(type(s))
5
```

What does this "`_main_`" mean?

The script invoked directly is considered to be in the **main** module. It can be imported and accessed the same way as any other module.

other module.
Modules:
https://www.w3schools.com/python/python_modules.asp
Consider a module to be the same as a code library. A file containing a set of functions you want to include in your application.

variables, fun, classes

A module will have variables,functions,classes.

Modules:

Consider a module to be the same as a code library. A file containing a set of functions you want to include in your application.

```

▶ 1 print(s.shortURL("appliedaicourse"))
  2 print(s.shortURL("gate.appliedcourse.com"))

>User Profile
-----
AttributeError Traceback (most recent call last)
<ipython-input-52-84023b7d1bbb> in <module>
----> 1 print(s.shortURL("appliedaicourse"))
      2 print(s.shortURL("gate.appliedcourse.com"))

AttributeError: 'ShortURL' object has no attribute 'shortURL'

```

[SEARCH STACK OVERFLOW](#)

```

▶ 1 print(s.getShortURL("appliedaicourse.com"))
  2 print(s.getShortURL("gate.appliedcourse.com"))
  3

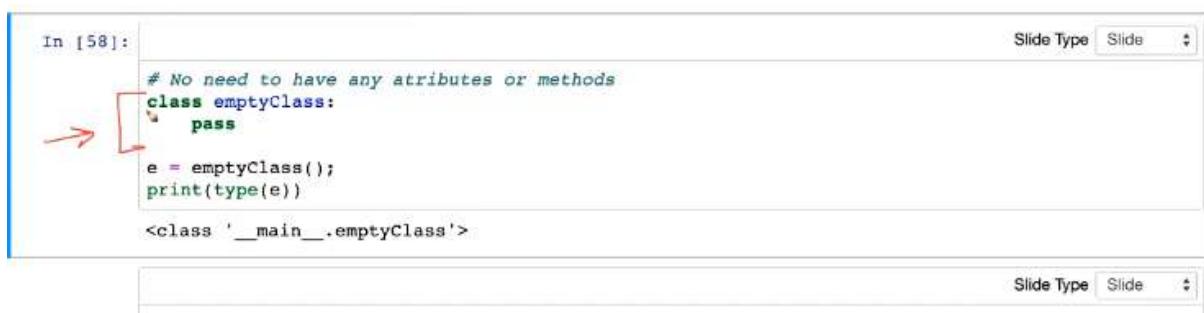
https://www.shortURL.com/alrpoy
https://www.shortURL.com/dlfjxuhff

```

Timestamp 19.45

Using the object we have created we call the function as shown above.

14.3 Empty Classes and documentation



```

In [58]:
# No need to have any attributes or methods
class emptyClass:
    pass

e = emptyClass();
print(type(e))

<class '__main__.emptyClass'>

```

- We can create empty classes in python as shown above.
- There are lots of internal and boundary cases for which reading documentation is a good idea.
- <https://docs.python.org/3/tutorial/classes.html>
- We will focus more on applied aspects in the context of ML/AI.

14.4 Class Variables and Private Members

Private members of the class are denied access from the environment outside the class. They can be handled only from within the class.

Public members (generally methods declared in a class) are accessible from outside the class. The object of the same class is required to invoke a public method.

```
5 class ShortURL:
6
7     URLPrefix = "https://www.shortURL.com/" # class variable shared by all objects
8
9     def __init__(self): # constructor; not must, but, good to have; initialize all attributes here
10        self.d=dict();
11
12    # given a long URL, get a short URL
13    def getShortURL(self, longURL): # first argument to all methods is "self" => this object
14        # length = random value in 6-10
15        l = random.randint(6,10);
16
17        # generate random characters into a string of length l
18        chars = string.ascii_lowercase
19        shortURL = ''.join(random.choice(chars) for i in range(l))
20
21
22        # check if this string is already present in dict d
23        if shortURL in self.d:
24            return getShortURL(longURL);
25        else:
26            self.d[shortURL] = longURL;
27
28
29        r = self.URLPrefix + shortURL
30        return r;
31
32    def getLongURL(self, shortURL):
33
34        # print(self.d); # print statement for debugging
35
36
37        # extract key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
38        k = shortURL[25:];
39
40
41        if k in self.d:
42            return self.d[k];
43        else:
44            return None;
```

Timestamp 1.35

The variable URLPrefix is a class variable

```
s1 = ShortURL1();

print(s1.getShortURL("appliedaicourse.com"))
print(s1.getShortURL("gate.appliedaicourse.com"))

https://www.shortURL.com/bqrkdpum
https://www.shortURL.com/jaxgluaeoh
```

9]:

```
print(s1.d)
print(s1.URLPrefix)

{'bqrkdpum': 'appliedaicourse.com', 'jaxgluaeoh': 'gate.appliedaicourse.com'}
https://www.shortURL.com/
```

1]:

```
s1a = ShortURL1();
print(s1a.URLPrefix);
print(s1a.d)

https://www.shortURL.com/
{}
```

Timestamp 3.21

Because URLprefix is a property of the class it can be shared by all objects of that class.

```
classVar = "test"; # Public => accesible directly from outside the class

__URLPrefix = "https://www.shortURL.com/"; # "Private" members. Member => attribute or method
```

Timestamp 8.47

- Adding two underscores(__) at the beginning makes a variable or a method **private** is the convention used by most **python** code
- If we convert URLprefix to a private variable we cannot access it from outside of class as shown below.

```

1 s2 = ShortURL2();
2 print(s2.classVar)
3 print(s2.__URLPrefix)

test
-----
AttributeError                                 Traceback (most recent call last)
<ipython-input-90-4fd735b2fbbe> in <module>
      1 s2 = ShortURL2();
      2 print(s2.classVar)
----> 3 print(s2.__URLPrefix)

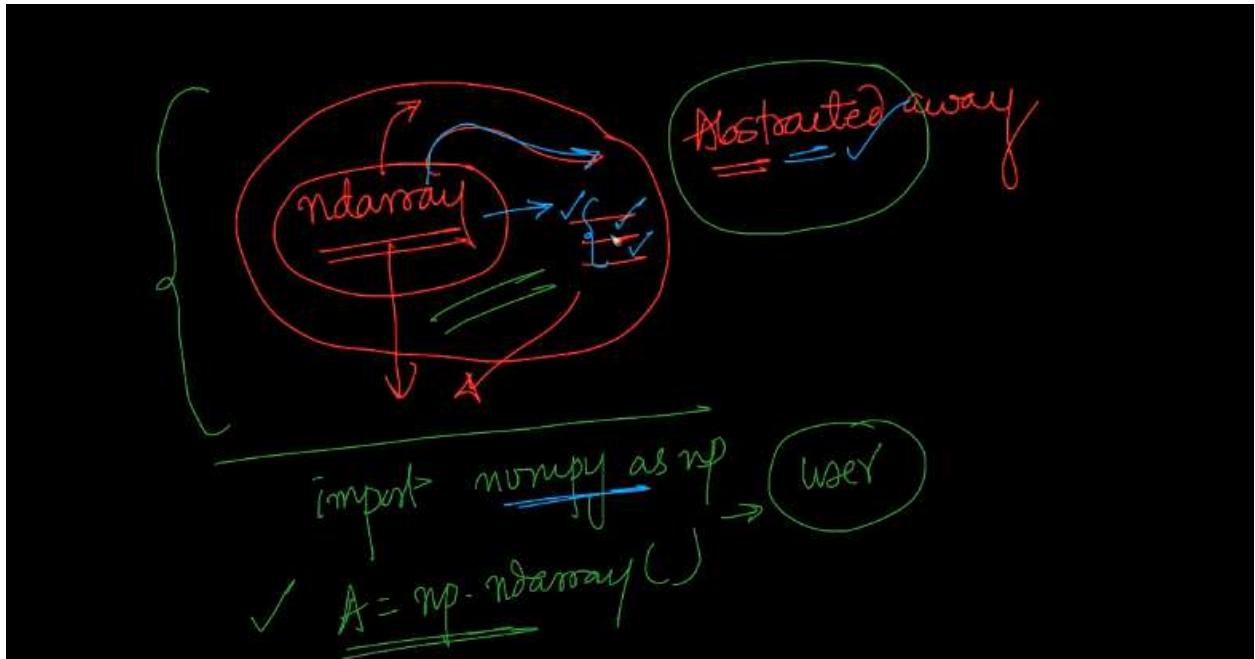
AttributeError: 'ShortURL2' object has no attribute '__URLPrefix'

```

[SEARCH STACK OVERFLOW](#)

14.5 Abstraction and Inheritance (with real-world examples)

- As a recommendation to the programmer, in its formulation by Benjamin C. Pierce in *Types and Programming Languages* (2002), the abstraction principle reads: “ Each significant piece of functionality in a program should be implemented in just one place in the source code. Where similar functions are carried out by distinct pieces of code, it is generally beneficial to combine them into one by abstracting out the varying parts. ”
- General concept in programming: libraries, classes
- "Its main goal is to handle complexity by hiding unnecessary details from the user. That enables the user to implement more complex logic on top of the provided abstraction without understanding or even thinking about all the hidden complexity." [Source: <https://stackify.com/oop-concept-abstraction/>]



We import numpy and we create ndarray using it ,we don't really need to understand how numpy is implemented the variables and functions declared it numpy and internals of how ndarray is implemented .The mathematical operations of ndarray are fairly complex and these things are abstracted away from user(as a programmer we don't need to know about this)

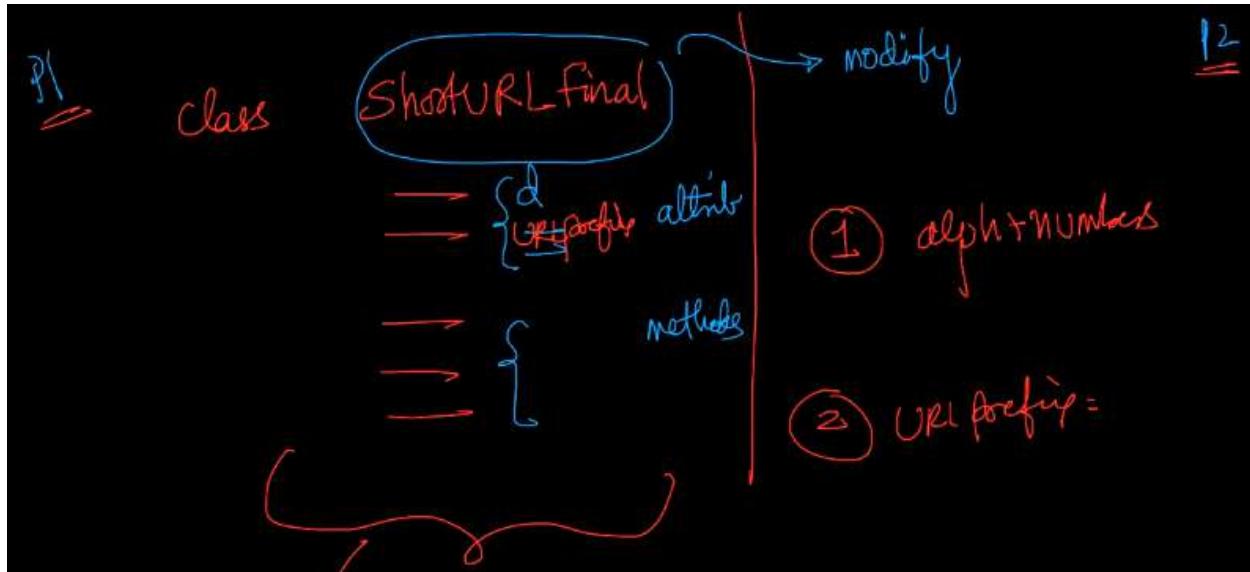
Encapsulation:

In object-oriented programming languages, and other related fields, encapsulation refers to one of two related but distinct notions, and sometimes to the combination thereof:

1. A language mechanism for restricting direct access to some of the object's components.
2. A language construct that facilitates the bundling of data with the methods (or other functions) operating on that data.

Design a MyURLShortner built on top of ShortURLFinal (given in a module) with some changes

- Change getShortURL to have both alphabets and numbers
- Change the URLPrefix to my website (myurlshortner.com)



Timestamp 10.37

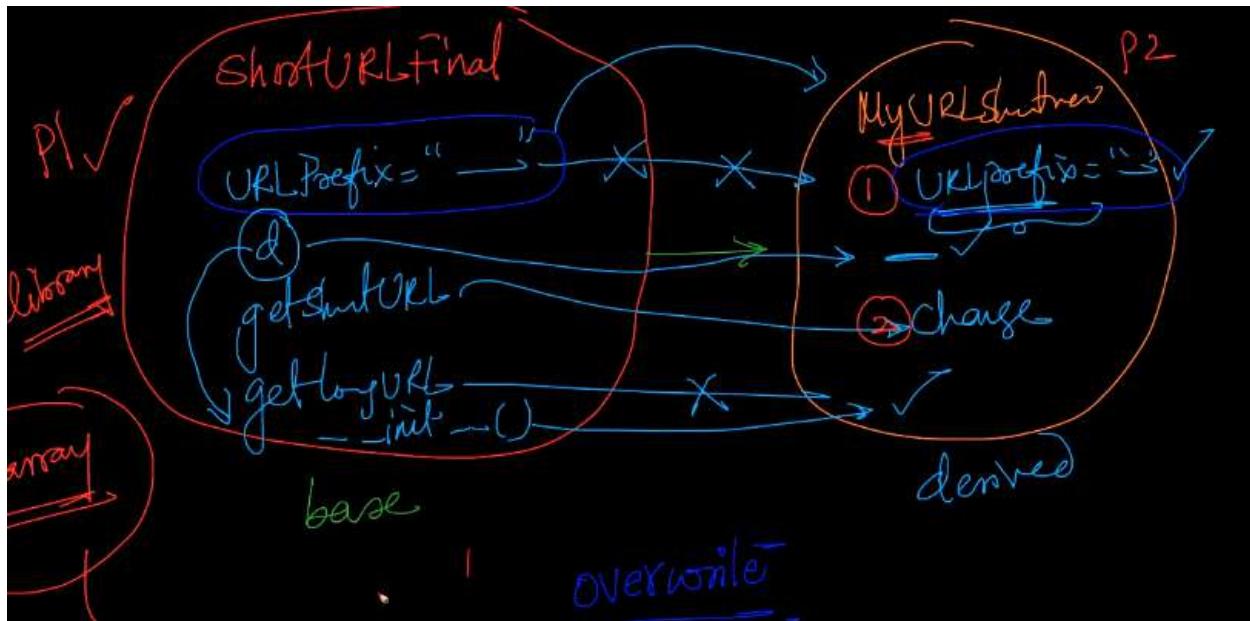
- Let's say we have implemented a class as shown and now we want to do some modifications as specified. We use inheritance for that as shown below.
- We derive all the functionalities in base class into our derived class. We do this by specifying base class in function definition as shown below .

```

1 # Lets define MyURLShortner based on ShortURLFinal
2
3 # Inheritance: baseclass, derived class [https://docs.python.org/3/tutorial/classes.html#inheritance]
4 class MyURLShortner(ShortURLFinal):
5     URLPrefix = "www.myurlshortner.com/" # overriding as same name as base-class
6
7
8     # given a long URL, get a short URL
9     def getShortURL(self, longURL): # overriding as same name as base-class, use both digits and lowercase
10        # length = random value in 6-10
11        l = random.randint(6,10);
12
13
14        # generate random characters into a string of length l
15        chars = string.ascii_lowercase + string.digits # both digits and lowercase
16        shortURL = ''.join(random.choice(chars) for i in range(l))
17
18
19        # check if this string is already present in dict d
20        if shortURL in self.d:
21            return getShortURL(longURL);
22        else:
23            self.d[shortURL] = longURL;
24
25
26        r = self.URLPrefix + shortURL
27        return r;
28
29    # getLongURL and dict "d" not changed

```

Here MyURLShortner is derived from ShortURLFinal base class.



Timestamp 19.08

As shown above we derive the properties and methods from base class and we override the methods that we want to modify.

```

1 m1 = MyURLShortner(); # base-class constructor is executed
2
3 print(m1.d)
4
{}
```

```

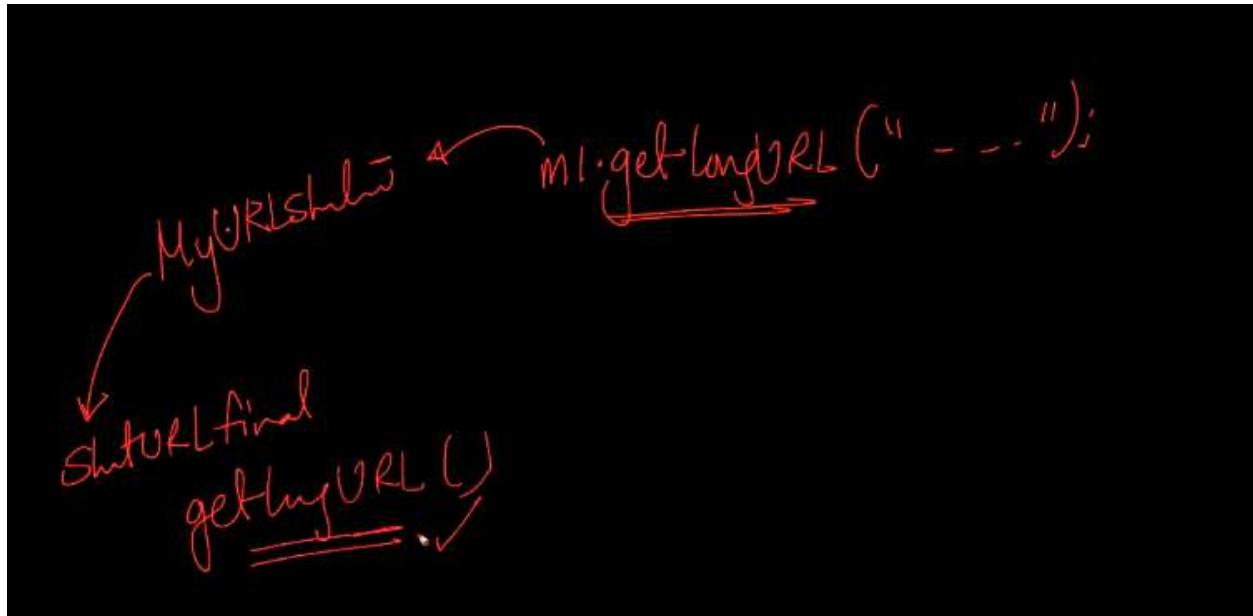
1 print(m1.getShortURL("amazon.com"))
2 print(m1.getShortURL("google.com"))
```

www.myurlshortner.com/mt2e9h
www.myurlshortner.com/ymnsyufq3

```

1 print(m1.d)

{'mt2e9h': 'amazon.com', 'ymnsyufq3': 'google.com'}
```



Timestamp 23.25

When we create an object m1 and since we haven't defined a constructor in derived class the constructor in base class gets executed as shown above.

14.7 Example: OOP for representing Shapes

Below is an example of object oriented programming for representing shapes.

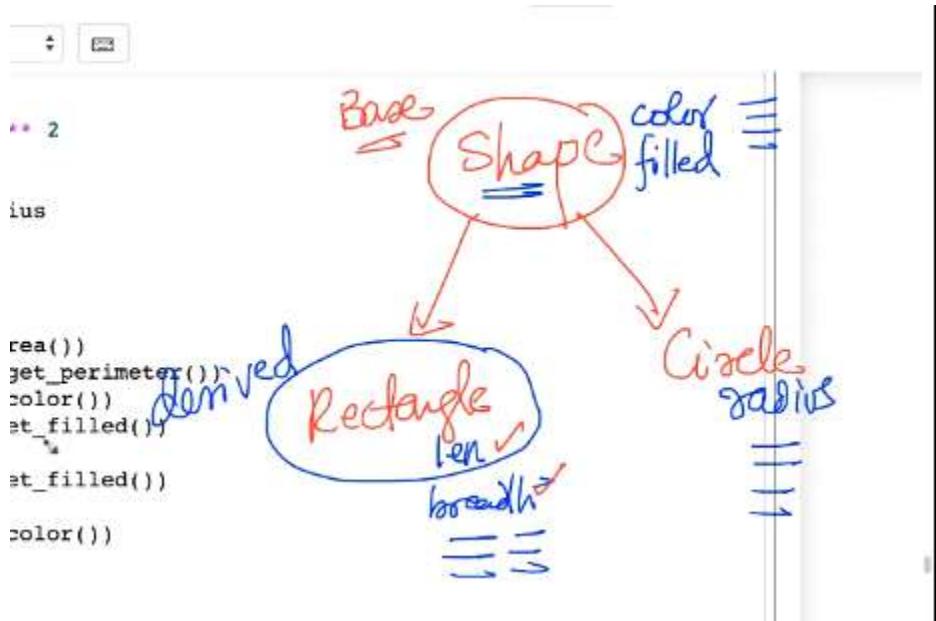
```
1 # example from geometry: [Source: https://overiq.com/python-101/inheritance/]
2 import math
3
4 class Shape:
5
6     def __init__(self, color='black', filled=False):
7         self.__color = color
8         self.__filled = filled
9
10    def get_color(self):
11        return self.__color
12
13    def set_color(self, color):
14        self.__color = color
15
16    def get_filled(self):
17        return self.__filled
18
19    def set_filled(self, filled):
20        self.__filled = filled
21
22
```

```
23 class Rectangle(Shape):
24
25     def __init__(self, length, breadth):
26         super().__init__()
27         self.__length = length
28         self.__breadth = breadth
29
30     def get_length(self):
31         return self.__length
32
33     def set_length(self, length):
34         self.__length = length
35
36     def get_breadth(self):
37         return self.__breadth
38
39     def set_breadth(self, breadth):
40         self.__breadth = breadth
41
42     def get_area(self):
43         return self.__length * self.__breadth
44
45     def get_perimeter(self):
46         return 2 * (self.__length + self.__breadth)
47
```

```

49 class Circle(Shape):
50     def __init__(self, radius):
51         super().__init__()
52         self._radius = radius
53
54     def get_radius(self):
55         return self._radius
56
57     def set_radius(self, radius):
58         self._radius = radius
59
60     def get_area(self):
61         return math.pi * self._radius ** 2
62
63     def get_perimeter(self):
64         return 2 * math.pi * self._radius
65

```



Timestamp 6.56

- Classes Rectangle, Circle inherited from base class Shape

```

1 ### object is the base class for all classes in Python
2
3 class MyClass(object): # object is the base-class by default and implicitly.
4     pass
5
6
7 # __new__() : creates a new object and calls the __init__()
8 # __init__() : default constructor
9 # __str__() : write code to convert object into string for printing.

```

```

1 a = [1,2,3,4];
2 print(type(a))
3
4 print(a)

```

<class 'list'>
[1, 2, 3, 4]

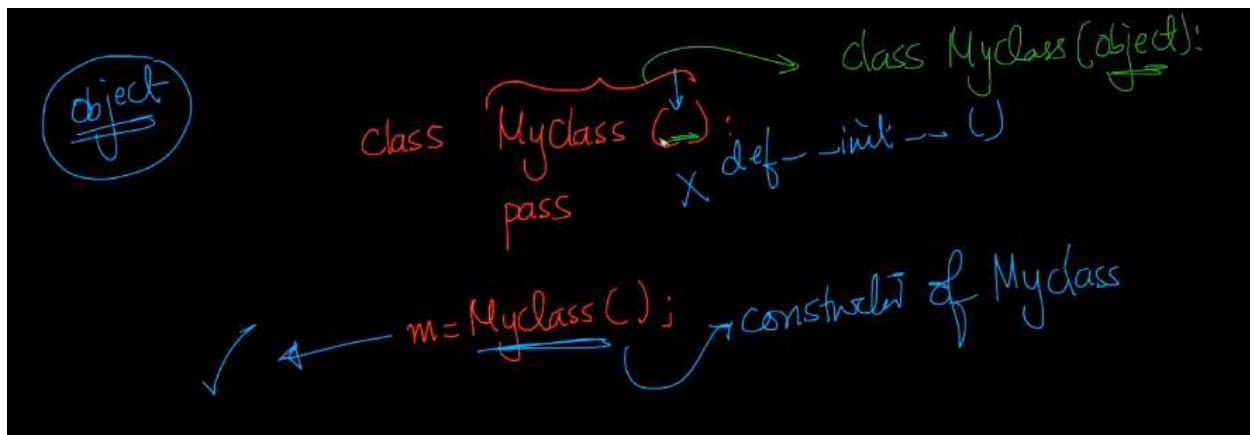
```

] 1 class ClassWithStr(): #default object is the base class
2     def __str__(self):
3         return "any string representation of the object of this class that we want"
4
5 c1 = ClassWithStr();
6 print(c1)

```

any string representation of the object of this class that we want

Timestamp 7.58



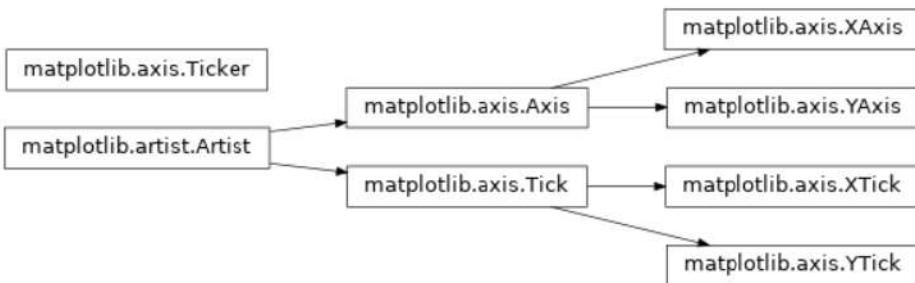
Timestamp 10.26

- Objectclass is baseclass for all the classes in python,every class in python inherits from object class
- Object class has the below functionalities .

1. `__new__()` : creates a new object and calls the `__init__()`
2. `__init__()` : default constructor
3. `__str__()` : write code to convert object into string for printing

Classes for the ticks and x and y axis.

Inheritance



- The above flowchart gives a nice intuition about inheritance.

14.8 How is OOP typically used in an ML role:

- Using existing Classes.
- Reading documentation to understand how to use a function/class/module.
- Fixing code bugs and understanding error messages.
- Extending existing classes to modify some functionality in an existing class
- Working with Software engineers to build some ML classes for them to use in the larger software.
- Do not perform OOD without understanding it well. Typically done by senior engineers/architects. A good beginner's book:
<https://learning.oreilly.com/library/view/head-first-design/0596007124/>

14.9 Multiple Inheritance

<https://docs.python.org/3/tutorial/classes.html#multiple-inheritance>

```
class DerivedClassName(Base1, Base2, Base3):
```

```
....
```

```
1 # toy-example: Modifications on https://overiq.com/python-101/inheritance-and-polymorphism-in-python/
2
3 class A:
4     def explore(self):
5         print("explore in A called")
6 class B:
7     def search(self):
8         print("search in B called")
9
10    def explore(self):
11        print("explore in B called")
12
13 class C:
14     def discover(self):
15         print("discover() in C called")
16
17 class D(A, B, C): # multiple inheritance
18     def test(self):
19         print("test() in D called")
20
21
22 d_obj = D()
23 d_obj.explore()
24 d_obj.search()
25 d_obj.discover()
26 d_obj.test()
```



```
explore in A called
search in B called
discover() in C called
test() in D called
```

Timestamp 4.26

Multiple inheritance is a feature of some object-oriented computer programming languages in which an object or class can **inherit** characteristics and features from more than one parent object or parent class.

In the above example you can clearly see that class D is inheriting from base classes A,B,C. The order is important because While inheriting from another class, the interpreter needs a way to resolve the methods that are being called via an instance. Thus we need the method resolution order

```
# toy-example: Diamond inheritance a.k.a. Deadly diamond

class A:
    def explore(self):
        print("explore in A called")

class B(A):
    def explore(self):
        print("explore in B called")

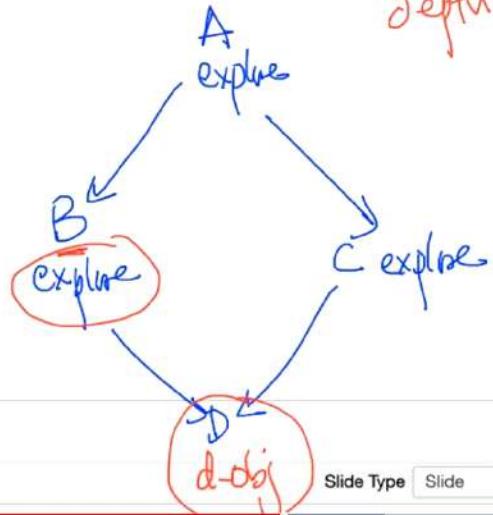
class C(A):
    def explore(self):
        print("explore in C called")

class D(B, C): # multiple inheritance
    pass;

d_obj = D()
d_obj.explore()

explore in B called
```

Same-class
L → R
Depth-first



Timestamp 9.37

There is a problem called the deadly diamond problem with Multiple inheritance as shown above.

14.10 Polymorphism (with real world examples)

- Different forms
- Operator level Polymorphism: $2+3$, "abc" + "def"
The + operator is performing addition operation and also string concatenation
- Function level Polymorphism: `len([1,2,3])`, `len ("abcdef")`, `len({1,2,3,4})`
The function `len()` accepts lists, strings, set but it returns the length.

```
1 print(len([1,2,3]));
2 print(len("abcdef"))
3 print(len({1,2,3,4}))
```

3
6
4

- Below is an example of Class level Polymorphism

Timestamp 2.34

```
1 #class level Polymorphism
2
3 class A:
4     def p(self):
5         return "function p in A"
6
7 class B:
8     def p(self):
9         return "function p in B"
10
11 a = A();
12 b = B();
13
14 for i in (a,b):
15     print(i.p()) # the function that runs depends on the object type making this code much more elegant and crisp
16
17
18 print("#####")
19
20 x=a;
21 print(x.p());
22
23 x=b;
24 print(x.p());
```

function p in A
function p in B

function p in A
function p in B

- Below is an example where polymorphism and inheritance both are implemented, Class Shape is inherited by Rectangle and Circle classes and class level polymorphism is used to iterate over objects belonging to different classes..

Timestamp 9.18

```
1 # Polymorphism + Inheritance
2
3 # example seen earlier: [Source: https://overiq.com/python-101/inheritance-and-polymorphism-in-python/]
4 import math
5
6 class Shape:
7
8     def __init__(self, color='black', filled=False):
9         self.__color = color
10        self.__filled = filled
11
12    def get_color(self):
13        return self.__color
14
15    def set_color(self, color):
16        self.__color = color
17
18    def get_filled(self):
19        return self.__filled
20
21    def set_filled(self, filled):
22        self.__filled = filled
23
24
25 class Rectangle(Shape):
26
27     def __init__(self, length, breadth):
28         super().__init__()
29         self.__length = length
30         self.__breadth = breadth
31
32     def get_length(self):
33         return self.__length
34
35     def set_length(self, length):
36         self.__length = length
37
38     def get_breadth(self):
39         return self.__breadth
40
41     def set_breadth(self, breadth):
42         self.__breadth = breadth
43
44     def get_area(self):
45         return self.__length * self.__breadth
46
47     def get_perimeter(self):
48         return 2 * (self.__length + self.__breadth)
49
```

```
51 class Circle(Shape):
52     def __init__(self, radius):
53         super().__init__()
54         self.__radius = radius
55
56     def get_radius(self):
57         return self.__radius
58
59     def set_radius(self, radius):
60         self.__radius = radius
61
62     def get_area(self):
63         return math.pi * self.__radius ** 2
64
65     def get_perimeter(self):
66         return 2 * math.pi * self.__radius
67 s = Shape();
68 r = Rectangle(10,20);
69 c = Circle(2);
70
71 for i in (s, r,c):
72     print(i.get_color())
73
74 for i in (r,c):
75     print(i.get_area())
```

```
black
black
black
200
12.566370614359172
```

```
1 # Polymorphism + Inheritance [inbuilt-ds]
2
3 d = {'a':1, 'b':2}
4 l = [1,2,3,4]
5 s = {1,2,3,4}
6
7 for i in (d,l,s):
8     print(i) # polymorphism + inheritance [__str__ from object]
9
```

```
{'a': 1, 'b': 2}
[1, 2, 3, 4]
{1, 2, 3, 4}
```

Timestamp 15.53

- We can observe the implementation of polymorphism and inheritance using inbuilt data structures as well.

Strings and Regex

- Focus: Basics of strings and regex in Python + Simple problem solving.
- Prereq: Basic knowledge of Strings and Regex in Python + previous code-sessions.
- Reference for basics:
 - <https://docs.python.org/3/howto/regex.html>
 - <https://docs.python.org/3/library/re.html>
 - https://www.w3schools.com/python/python_strings.asp
 - <https://www.geeksforgeeks.org/python-strings/>

A **regular expression** matches a broad or specific text pattern, and is strictly read left-to-right. It is input as a text string itself, and will compile into a mini program built specifically to identify that pattern. That pattern can be used to match, search, substring, or split text.

Here's a complete list of the metacharacters; their meanings will be discussed in the rest of this HOWTO.

. ^ \$ * + ? { } [] \ | ()

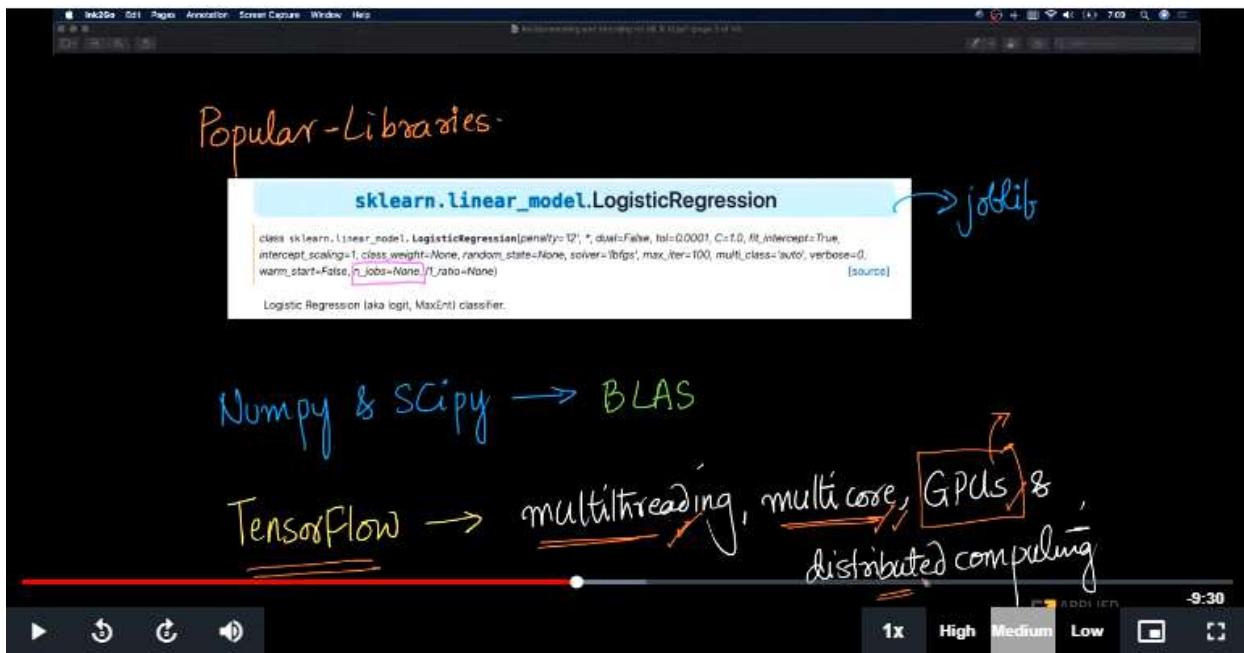
The first metacharacters we'll look at are [and]. They're used for specifying a character class, which is a set of characters that you wish to match. Characters can be listed individually, or a range of characters can be indicated by giving two characters and separating them by a '-'. For example, [abc] will match any of the characters a, b, or c; this is the same as [a-c], which uses a range to express the same set of characters. If you wanted to match only lowercase letters, your RE would be [a-z].

Metacharacters are not active inside classes. For example, [akm\$] will match any of the characters 'a', 'k', 'm', or '\$'; '\$' is usually a metacharacter, but inside a character class it's stripped of its special nature.

You can match the characters not listed within the class by complementing the set. This is indicated by including a '^' as the first character of the class. For example, [^5] will match any character except '5'. If the caret appears elsewhere in a character class, it does not have special meaning. For example: [5^] will match either a '5' or a '^'.

- A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.
- RegEx can be used to check if a string contains the specified search pattern.
- Python has a built-in package called re, which can be used to work with Regular Expressions.
- The re module offers a set of functions that allows us to search a string for a match:

17.1 Applied Introduction and Overview



Timestamp: 8:02

Many popular machine learning libraries like sklearn, numpy, scipy and tensorflow uses some form of parallelization libraries. Sklearn uses joblib, numpy and script internally uses BLAS. Tensorflow can perform tasks using multithreading, multicore, it can even use GPUs and distributed computing.

```
In [7]: import time
def mean():

    #Sum using for loops. We can use inbuilt NumPy Sum operation for better speed.
    sum = 0
    n=d.size
    for i in range(n):
        sum +=d[i]

    #Mean
    mean = sum/n
    return mean

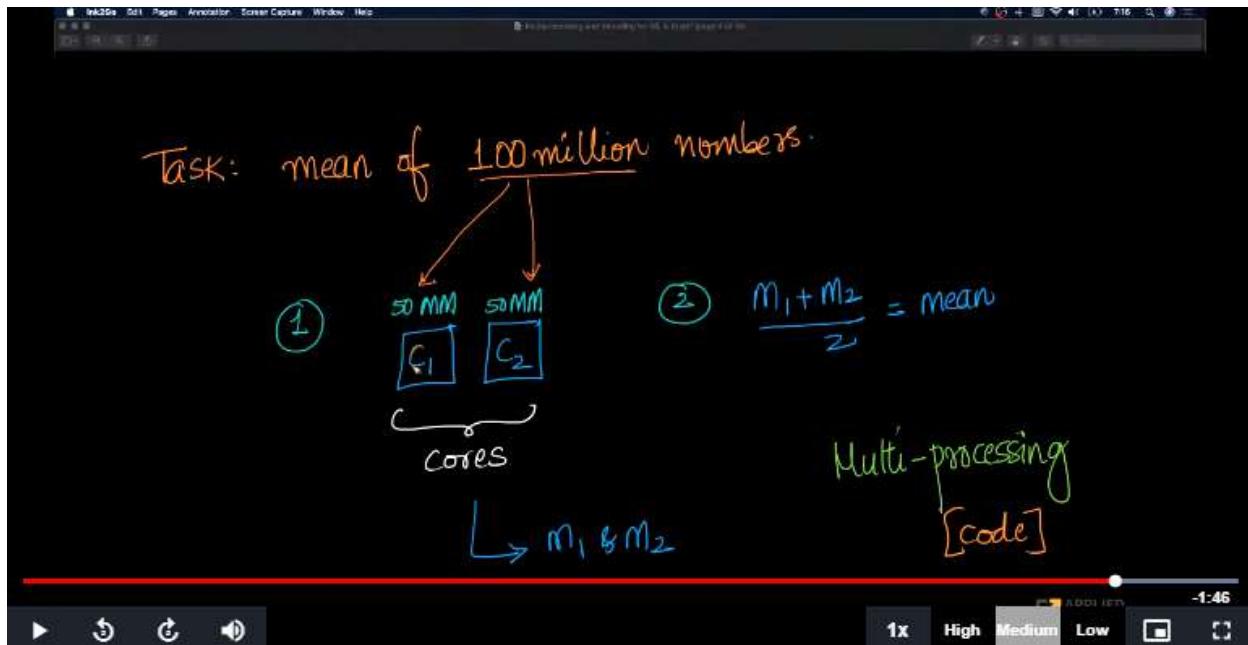
#Time the execution
start_time = time.time()
m = mean() # compute mean of 100MM numbers.
end_time = time.time()
print (end_time-start_time)
print(m)

20.98457510948181
0.49994777164597376
```

Multi-Processing Code

Timestamp: 14:58

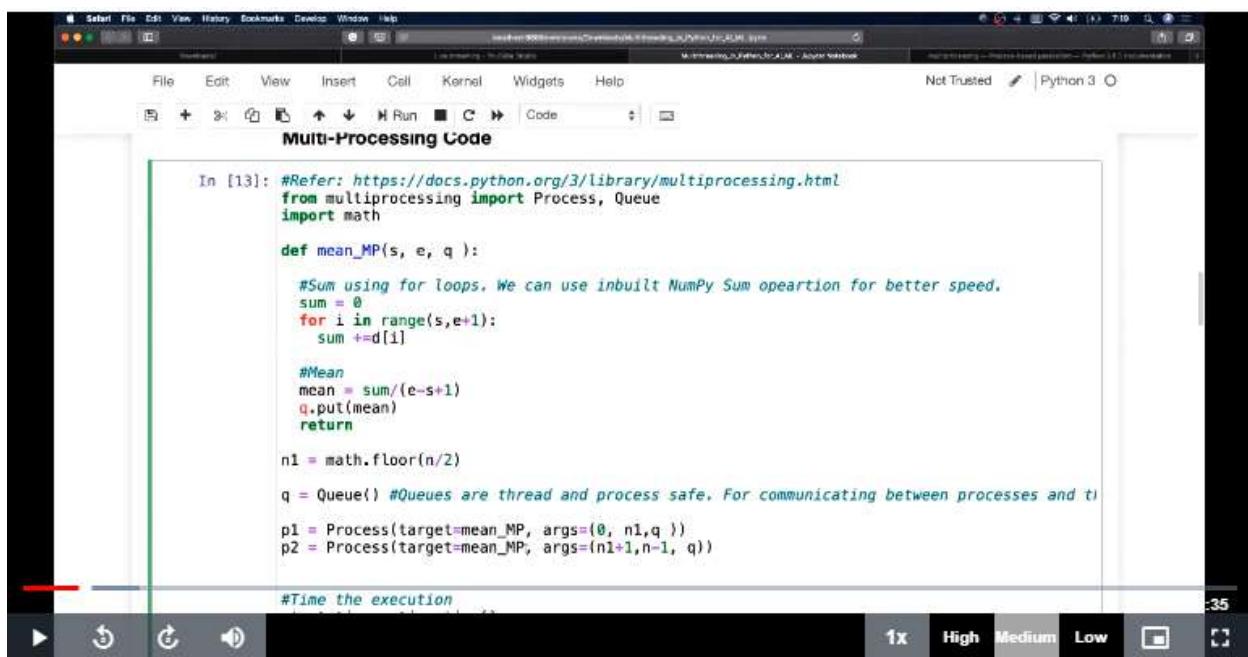
As shown in the above figure a program to calculate mean of 100 million numbers takes 20.98 sec. Note that this program runs on a single core of the processor with this code.



Timestamp: 15:46

We can parallelize the code as shown in the above figure by splitting the 100 million numbers into 50M and 50M and compute the means m1,m2 of both the parts using cpu cores c1 & c2 respectively. Finally, we can combine both the means as $(m1+m2)/2$. Since each core can run independently, both run parallely to calculate respective means, hence the time taken will be reduced largely.

17.2 Multiprocessing: Compute the mean of 100 Million numbers



The screenshot shows a Jupyter Notebook interface with a single code cell titled "Multi-Processing Code". The code implements a parallel processing algorithm to calculate the mean of an array of 100 million numbers. It uses the `multiprocessing` library to create two processes, `p1` and `p2`, which compute the mean of two halves of the array separately. The results are then combined to get the final mean. The code includes comments explaining the use of `for` loops for summing and the use of NumPy's `sum` function for better speed.

```
In [13]: #Refer: https://docs.python.org/3/library/multiprocessing.html
from multiprocessing import Process, Queue
import math

def mean_MP(s, e, q):
    #Sum using for loops. We can use inbuilt NumPy Sum opeartion for better speed.
    sum = 0
    for i in range(s,e+1):
        sum +=d[i]

    #Mean
    mean = sum/(e-s+1)
    q.put(mean)
    return

n1 = math.floor(n/2)

q = Queue() #Queues are thread and process safe. For communicating between processes and threads.

p1 = Process(target=mean_MP, args=(0, n1,q ))
p2 = Process(target=mean_MP, args=(n1+1,n-1, q))

#Time the execution
```

Timestamp: 1:10

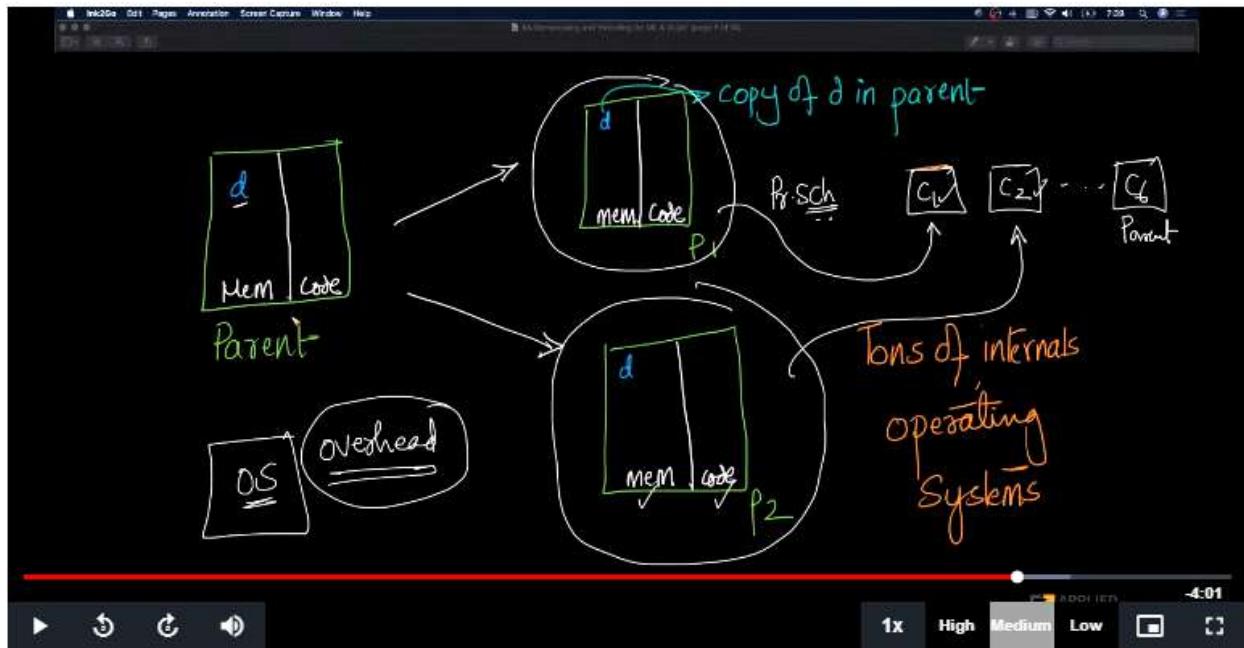
As shown in the above figure, we import Process, Queue from multiprocessing library which will help us create processes and the Queue using which the processes can communicate. We define a function mean_Mp that calculate the mean of the array elements from the start index s to the index e and store the mean corresponding to the part of the array in the queue.

We define the process p1 and p2 to compute the means of the first and second part of the array d respectively.

```
File Edit View Insert Cell Kernel Widgets Help  
Not Trusted Python 3  
q = Queue() #Queues are thread and process safe. For communicating between processes and threads  
p1 = Process(target=mean_MP, args=(0, n1,q ))  
p2 = Process(target=mean_MP, args=(n1+1,n-1, q))  
p1.start()✓ p2.start()  
p1.join() # Wait till p1 finishes ✓  
p2.join()  
m=0;  
while not q.empty():  
    m += q.get()  
m /= 2;  
end_time = time.time()  
print (end_time-start_time)  
print(m)  
11.001178979873657  
1x High Medium Low
```

Timestamp: 14:37

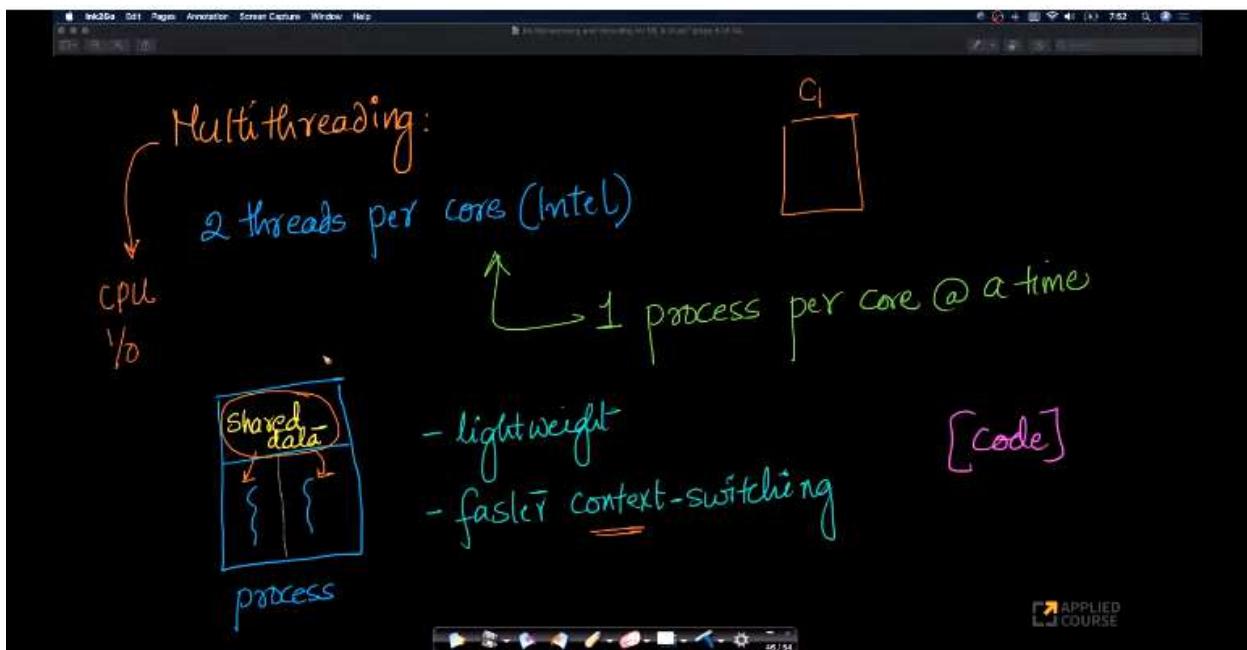
Notice in the above figure, we start the processes using `start()` and we combine them using `join()`. `join()` is used, so that the parent process which actually runs this code waits for the two child processes responsible for calculating the means. So once the child processes complete calculating means and storing in the queue, we then calculate the mean of the entire array in the parent process using `get` method of queue. Notice that the time taken for calculating the mean of the array using two process is 11.00 sec. This is a significant improvement over time taken by 1 process.



Timestamp: 18:44

Notice that while creating these child processes, the parent process copies d to both processes P_1 and P_2 . Hence both these processes has access to d for calculating the means. If we notice, with 2 processes it took 11.00 sec instead of $20.98/2=10.47$ sec (due to two cores). This additional time came from the OS overhead of creating two processes with the required code, memory and allocating them to individual cores. Note that during the entire time when both the child processes is computing the means, the parent process sleeps since it cannot proceed further unless the child processes finishes executing their code.

17.3 Multi-threading with code



Timestamp: 9:00

Using multi-threading, we can run upto 2 threads per core. Threads are known as light weight processes with faster context switching. Context switching is the time to create two process with all the required code and memory. Since threads are executed on the same core and use same shared data they have faster context-switching times.

Multithreading is helpful when we have two threads both executing CPU bound and I/O bound operations, so that when one thread is performing the I/O operations, the other thread can run its code on the cpu.

```

In [17]: #Refer: https://docs.python.org/3/library/threading.html
from threading import Thread

means = [0,0]
def mean_MT(s, e, threadNum):
    #Sum using for loops. We can use inbuilt NumPy Sum operation for better speed.
    sum = 0
    for i in range(s,e+1):
        sum +=d[i]
    #Mean
    mean = sum/(e-s+1)
    means[threadNum] = mean; # means is a shared varibile between the threads
    return
n1 = math.floor(n/2)

t1 = Thread(target=mean_MT, args=(0, n1,0)) # Third aparam is the thread number
t2 = Thread(target=mean_MT, args=(n1+1,n-1,1))

#Time the execution
start_time = time.time()

```

Timestamp: 12:56

Multithreading can be done in python using the threading library. Similar to how we have created processes we have created threads but in addition we give the thread number 0 and 1 to the function mean_MT and store the means in 0 and 1 st index of the means since both the threads share common memory.

```

#Time the execution
start_time = time.time()

t1.start()
t2.start()

t1.join() # Wait till t1 finishes
t2.join()

m = (means[0]+means[1])/2

end_time = time.time()
print (end_time-start_time)
print(m)

19.538660049438477
0.4999477716457993

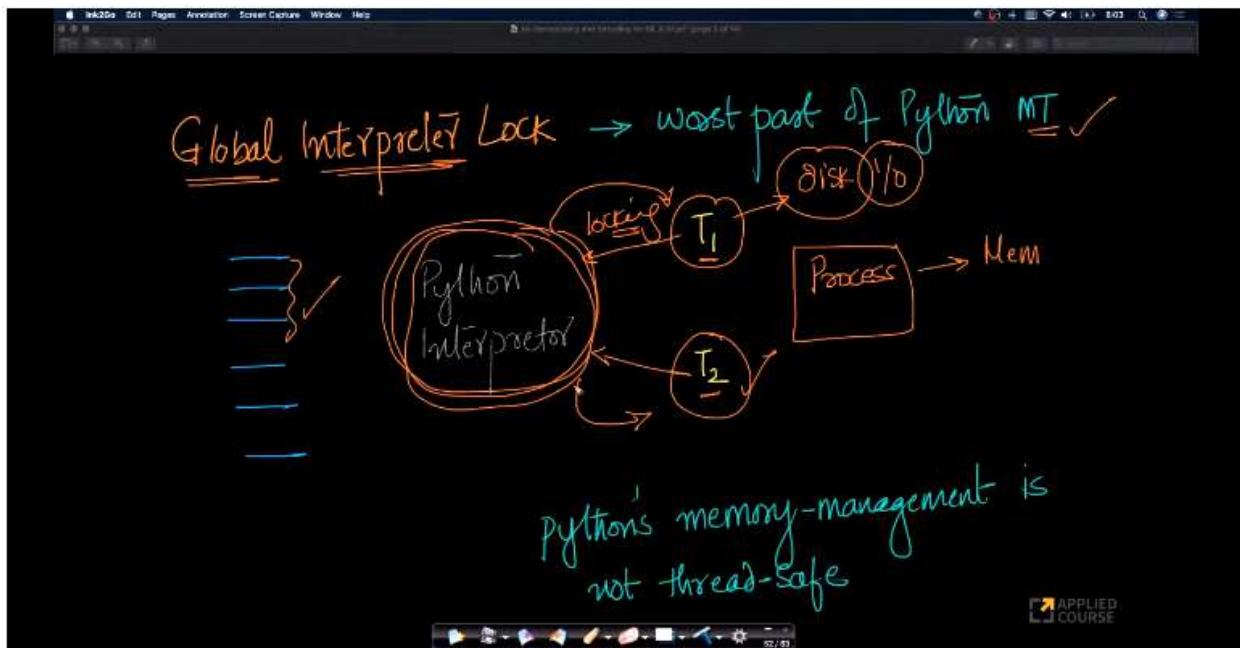
Joblib

Caching of function output values

```

Timestamp: 14:53

Notice that the speed from 1 thread to 2 threads for the given problem is very less. This is since both are CPU bound process there is not much parallelism going on. It is also due to a problem called GIL to be explained below.

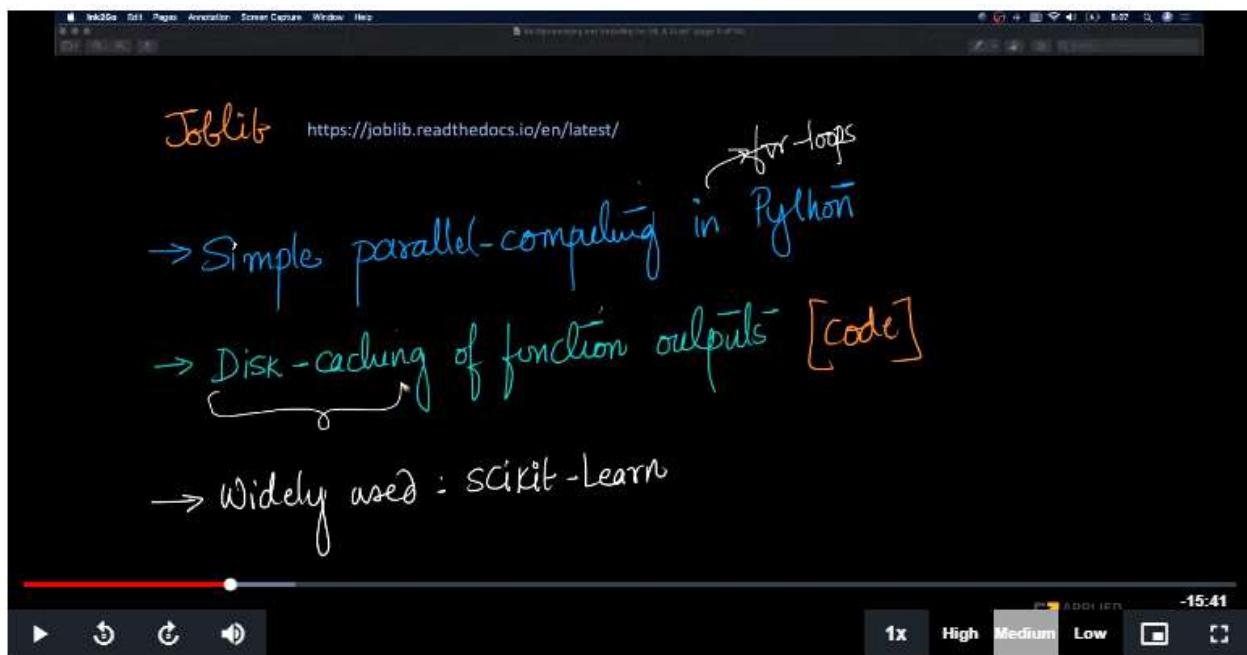


Timestamp: 20:14

While using multithreading in python, both the threads need access to the python interpreter to execute their code. But in python only one thread can access the python interpreter and the other thread is left waiting for the other thread to release the python interpreter. This is called global interpreter lock. This releasing happens when the thread is waiting for some I/O or disk operations.

Note that python's memory management is not thread safe meaning if different threads write to a single memory then it can get messed up.

17.4 Joblib: Simplified Parallel Programming in python



Timestamp: 3:14

Instead of using the raw multithreading and multiprocessing libraries we can work with joblib which is a simple parallel computing library in python. It can highly optimize for loops and it can use disk caching of function outputs.

So when we pass an input a to a function f and get output o, it stores that information that a function f when given input a gives output o. So the next time when someone calls the same function with the same input a, it searches whether it has seen the same input for the same function, if found in cache it returns the corresponding output o without computing it again. It is widely used in scikit learn.

```
def f(i):
    # some computations that take time
    x=10000
    p =1;
    for j in range(x):
        for k in range(j):
            p *= k

    return sqrt(i ** 2);

# Find sqrt of first n numbers
n=10;
start_time = time.time()

for i in range(n):
    f(i)

end_time = time.time()
print (end_time-start_time)
17.116321802139282

In [24]: from joblib import Parallel, delayed
```

start_time = time.time()

Timestamp: 9:01

Notice in the above code with no parallelization, it took 17.11 seconds to run the function $f(i)$ for i in $\text{range}(n)$.

```
# Find sqrt of first n numbers
n=10;

start_time = time.time()

for i in range(n):
    f(i)

end_time = time.time()
print (end_time-start_time)
17.116321802139282

In [24]: from joblib import Parallel, delayed
| start_time = time.time()
|
| a = Parallel(n_jobs=2)(delayed(f)(i ** 2) for i in range(n))
|
| # Why we need delayed(): https://stackoverflow.com/questions/42220458/what-does-the-delayed-function-do-in-joblib
|
| end_time = time.time()
| print (end_time-start_time)

9.58085012435913

In [25]: # Multi threading: GIL is an issue
```

Timestamp: 10:10

Notice that using joblib with 2 cores specified by n_jobs it took only 9.58 sec. Please refer the below link to learn about delayed. But what it does is to say the function we want to run and the inputs to run it with and let all the inputs be ready before we parallelize.

<https://stackoverflow.com/questions/42220458/what-does-the-delayed-function-do-when-used-with-joblib-in-python#:~:text=Delayed%20is%20a%20decorator%20that,an%20popped%20out%20as%20needed.>

The screenshot shows a Jupyter Notebook interface with several code cells and handwritten annotations. The first cell (In [24]) contains a single-line print statement that outputs 9.58 seconds. The second cell (In [25]) demonstrates multi-threading with 2 threads, showing a time of 17.12 seconds. The third cell (In [26]) shows a comparison with 6 jobs, which takes 17.34 seconds. Handwritten notes indicate that 1 thread (1pr, 1th) takes 17.12, 2 threads (2pr, 1th) take 9.58, and 6 threads (1pr, 2th) take 17.34 seconds. A note also mentions 'GIL & CPU bound'.

```
# May we need delayed(): https://stackoverflow.com/questions/42220458/what-does-the-delayed-function-do-when-used-with-joblib-in-python#:~:text=Delayed%20is%20a%20decorator%20that,an%20popped%20out%20as%20needed.

end_time = time.time()
print (end_time-start_time)

9.58085012435913

In [25]: # Multi threading: GIL is an issue
start_time = time.time()

a = Parallel(n_jobs=2, prefer="threads")(delayed(f)(i ** 2) for i in range(n))

end_time = time.time()
print (end_time-start_time)

17.342177867889404

In [26]:
# 6 jobs

from joblib import Parallel, delayed

start_time = time.time()

a = Parallel(n_jobs=6)(delayed(f)(i ** 2) for i in range(n))

# Why we need delayed(): https://stackoverflow.com/questions/42220458/what-does-the-delayed-function-do-when-used-with-joblib-in-python#:~:text=Delayed%20is%20a%20decorator%20that,an%20popped%20out%20as%20needed.

end_time = time.t
```

Timestamp: 17:41

We can also do multithreading using joblib as shown in the above figure. Note that the time taken for 2 threads is longer than the time taken with 1 thread. This is due to GIL and since both the threads are cpu bound.

```
a = Parallel(n_jobs=2, prefer="threads")(delayed(f)(i ** 2) for i in range(n))
end_time = time.time()
print (end_time-start_time)
17.342177867889404
```

```
In [26]: # 6 jobs
from joblib import Parallel, delayed
start_time = time.time()
a = Parallel(n_jobs=6)(delayed(f)(i ** 2) for i in range(n))
# Why we need delayed(): https://stackoverflow.com/questions/42220458/what-does-the-delayed-function-do
end_time = time.time()
print (end_time-start_time)
4.557589054107666
```

1pr, 1th → 17.12
2pr, 1th → 9.58
1pr, 2th → 17.34 sec
6pr, 1th → 4.55 sec
Cell 8 CPU bound

Timestamp: 18:35

Notice that with 6 processes it took 4.55 which is longer than $17.12/6=2.85$. This is because of the overhead of creating these 6 process and tracking them and context switch.

```
USE imdb;
SHOW TABLES;
DESCRIBE movies;
```

```
SELECT * FROM movies;
# more data transfer
```

#result-set: a set of rows that form the result of a query along with column-names and meta-data.

```
SELECT name,year FROM movies;
```

```
SELECT rankscore,name FROM movies;
#row order same as the one in the table
```

LIMIT:

```
SELECT name,rankscore FROM movies LIMIT 20;
```

```
SELECT name,rankscore FROM movies LIMIT 20 OFFSET 40;
```

ORDER BY:

```
# list recent movies first
```

```
SELECT name,rankscore,year FROM movies ORDER BY year DESC LIMIT 10;
```

```
# default:ASC
```

```
SELECT name,rankscore,year FROM movies ORDER BY year LIMIT 10;
```

the output row order maynot be same as the one in the table due to query optimzier and internal data-structres/indices.

```
*****
```

DISTINCT:

```
# list all genres of
SELECT DISTINCT genre FROM movies_genres;
```

```
# multiple-column DISTINCT
SELECT DISTINCT first_name, last_name FROM directors;
```

```
*****
```

WHERE:

```
# list all movies with rankscore>9
SELECT name,year,rankscore FROM movies WHERE rankscore>9 ;
```

```
SELECT name,year,rankscore FROM movies WHERE rankscore>9 ORDER BY rankscore
DESC LIMIT 20;
```

```
# Condition's outputs: TRUE, FALSE, NULL
```

```
# Comparison Operators: = , <> or != , < , <= , > , >=
SELECT * FROM movies_genres WHERE genre = 'Comedy';

SELECT * FROM movies_genres WHERE genre <> 'Horror';
```

```
NULL => doesnot-exist/unknown/missing
```

```
# "=" doesnot work with NULL, will give you an empty result-set.
SELECT name,year,rankscore FROM movies WHERE rankscore = NULL;
```

```
SELECT name,year,rankscore FROM movies WHERE rankscore IS NULL LIMIT 20;
```

```
SELECT name,year,rankscore FROM movies WHERE rankscore IS NOT NULL LIMIT 20;
```

```
*****
```

```
# LOGICAL OPERATORS: AND, OR, NOT, ALL, ANY, BETWEEN, EXISTS, IN, LIKE, SOME
```

```
# website search filters
```

```
SELECT name,year,rankscore FROM movies WHERE rankscore>9 AND year>2000;
```

```
SELECT name,year,rankscore FROM movies WHERE NOT year<=2000 LIMIT 20;
```

```
SELECT name,year,rankscore FROM movies WHERE rankscore>9 OR year>2007;
```

```
# will discuss about ANY and ALL when we discuss sub-queries
```

```
SELECT name,year,rankscore FROM movies WHERE year BETWEEN 1999 AND 2000;
```

```
#inclusive: year>=1999 and year<=2000
```

```
SELECT name,year,rankscore FROM movies WHERE year BETWEEN 2000 AND 1999;
```

```
#lowvalue <= highvalue else you will get an empty result set
```

```
SELECT director_id, genre FROM directors_genres WHERE genre IN ('Comedy','Horror');
```

```
# same as genre='Comedy' OR genre='Horror'
```

```
SELECT name,year,rankscore FROM movies WHERE name LIKE 'Tis%';
```

```
# % => wildcard character to imply zero or more characters
```

```
SELECT first_name, last_name FROM actors WHERE first_name LIKE '%es';
```

```
# first name ending in 'es'
```

```
SELECT first_name, last_name FROM actors WHERE first_name LIKE '%es%';
```

```
#first name contains 'es'
```

```
SELECT first_name, last_name FROM actors WHERE first_name LIKE 'Agn_s';
```

```
# '_' implies exactly one character.
```

If we want to match % or _, we should use the backslash as the escape character: \% and _

```
SELECT first_name, last_name FROM actors WHERE first_name LIKE 'L%' AND first_name  
NOT LIKE 'Li%';
```

Aggregate functions: Computes a single value on a set of rows and returns the aggregate
COUNT, MIN, MAX, SUM, AVG

```
SELECT MIN(year) FROM movies;
```

```
SELECT MAX(year) FROM movies;
```

```
SELECT COUNT(*) FROM movies;
```

```
SELECT COUNT(*) FROM movies where year>2000;
```

```
SELECT COUNT(year) FROM movies;
```

GROUP-BY

```
# find number of movies released per year
```

```
SELECT year, COUNT(year) FROM movies GROUP BY year;
```

```
SELECT year, COUNT(year) FROM movies GROUP BY year ORDER BY year;
```

```
SELECT year, COUNT(year) year_count FROM movies GROUP BY year ORDER BY year_count;  
# year_count is an alias.
```

```
# often used with COUNT, MIN, MAX or SUM.  
# if grouping columns contain NULL values, all null values are grouped together.
```

```
*****
```

HAVING:

```
# Print years which have >1000 movies in our DB [Data Scientist for Analysis]
```

```
SELECT year, COUNT(year) year_count FROM movies GROUP BY year HAVING year_count>1000;  
# specify a condition on groups using HAVING.
```

Order of execution:

1. GROUP BY to create groups
2. apply the AGGREGATE FUNCTION
3. Apply HAVING condition.

```
# often used along with GROUP BY. Not Mandatory.
```

```
SELECT name, year FROM movies HAVING year>2000;  
# HAVING without GROUP BY is same as WHERE
```

```
SELECT year, COUNT(year) year_count FROM movies WHERE rankscore>9 GROUP BY year HAVING year_count>20;
```

```
# HAVING vs WHERE  
## WHERE is applied on individual rows while HAVING is applied on groups.  
## HAVING is applied after grouping while WHERE is used before grouping.
```

```
*****
```

JOINS:

#combine data in multiple tables

For each movie, print name and the genres

```
SELECT m.name, g.genre from movies m JOIN movies_genres g ON m.id=g.movie_id LIMIT 20;
```

table aliases: m and g

natural join: a join where we have the same column-names across two tables.

#T1: C1, C2

#T2: C1, C3, C4

```
SELECT * FROM T1 JOIN T2;
```

```
SELECT * FROM T1 JOIN T2 USING (C1);
```

returns C1,C2,C3,C4

no need to use the keyword "ON"

Inner join (default) vs left outer vs right outer vs full-outer join.

T1: C1, C2, C3

```
SELECT m.name, g.genre from movies m LEFT JOIN movies_genres g ON m.id=g.movie_id LIMIT 20;
```

#LEFT JOIN or LEFT OUTER JOIN

#RIGHT JOIN or RIGHT OUTER JOIN

#FULL JOIN or FULL OUTER JOIN

#JOIN or INNER JOIN

NULL for missing counterpart rows.

3-way joins and k-way joins

```
SELECT a.first_name, a.last_name FROM actors a JOIN roles r ON a.id=r.actor_id JOIN movies m on m.id=r.movie_id AND m.name='Officer 444';
```

```
#Practical note about joins: Joins can be expensive computationally when we have large tables.
```

```
*****
```

Sub-Queries or Nested Queries or Inner Queries

```
# List all actors in the movie Schindler's List  
#https://www.imdb.com/title/tt0108052/fullcredits/?ref\_=tt\_ov\_st\_sm
```

```
SELECT first_name, last_name from actors WHERE id IN  
    ( SELECT actor_id from roles WHERE movie_id IN  
        (SELECT id FROM movies where name='Schindler's List')  
    );
```

```
# Syntax:  
SELECT column_name [, column_name ]  
FROM table1 [, table2 ]  
WHERE column_name OPERATOR  
(SELECT column_name [, column_name ]  
FROM table1 [, table2 ]  
[WHERE])
```

```
# first the inner query is executed and then the outer query is executed using the output values  
in the inner query
```

```
# IN, NOT IN, EXISTS, NOT EXISTS, ANY, ALL, Comparison operators
```

```
#EXISTS returns true if the subquery returns one or more records or NULL  
# ANY operator returns TRUE if any of the subquery values meet the condition.  
# ALL operator returns TRUE if all of the subquery values meet the condition.
```

```
SELECT * FROM movies where rankscore >= ALL (SELECT MAX(rankscore) from movies);  
# get all movies whose rankscore is same as the maximum rankscore.
```

```
# e.g: rankscore <> ALL(...)
```

```
# https://en.wikipedia.org/wiki/Correlated_subquery
```

```
*****
```

Data Manipulation Language: SELECT, INSERT, UPDATE, DELETE

```
INSERT INTO movies(id, name, year, rankscore) VALUES (412321, 'Thor', 2011, 7);
```

```
INSERT INTO movies(id, name, year, rankscore) VALUES (412321, 'Thor', 2011, 7), (412322, 'Iron Man', 2008, 7.9), (412323, 'Iron Man 2', 2010, 7);
```

```
# INSERT FROM one table to another using nested sub query:
```

```
https://en.wikipedia.org/wiki/Insert\_\(SQL\)#Copying\_rows\_from\_other\_tables
```

```
*****
```

```
# UPDATE Command
```

```
UPDATE <TableName> SET col1=val1, col2=val2 WHERE condition
```

```
UPDATE movies SET rankscore=9 where id=412321;
```

```
# Update multiple rows also.
```

```
# Can be used along with Sub-queries.
```

```
*****
```

```
#DELETE
```

```
DELETE FROM movies WHERE id=412321;
```

```
# Remove all rows: TRUNCATE TABLE TableName;
```

```
# Same as select without a WHERE Clause.
```

```
*****
```

Data Definition Language

```
CREATE TABLE language ( id INT PRIMARY, lang VARCHAR(50) NOT NULL);
```

```
# Datatypes: https://www.journaldev.com/16774/sql-data-types
```

```
# Constraints: https://www.w3schools.com/sql/sql\_constraints.asp
```

NOT NULL - Ensures that a column cannot have a NULL value

UNIQUE - Ensures that all values in a column are different

PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - Uniquely identifies a row/record in another table

CHECK - Ensures that all values in a column satisfies a specific condition

DEFAULT - Sets a default value for a column when no value is specified

INDEX - Used to create and retrieve data from the database very quickly

ALTER: ADD, MODIFY, DROP

ALTER TABLE language ADD country VARCHAR(50);

ALTER TABLE language MODIFY country VARCHAR(60);

ALTER TABLE langauge DROP country;

Removes both the table and all of the data permanently.
DROP TABLE Tablename;

DROP TABLE TableName IF EXISTS;

#<https://dev.mysql.com/doc/refman/8.0/en/drop-table.html>

TRUNCATE TABLE TableName;

as discussed earlier same as DELETE FROM TableName;

Data Control Language for DB Admins.

https://en.wikipedia.org/wiki/Data_control_language

<https://dev.mysql.com/doc/refman/8.0/en/grant.html>

<https://dev.mysql.com/doc/refman/8.0/en/revoke.html>

20.1 Introduction to IRIS dataset and 2D scatter plot

Exploratory Data Analysis - Definition

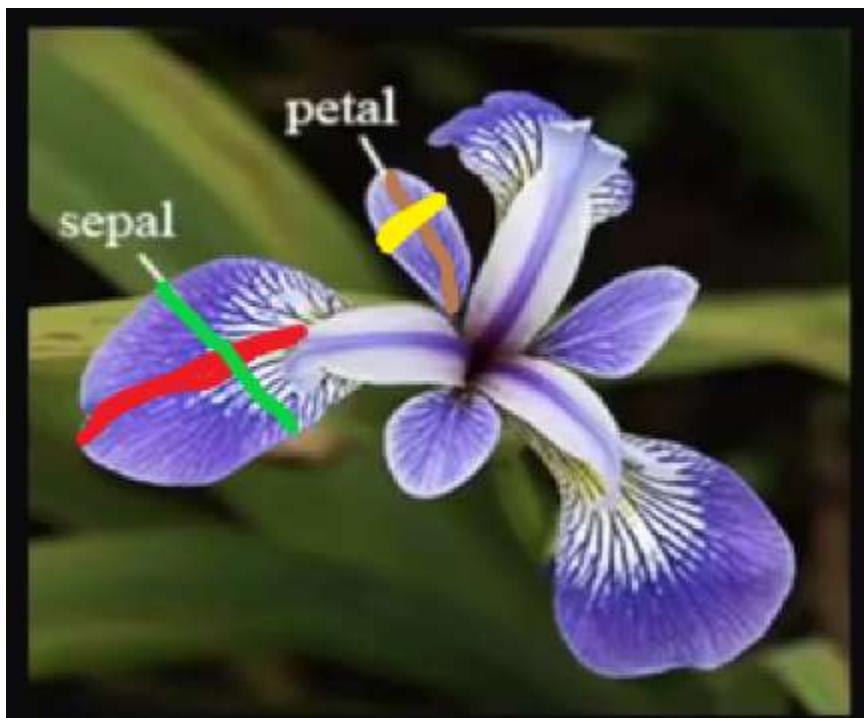
- Exploratory Data Analysis is an approach of analyzing the datasets to summarize the main characteristics, often with the visual methods.
- Whether a statistical model is used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.

IRIS Dataset Description

The IRIS dataset consists of 5 columns. They are

- 1) Sepal Length
- 2) Sepal Width
- 3) Petal Length
- 4) Petal Width
- 5) Species

Below is the image of the IRIS flower that was shown at the timestamp 4:57 in the video.



- | | | |
|-------------|---|----------------------------|
| Red Line | → | Indicates the Sepal Length |
| Green Line | → | Indicates the Sepal Width |
| Brown Line | → | Indicates the Petal Length |
| Yellow Line | → | Indicates the Petal Width |

The first 4 features denote different dimensions of the IRIS flower and they all accept only continuous numerical values. The ‘Species’ feature denotes the class to which the flower belongs to and it is a discrete feature. We have 3 classes of IRIS flower and they are

- 1) Setosa
- 2) Versicolor
- 3) Virginica

Our main objective here is to classify a given species into any one of these 3 categories.

Reading a dataset into a Pandas Dataframe

read_csv()

- The function read_csv() is used to read the dataset present in a CSV file into a pandas dataframe.
- The reason for using pandas dataframes is because data importing and data manipulations are easier with the functions and attributes defined in Pandas module.

Below is an example on how to read the dataset into a pandas dataframe and it has been discussed at the timestamp 12:30 in the video.

```
] import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

'''download iris.csv from https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv'''
#Load Iris.csv into a pandas DataFrame.
iris = pd.read_csv("iris.csv")
```

shape and columns attributes

- The ‘shape’ attribute is used to return the number of rows and columns present in the dataframe. The result is returned in the form of a tuple with two values.
- The ‘columns’ attribute is used to return the names of all the columns of the dataframe.

Below is an example that shows us the usage of these two attributes and it was discussed starting from the timestamp 13:10.

```
# (Q) how many data-points and features?  
print (iris.shape)  
  
(150, 5)  
  
#(Q) What are the column names in our dataset?  
print (iris.columns)  
  
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',  
       'Species'],  
      dtype='object')
```

value_counts()

- Whenever a categorical column in a pandas dataframe consists of multiple categories, then in order to get the count of rows/data points for each category, we use the **value_counts()** function in Pandas.
- It is recommended to use this function only for categorical features, but not for the continuous numerical features, as it is not a good practice.

Below is an example that illustrates the usage of **value_counts()** function and it was discussed at the timestamp 14:45.

```
#(Q) How many data points for each class are present?  
#(or) How many flowers for each species are present?  
  
iris["Species"].value_counts()  
# balanced-dataset vs imbalanced datasets  
#Iris is a balanced dataset as the number of data points for every class is 50.  
  
Iris-versicolor    50  
Iris-virginica    50  
Iris-setosa        50  
Name: Species, dtype: int64
```

2-D Scatter Plot

- Scatter plots use dots to represent values for two different numerical variables. The position of each dot on the horizontal and the vertical axes indicate values for an individual data point.

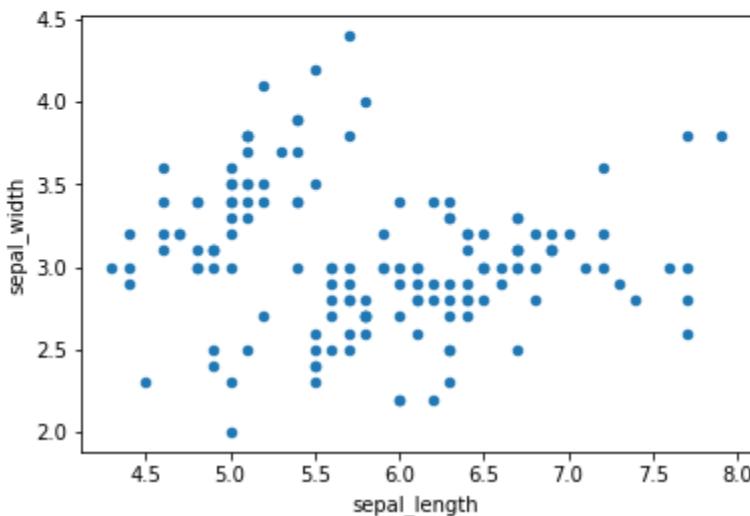
- Scatter plots are used to observe the relationships between two numeric variables.
- A scatter plot can be used when one of the two variables is an independent variable and the other is a dependent variable. Or we also can use when both the variables are independent.
- The dots in a scatter plot not only report the values of individual data points, but also the patterns when the data is taken as a whole.

Below is an example that illustrates how to plot a scatter plot using `plot()` function in Pandas and it was discussed at the timestamp 17:45.

```
: #2-D scatter plot:
#ALWAYS understand the axis: labels and scale.

iris.plot(kind='scatter', x='sepal_length', y='sepal_width') ;
plt.show()

#cannot make much sense out it.
#What if we color the points by their class-label/flower-type.
```



In this example, we are building a scatter plot using the `plot()` function in Pandas. The value passed to the 'kind' parameter indicates what kind of plot we need. The value 'scatter' indicates scatter plot. The values passed to the 'x' and 'y' parameters are those column names with which we want to build the scatter plots.

Note: The point of intersection of axes in coordinate geometry is the origin, but in this plot, it is not the origin. It is because in plotting, for both the features (ie., 'sepal_length' and 'sepal_width'), the maximum and minimum values are obtained and the only region in between these limits is displayed in the plot.

For every plot, we should clearly observe the scale of the axes and the point of intersection of the axes and also the legends.

You can go through the documentation of `pandas.plot()` [here](#) to explore more about this function. The statement `plot.show()` is used to display our final resultant plot and it should be the last statement among all the plotting steps.

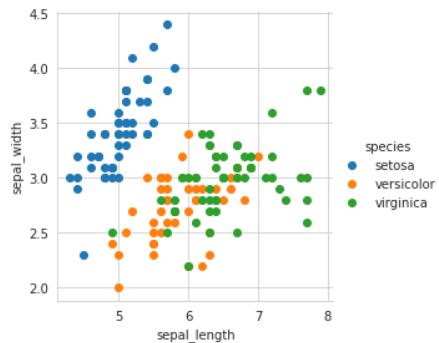
When we observe the above plot, we see all the points belonging to different classes are in the same color. So it is difficult for one to differentiate the points belonging to one class from the points belonging to other classes. So we shall apply color coding, so that points belonging to one class will have the same color and in this way, each class is assigned with a unique color.

Below is an example of such a plot which was discussed at the timestamp 20:50.

```
# 2-D Scatter plot with color-coding for each flower type/class.
# Here 'sns' corresponds to seaborn.
sns.set_style("whitegrid");
sns.FacetGrid(iris, hue="species", size=4) \
    .map(plt.scatter, "sepal_length", "sepal_width") \
    .add_legend();
plt.show();

# Notice that the blue points can be easily separated
# from red and green by drawing a Line.
# But red and green data points cannot be easily separated.
# Can we draw multiple 2-D scatter plots for each combination of features?
# How many combinations exist? 4C2 = 6.

/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:316: UserWarning: The `size` parameter has been renamed to `height`;
please update your code.
warnings.warn(msg, UserWarning)
```



In this example, we are using the seaborn library to build the same plot, but in a much more attractive and an understandable format.

Here the first statement is `sns.set_style()` which indicates how the background of our plot should look like. As we have set the style to ‘`whitegrid`’, the background is white in color with a grid(ie., the black horizontal and vertical lines). We also can use values like ‘`darkgrid`’, ‘`dark`’ or ‘`white`’ as the background styles. You can refer to the documentation of `set_style()` [here](#).

The next statement is `sns.FacetGrid()` which initializes a multi-plot grid for plotting conditional relationships. You always have to pass only a pandas dataframe as the data input and the value passed to the ‘`hue`’ parameter indicates the column name on the basis of whose values, we have to assign the colors. Here as we are passing ‘`species`’ as the value to the ‘`hue`’ parameter, as this parameter contains 3 values (ie., ‘Iris-setosa’, ‘Iris-versicolor’ and ‘Iris-virginica’), all the points belonging to the

'Iris-setosa' class will have one color, the points belonging to the 'Iris-versicolor' class will have other color and all the points belonging to the 'Iris-virginica' class will have another color.

The '**size**' parameter was deprecated and is now replaced by '**height**' and it takes only numerical values, which indicates the height of the plot. The more the value, the more large the plot would be.

Next is the **map()** function which takes a function name as the first argument and then applies that function to the following mentioned column names. Here it applied the **plt.scatter()** function in between the '**sepal_length**' and '**sepal_width**' columns.

The function **add_legend()** adds the labels to the plot about which color indicates which class of points.

When we look at the final plot obtained, we see the that the points belonging to 'Iris-setosa' class are easily separable by a line, whereas the points belong to the 'Iris-versicolor' and the 'Iris-virginica' classes are not easily separable by a line.

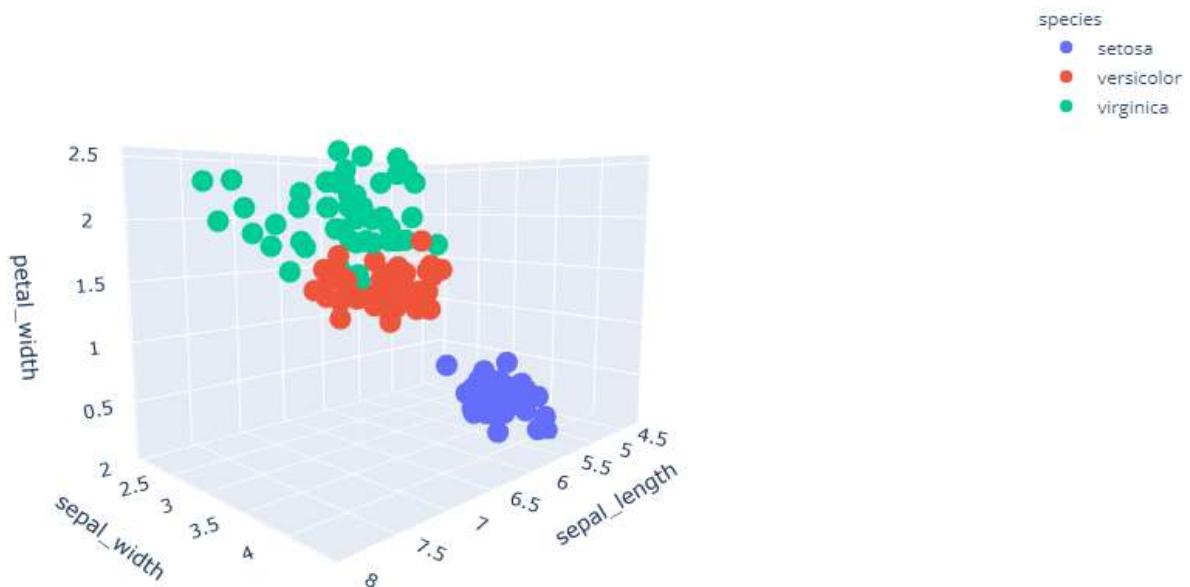
You can explore **sns.FacetGrid()** by referring to the documentation [here](#).

20.2 3D Scatter Plots

A 3D scatter plot also works the same way as a 2D scatter plot, but the only difference was in a 2D scatter plot, we have used only 2 features, whereas in a 3D scatter plot, we have to use 3 features (one on 'X' axis, one on 'Y' axis and the other on 'Z' axis).

For plotting, the maximum number of features we use is 3. Beyond that it is difficult to visualize and also represent it on a paper or by using any visualization software.

Below is an example of how a 3D scatter plot looks like when we use 'sepal_length', 'sepal_width' and 'petal_width'.



In this plot, we are representing 'sepal_length' on one axis, 'sepal_width' and 'petal_width' on two different axes.

20.3 Pairplots

As it is difficult for us to build scatter plots with more than 3 dimensions, we apply a hack here and this hack is known as pairplot. Here pair plots deal with plotting 2D scatter plots using all the 4 numerical features (taken 2 at a time). That is where we get 2D scatter plots with (SL, SW), (SW, SL), (SL, PL), (PL, SL), (SL, PW), (PW, SL), (SW, PL), (PL, SW), (PW, SW), (SW, PW), (PL, PW), (PW, PL) combinations. And we get histograms each for SL, SW, PL and PW features.

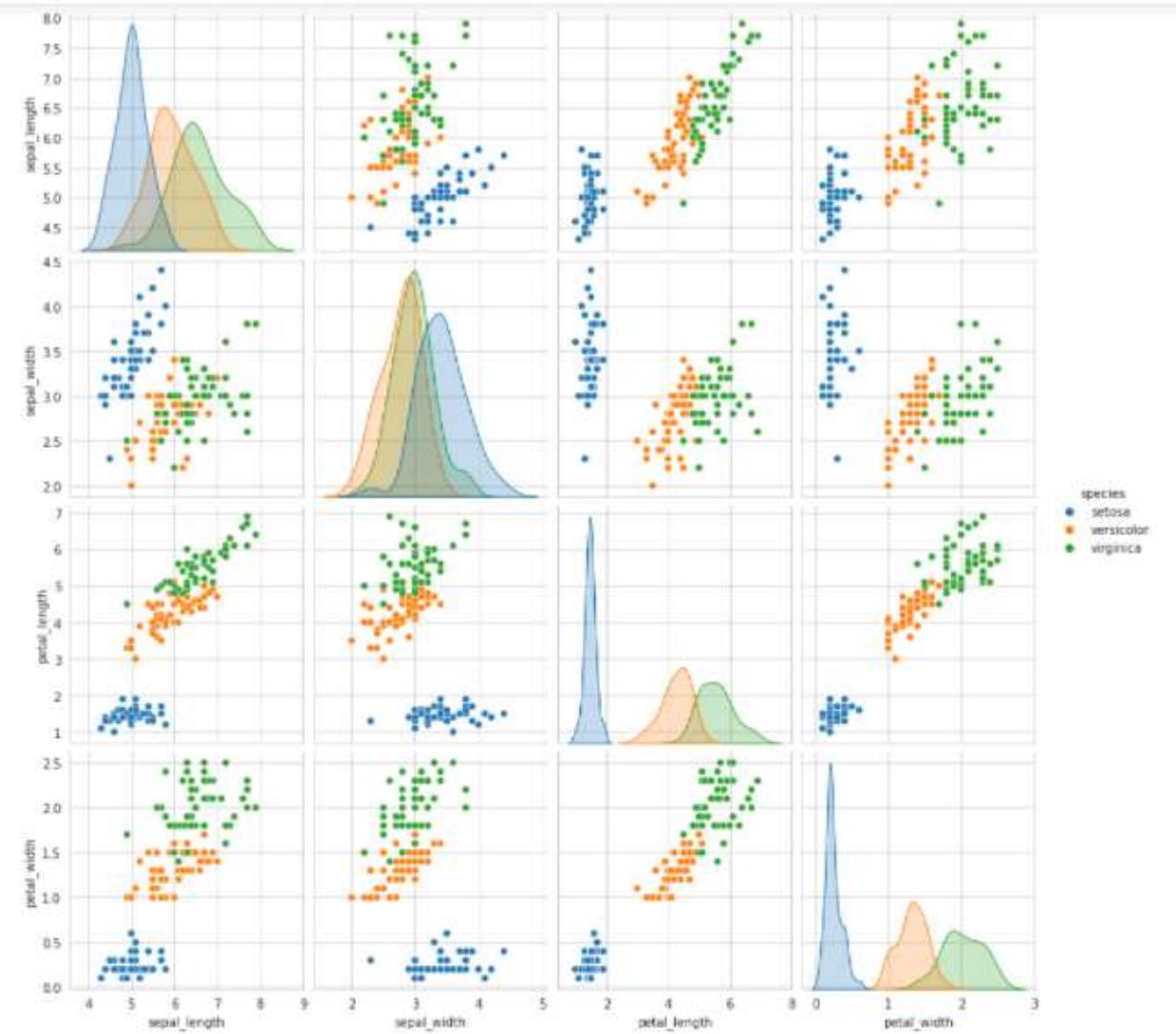
SL → Sepal Length
SW → Sepal Width
PL → Petal Length
PW → Petal Width

Here as we are using 4 numerical features for building pairplots,
Total number of scatter plots = $2^*4C_2 = 2^*6 = 12$
Total number of univariate plots(ie., histograms/PDF) = 4
Total number of plots obtained = $12 + 4 = 16$

The univariate plots occur in the diagonal positions among the plots. So let us look at the example that was discussed at the timestamp 0:05

```
# pairwise scatter plot: Pair-Plot
# Dis-advantages:
##Can be used when number of features are high.
##Cannot visualize higher dimensional patterns in 3-D and 4-D.
#Only possible to view 2D patterns.
plt.close();
sns.set_style("whitegrid");
sns.pairplot(iris, hue="species", size=3);
plt.show()
# NOTE: the diagonal elements are PDFs for each feature. PDFs are explained below.

/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:1912: UserWarning: The `size` parameter has been renamed to `height
`; please update your code.
warnings.warn(msg, UserWarning)
```



From the above plots, we could easily conclude that the ‘Petal_Length’ and the ‘Petal_Width’ features are much more useful in identifying the IRIS flowers of different classes. The ‘Iris-setosa’ flowers can be easily distinguished and are easily separable whereas in case of ‘Iris-versicolor’ and ‘Iris-virginica’, there is a little overlap and they cannot be distinguished perfectly all the time.

When we look at the code above, we are first setting the background style to ‘whitegrid’ and then in the `sns.pairplot()` function, we are passing the dataframe as the first argument. We are passing ‘species’ as the value for the ‘hue’ parameter which indicates the points belonging to each class of the ‘species’ column should be represented in a different color. The ‘size’ parameter denotes the height of the plot.

When we clearly observe the plots, we can see the ‘petal_length’ and the ‘petal_width’ features play a key role in separating the ‘Iris-setosa’ flowers easily from the other two species. As we are able to separate the ‘Iris-setosa’ species from the

other species easily, we can derive a condition from the scatter plot between ‘petal_length’ and ‘petal_width’ as

```
if petal_length<2 && petal_width<1:  
    then 'Iris-setosa'
```

Now for separating the points of ‘Iris-versicolor’ from the points of ‘Iris-virginica’, we can write the condition as

```
if peta_width<2 && petal_width>1 && petal_length<5 && petal_length>2.5:  
    then 'Iris-versicolor'
```

This condition still gives a few misclassifications, but in around 90-95% of the cases, it gives the correct classification.

So from the above plots, we can conclude that the ‘petal_length’ and the ‘petal_width’ features are majorly useful in classifying the points. Just by using these two features and simple if-else conditions, we could easily separate the flowers to a major extent. There are a few overlaps, but still it is accepted.

20.4 Limitations of Pairplots

As far as the number of dimensions in the dataset is small, we can go for pairplots. But if the number of dimensions is high, then we get a huge number of scatter plots and it is difficult to make analysis and draw conclusions from such a huge number of plots. This is a major disadvantage with the pairplots.

One way to get rid of this huge dimensionality is by reducing the number of dimensions. We have techniques like PCA, T-SNE, etc that reduce the dimensionality of the dataset and these techniques will be covered in this course in the future lectures.

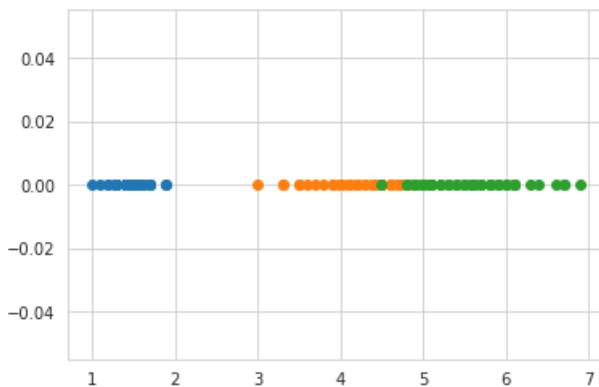
20.5 Histogram and Introduction to PDF (Probability Density Function)

1D Scatter plots

So far we have seen 2D scatter plots, 3D scatter plots and pairplots. Now let us look at 1D scatter plot. We shall plot a 1D scatter plot using the ‘petal_length’ column. We need at least 2 axes in order to represent a plot. So we shall take the ‘petal_length’ values as the ‘X’ coordinates and zeros as the ‘Y’ coordinates.

Below is the code and the 1D scatter plot graph that was discussed starting from the timestamp 0:35.

```
# What about 1-D scatter plot using just one feature?  
#1-D scatter plot of petal-length  
import numpy as np  
iris_setosa = iris.loc[iris["species"] == "setosa"];  
iris_virginica = iris.loc[iris["species"] == "virginica"];  
iris_versicolor = iris.loc[iris["species"] == "versicolor";  
#print(iris_setosa["petal_length"])  
plt.plot(iris_setosa["petal_length"], np.zeros_like(iris_setosa['petal_length']), 'o')  
plt.plot(iris_versicolor["petal_length"], np.zeros_like(iris_versicolor['petal_length']), 'o')  
plt.plot(iris_virginica["petal_length"], np.zeros_like(iris_virginica['petal_length']), 'o')  
  
plt.show()  
#Disadvantages of 1-D scatter plot: Very hard to make sense as points  
#are overlapping a lot.  
#Are there better ways of visualizing 1-D scatter plots?
```



In the above code, we are first separating the points belonging to each class into a separate data frame and then building the 1D scatter plots with them. As all the ‘Y’ coordinates are the same, we see all the points at the same level on the graph.

But the problem here with the 1D scatter plots is that we could see a huge overlap of the points belonging to the same classes. We do not know how many points of a particular class are present in an interval. In between 1 and 2 on the ‘X’ axis, we see a huge number of points of ‘Iris-setosa’ class. Also we see a huge number of points of ‘Iris-versicolor’ and ‘Iris-virginica’ classes overlapping in between the values 3.5 and 6 on the ‘X’ axis. As we aren’t clear about the number of points, we

could not draw any conclusions from the 1D scatter plots. Hence in order to perform univariate analysis, we go with another type of plot known as '**Histogram**'.

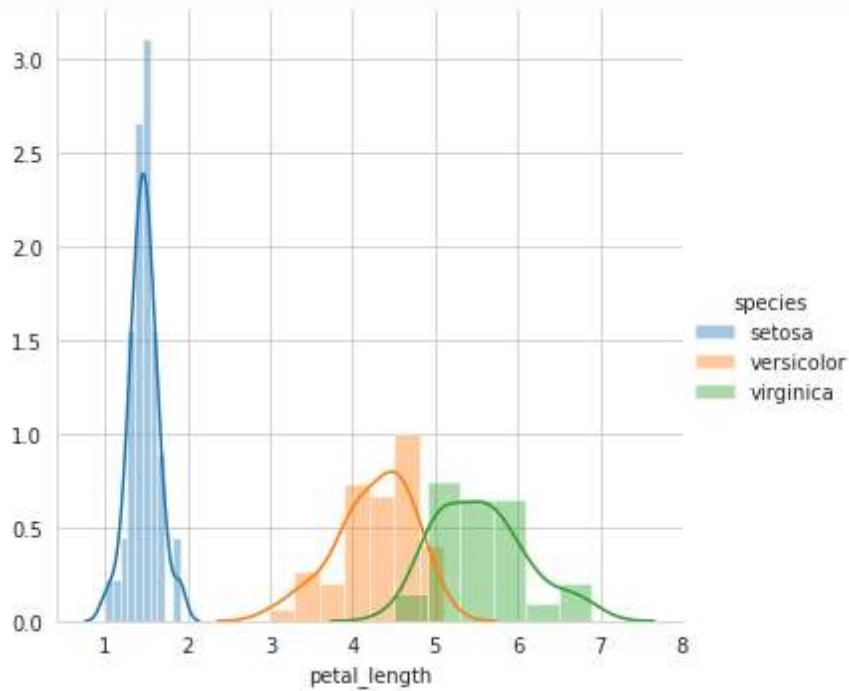
Histogram

- A Histogram is an accurate representation of the distribution of numerical data.
- To construct a histogram, the first step is to divide the entire range of values into a series of intervals and then count how many values fall into each interval.
- The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins must be adjacent and are often of equal size. Histogram is used when we have a large number of data points.
- In a histogram, the 'X' axis represents the class interval and the 'Y' axis represents the class/bin frequencies. The number of points present in a bin is called bin frequency. The width of each bin is called bin width and it is the same for all the bins.
- The choice of the bin-width determines the number of class intervals. The shape of the histogram gets affected by the bin-width and the starting point.
- PDF and Histogram are the best tools to be used for univariate analysis. PDF is the smoothed form of a histogram.
- To smooth the histogram, we use something called Kernel Density Estimation(KDE) which results in the curve we are seeing for the PDF.

Note: PDF and KDE will be discussed in much more detail in the Statistics chapter of the future lectures.

Below is the code snippet to build a distribution plot and it was discussed at the timestamp 5:00.

```
sns.FacetGrid(iris, hue="species", size=5) \
    .map(sns.distplot, "petal_length") \
    .add_legend();
plt.show();
```



When we use the 'petal_length' feature, we could see the 'Iris-setosa' class gets separated easily. We can make the below conclusions.

if PL<2:

Then 'Setosa'

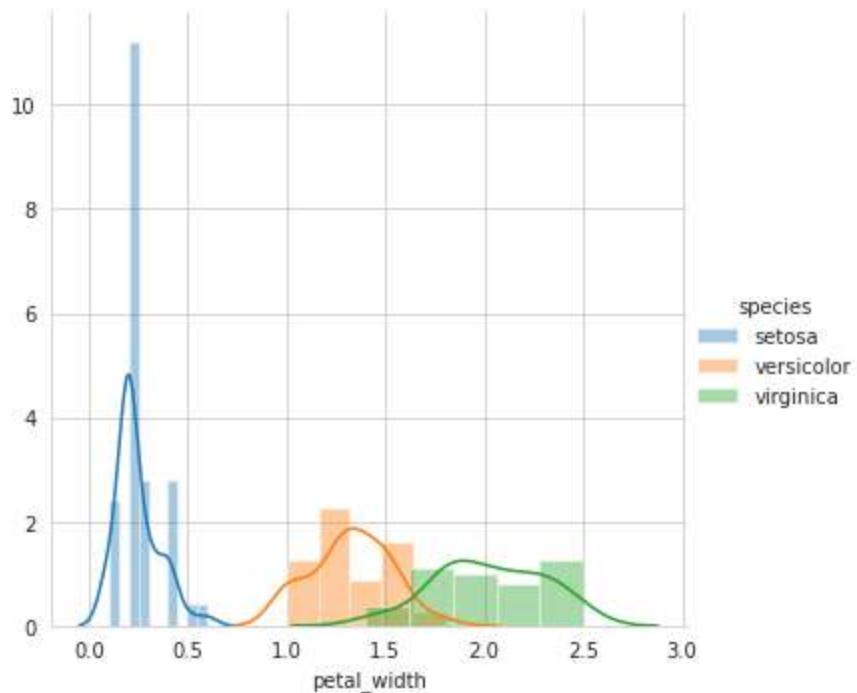
if PL < 4.7:

Then 'versicolor'

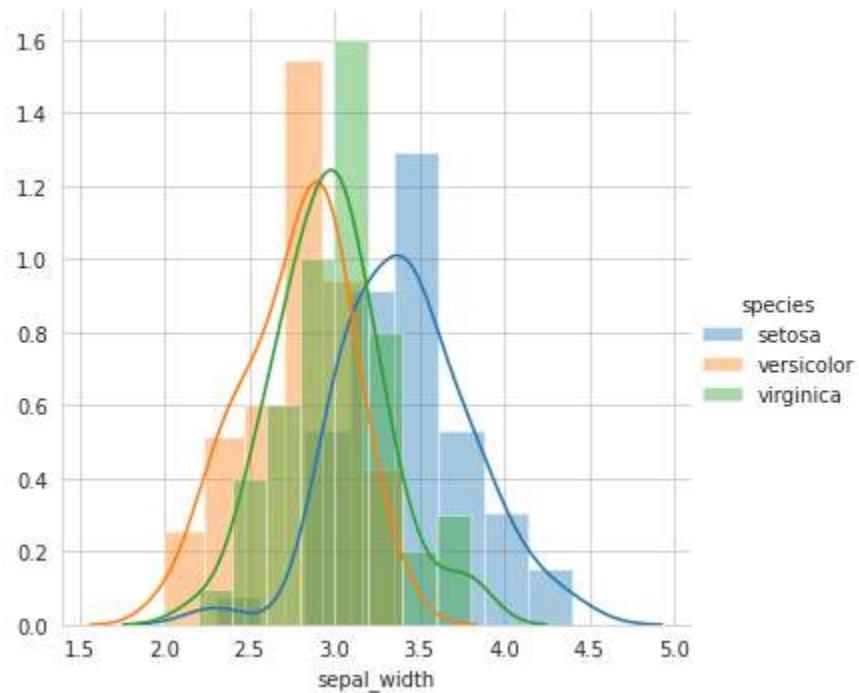
else:

'Virginica'

```
sns.FacetGrid(iris, hue="species", size=5) \
    .map(sns.distplot, "petal_width") \
    .add_legend();
plt.show();
```

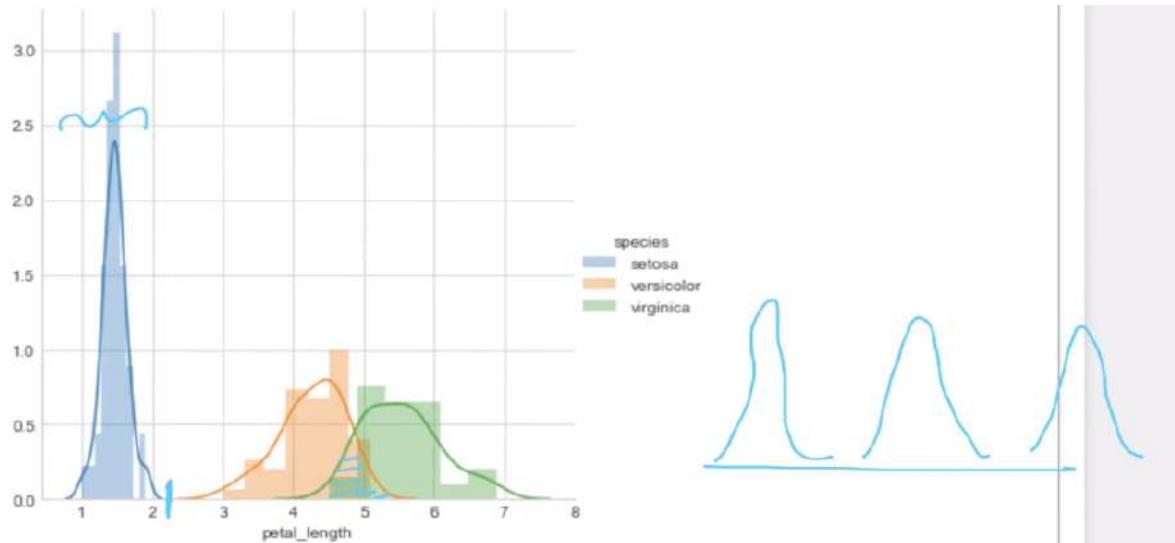


```
sns.FacetGrid(iris, hue="species", size=5) \
    .map(sns.distplot, "sepal_length") \
    .add_legend();
plt.show();
```

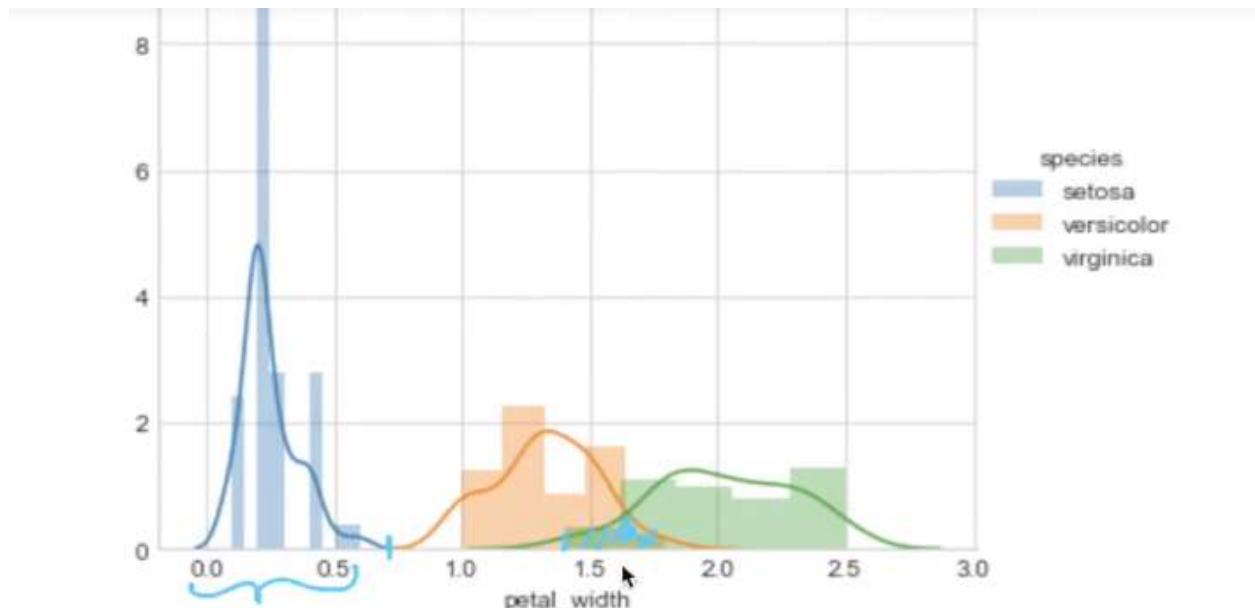


20.6 Univariate Analysis using density function

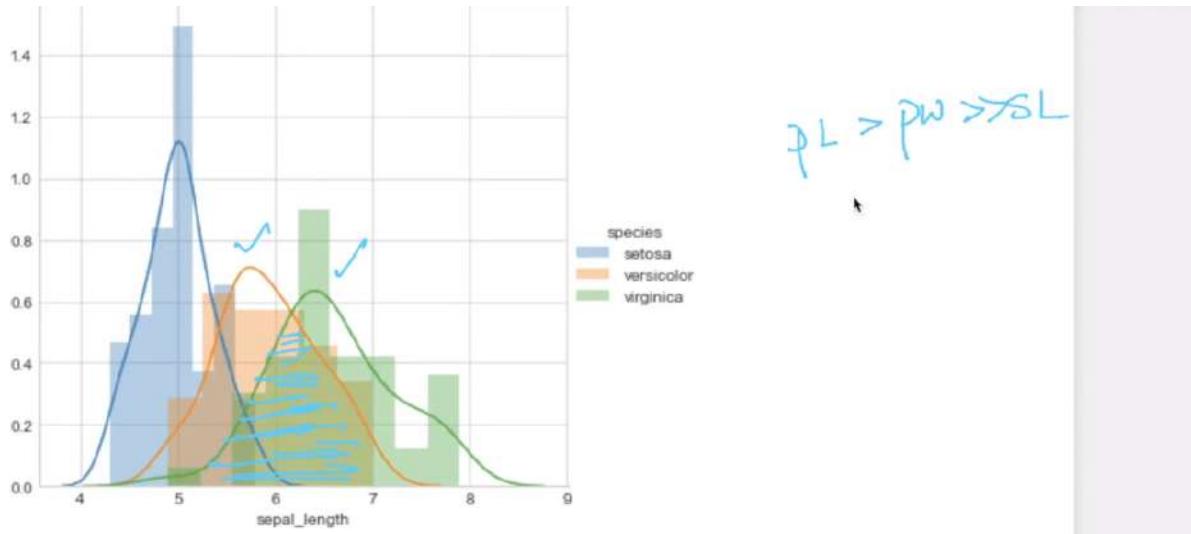
We use the univariate plots like PDFs, histograms to perform univariate analysis. Below are the plots of the univariate analyses of all the 4 features of Iris dataset and are discussed starting from the timestamp 0:50.



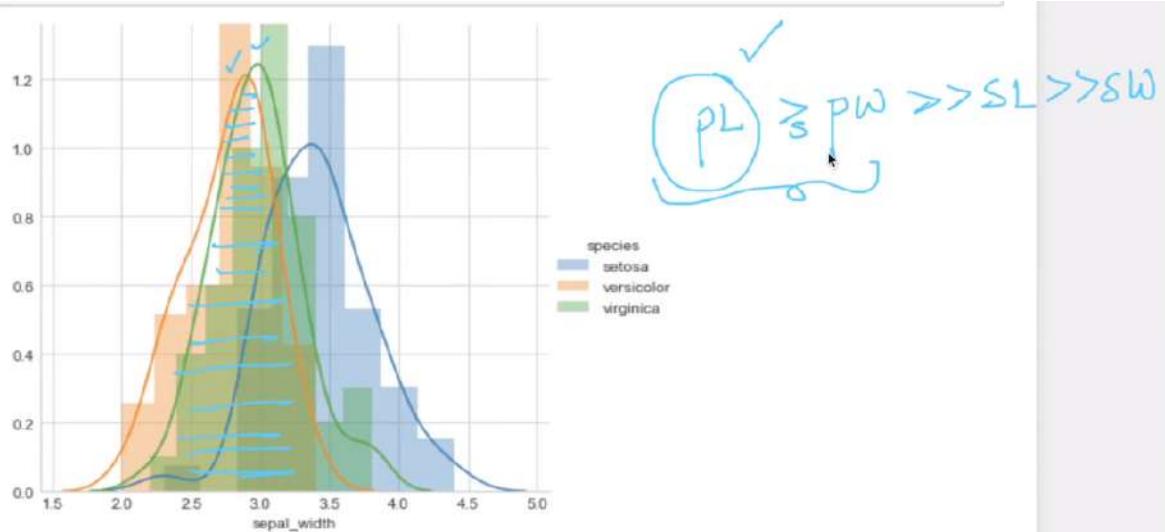
From the above plot, we can see that by using the 'petal_length' values, we could separate the 'Setosa' species from the other 2 species. We observe a little overlap between the 'Versicolor' and the 'Virginica' groups.



Even from the univariate analysis of the 'petal_width' column, we can clearly say that the 'Setosa' species can be separated from the other 2 species. Also we see a small overlap between the 'Versicolor' and 'Virginica' groups.



Here from the univariate analysis of the 'sepal_length', we observe there are overlaps not only between 'Versicolor' and 'Virginica', but also the 'Setosa' species points get overlapped with these two.



In the univariate analysis of 'sepal_width', the overlapped region is increased among all the 3 categories of species.

So we can say that in order to classify the points belonging to different species, the order of preference of the features, useful in decision making is

$PL > PW > SL > SW$

20.7 Cumulative Distribution Function (CDF)

- CDF value at a particular point ' x_1 ' on the 'X' axis denotes the probability/proportion of the points with the 'x' value less than or equal to ' x_1 '.
- The CDF value at a point 'x' is equal to the sum of probabilities of all values less than or equal to 'x'.

Below are the code snippet and the graphs for the CDF that were discussed starting from the timestamp 0:17.

```
# Plots of CDF of petal_length for various types of flowers.

# Misclassification error if you use petal_length only.

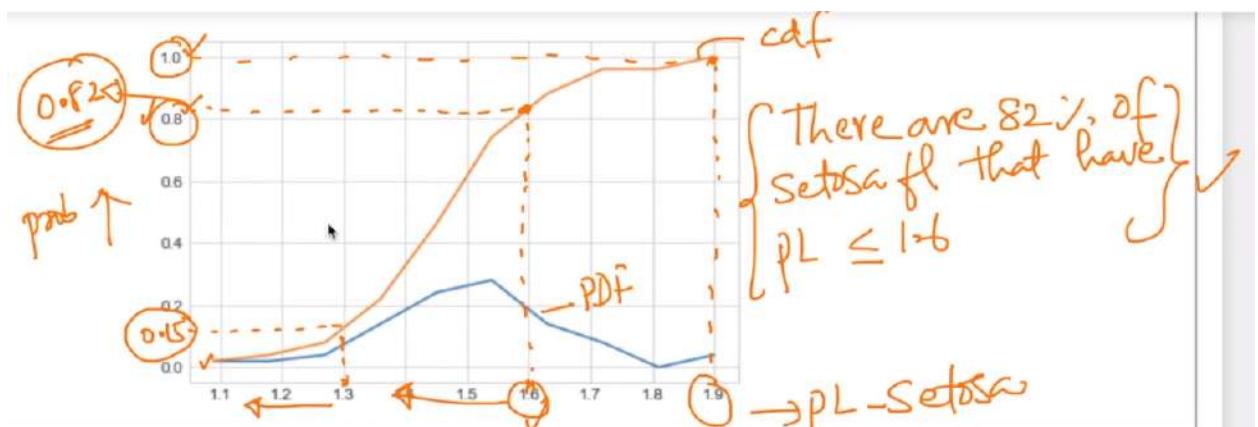
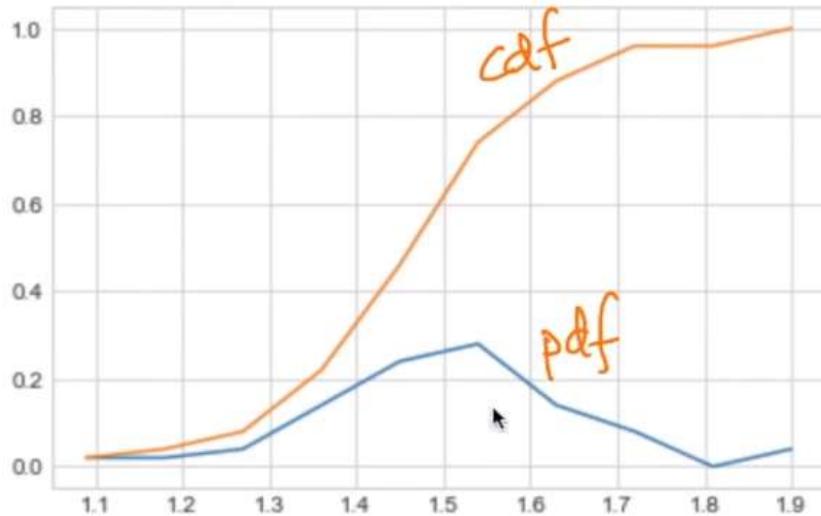
counts, bin_edges = np.histogram(iris_setosa['petal_length'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)

# virginica
counts, bin_edges = np.histogram(iris_virginica['petal_length'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)

#versicolor
counts, bin_edges = np.histogram(iris_versicolor['petal_length'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)

plt.show();

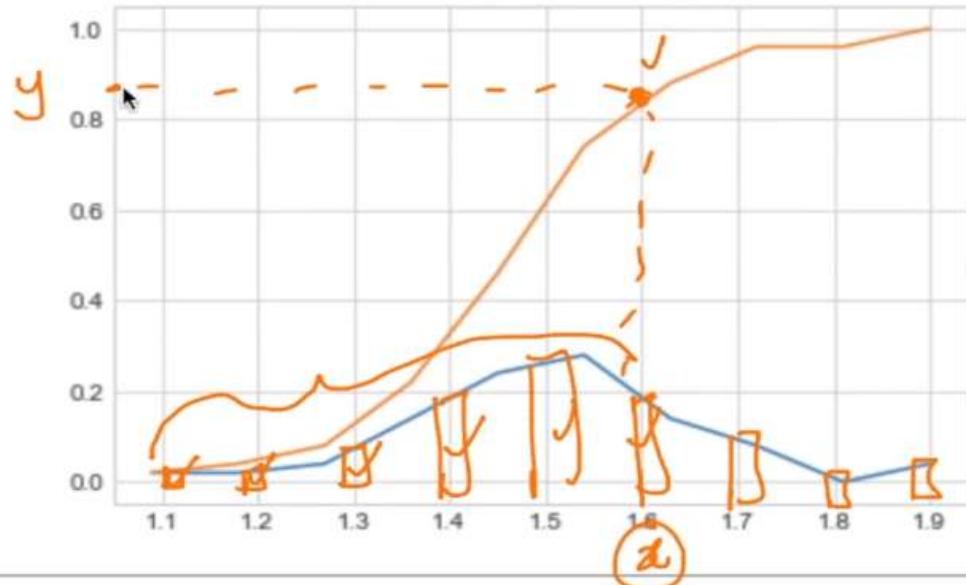
[ 0.02  0.02  0.04  0.14  0.24  0.28  0.14  0.08  0.   0.04]
[ 1.    1.09  1.18  1.27  1.36  1.45  1.54  1.63  1.72  1.81  1.9 ]
[ 0.02  0.1   0.24  0.08  0.18  0.16  0.1   0.04  0.02  0.06]
[ 4.5   4.74  4.98  5.22  5.46  5.7   5.94  6.18  6.42  6.66  6.9 ]
[ 0.02  0.04  0.06  0.04  0.16  0.14  0.12  0.2   0.14  0.08]
[ 3.    3.21  3.42  3.63  3.84  4.05  4.26  4.47  4.68  4.89  5.1 ]
```



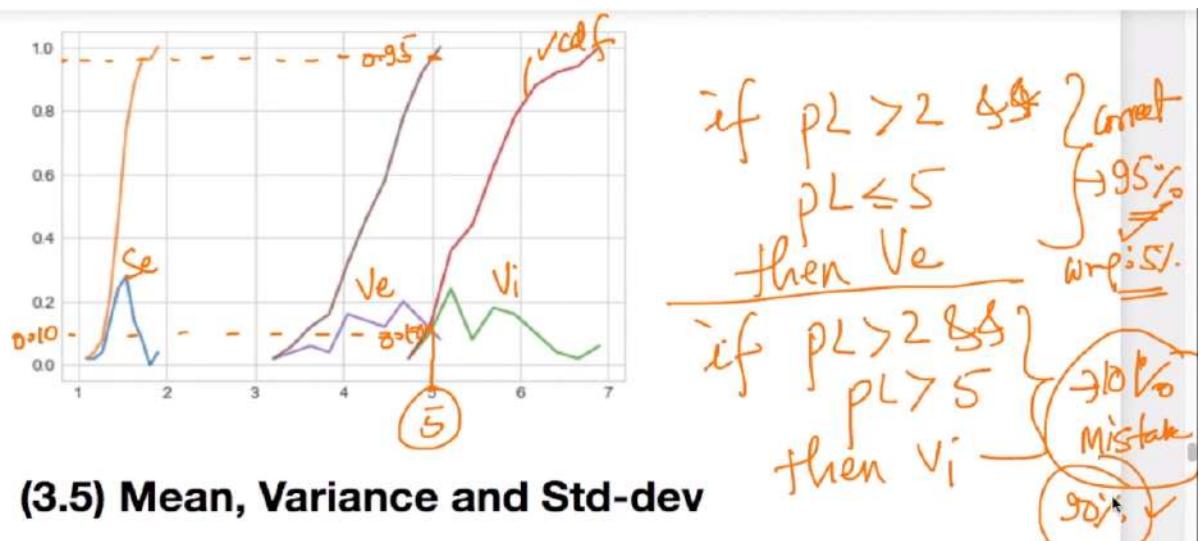
Here we are plotting the CDF and PDF for the 'petal_length' of the 'Iris-setosa' flower species.

The value of CDF for $\text{petal_length} = 1.3$ is 0.15. It means 15% of the 'Iris-setosa' flowers have a petal length of less than or equal to 1.3.

The value of CDF for $\text{petal_length} = 1.6$ is 0.82. It means 82% of the 'Iris-setosa' flowers have a petal length of less than or equal to 1.6.



Here the blue curve represents the PDF of the 'petal_length' values of 'Iris-setosa' flowers and the orange curve represents the CDF. The CDF value at a particular point is equal to the sum of all the PDF values present before that point.



(3.5) Mean, Variance and Std-dev

Here we make conclusions about the class labels as

if $PL \leq 2$:

then 'Setosa'

if $PL > 2 \text{ && } PL \leq 5$:

then 'Versicolor'

if $PL > 5$:

then 'Virginica'

We can get this information for decision making only with PDF, but not with the CDF.

20.8 Mean, Variance and Standard Deviation

Mean

- Mean is the total sum of all the values divided by the total number of values.
- Mean is one of the central measures of central tendency.

Let us assume the values we have are $x_1, x_2, x_3, \dots, x_n$, then the mean is defined as

$$\text{Mean}(\mu) = (x_1 + x_2 + x_3 + \dots + x_n)/n$$

Where 'n' → number of points/values

Standard Deviation

- Standard Deviation is the measure of the amount of variation of the values from the mean. Standard Deviation measures the spread of the data.
- Lower Standard Deviation indicates, the values are present closed to the mean, whereas higher Standard Deviation indicates, the values are present far away from the mean.

$$\text{Standard Deviation}(\sigma) = \sqrt{((x_1 - \mu)^2 + (x_2 - \mu)^2 + (x_3 - \mu)^2 + \dots + (x_n - \mu)^2)/n}$$

Where 'n' → number of points/values, 'μ' → Mean

Variance

- Variance is the squared deviation of the values from the mean.
- The more the variance is, the more the deviation would be. It is always recommended to have the variance as low as possible.
- Variance is the square of the Standard Deviation.

$$\text{Variance}(\sigma^2) = ((x_1 - \mu)^2 + (x_2 - \mu)^2 + (x_3 - \mu)^2 + \dots + (x_n - \mu)^2)/n$$

Where 'n' → number of points/values, 'μ' → Mean

Outliers

- Outliers are not just the greatest and the least values, but the values that are very different from the pattern established by the rest of the data.
- The statistics like mean, standard deviation and variance are easily affected by the presence of Outliers.
- In such cases, we have to remove the outliers from the data and then perform compute the statistics like mean and standard deviation.
- In case, if you do not want to remove the outliers, then instead of the mean and the standard deviation, you have to go for the statistics like median and median absolute deviation, which aren't easily affected by the presence of the outliers.

Below is the code snippet that was discussed in the video starting from the timestamp 1:30.

```
: #Mean, Variance, Std-deviation,
print("Means:")
print(np.mean(iris_setosa["petal_length"]))
#Mean with an outlier.
print(np.mean(np.append(iris_setosa["petal_length"],50)));
print(np.mean(iris_virginica["petal_length"]))
print(np.mean(iris_versicolor["petal_length"]))

print("\nStd-dev:");
print(np.std(iris_setosa["petal_length"]))
print(np.std(iris_virginica["petal_length"]))
print(np.std(iris_versicolor["petal_length"]))
```

Means:

1.464
2.41568627451
5.552
4.26

Std-dev:

0.171767284429
0.546347874527
0.465188133985

In this code snippet, we are computing the mean for the '**petal_length**' column values of the '**Iris-setosa**', '**Iris-versicolor**' and '**Iris-virginica**' species separately for each. In the 3rd statement, we are adding the value '**50**' which is an outlier to the '**petal_length**' column values and then when we compute the mean, we could see the mean got changed.

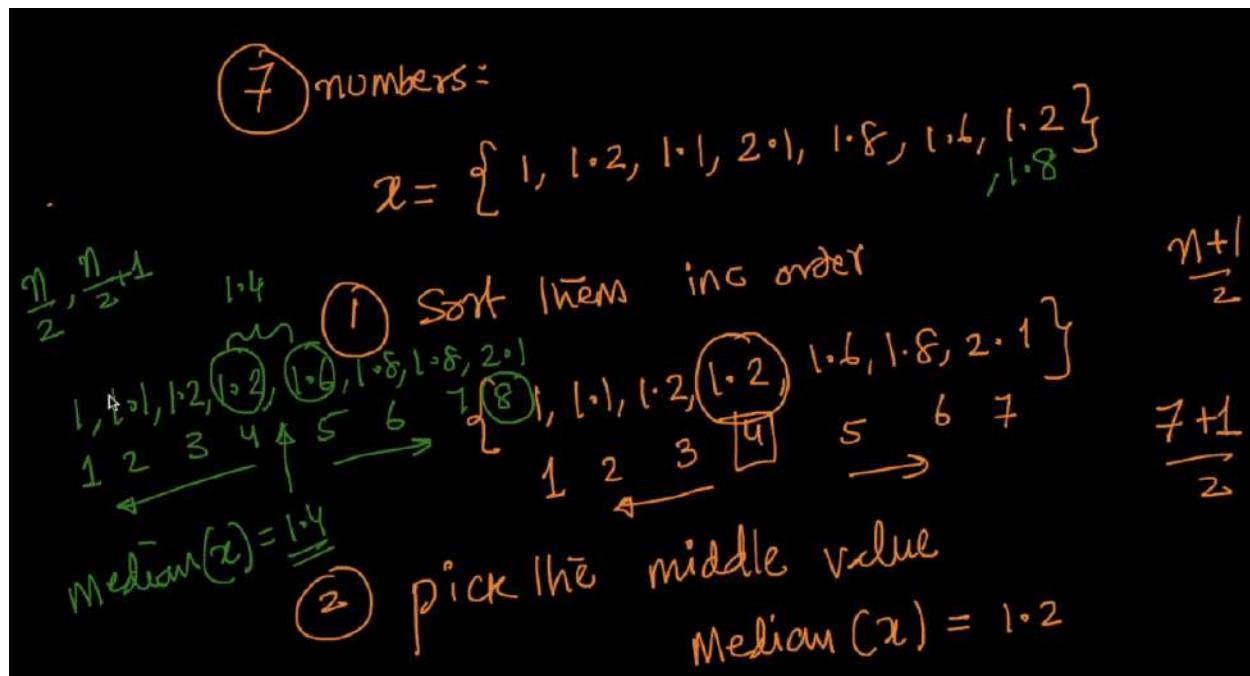
The metrics like Mean, Variance and Standard Deviation easily get affected with the presence of outliers in the data. Hence in order to handle this issue is

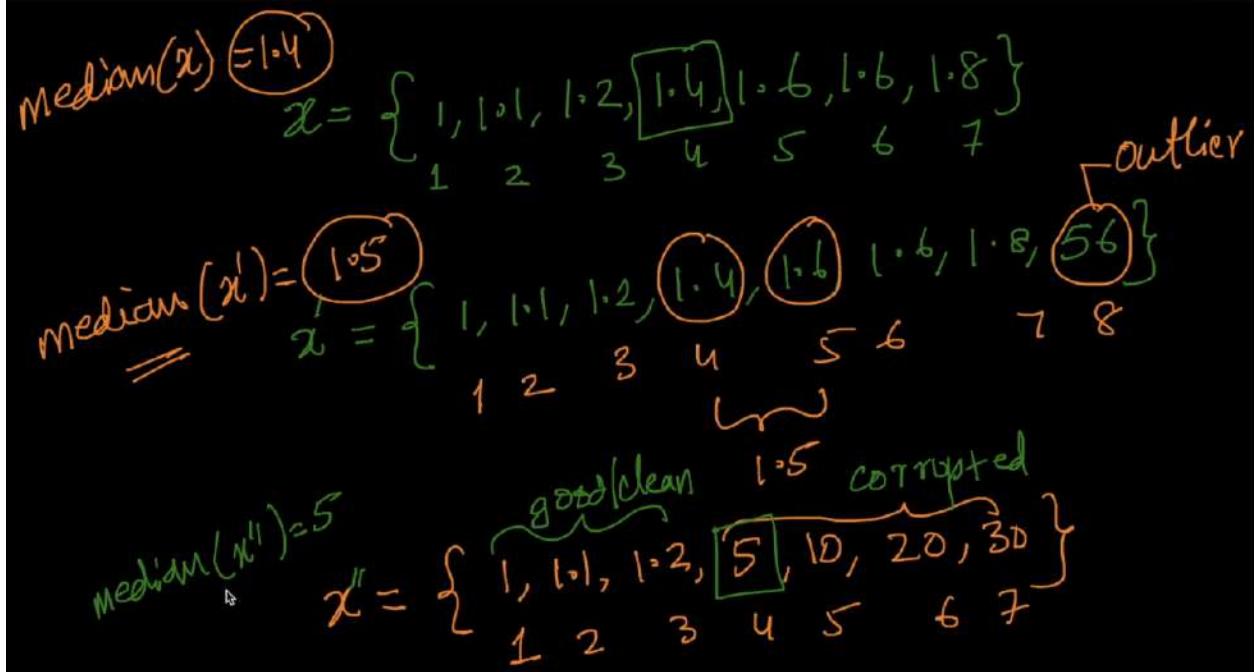
- Remove the outliers from the data and then compute the statistics Mean and Standard Deviation.
- If you do not want to remove the outliers, then better go for the statistics Median and Median Absolute Deviation.

20.9 Median

- Median is the value that lies in the middle among all the values/data points when the dataset is sorted.
- If the dataset has an odd number of points/values, then the middle value is the median.
- If the dataset has an even number of points/values, then the median is obtained by adding the two numbers in the middle and dividing the sum by 2.
- The median doesn't get affected even in the presence of outliers. Only if 50% or more values are outliers in the given dataset, then only the median gets affected.

Below are the examples that were discussed starting from the timestamp 2:45





Below is the code snippet on how to compute the median in Python and it was shown at the timestamp 9:50.

```
print("\nMedians:")
print(np.median(iris_setosa["petal_length"]))
#Median with an outlier
print(np.median(np.append(iris_setosa["petal_length"], 50)));
print(np.median(iris_virginica["petal_length"]))
print(np.median(iris_versicolor["petal_length"]))
```

```
Medians:
1.5
1.5
5.55
4.35
```

In the above code snippet, we are computing the median on the 'petal_length' column values for each species of Iris flowers.

20.10 Percentiles and Quantiles

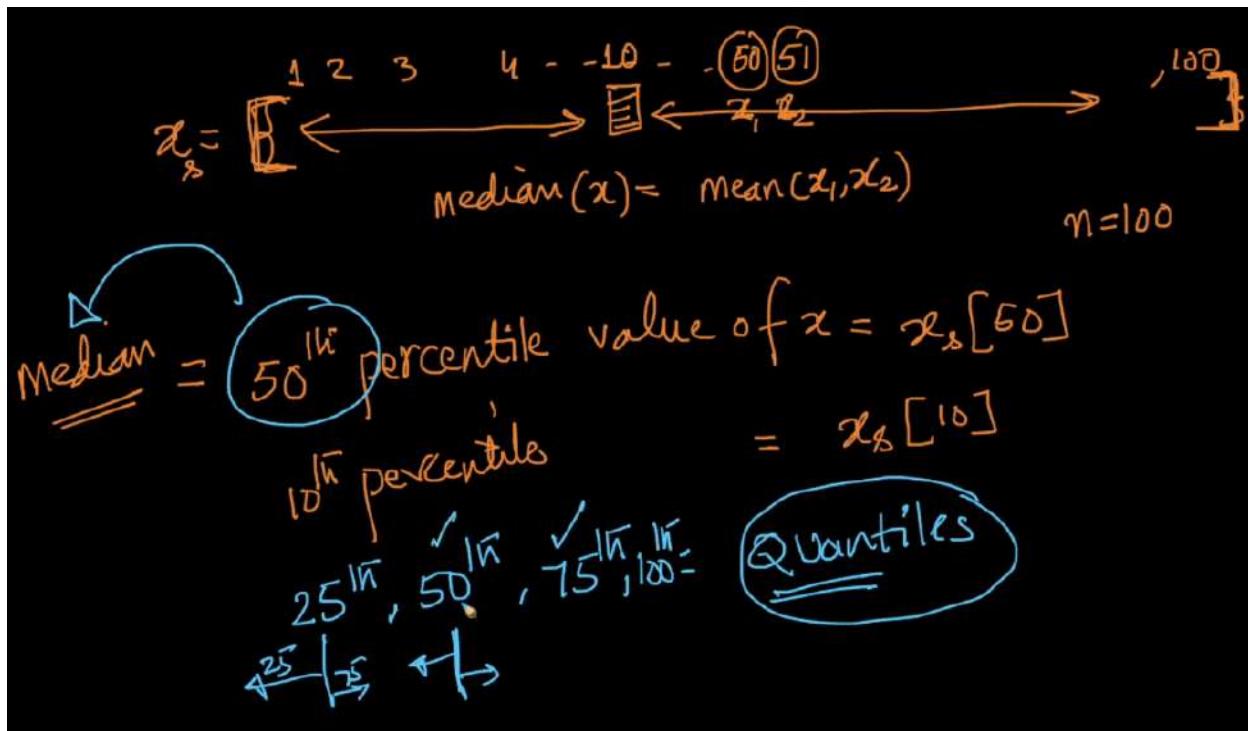
Percentile

- Percentile is a score below which a certain specified percentage of points/values fall and the remaining points/values fall above this score.
- For example, the k^{th} percentile indicates that about ' k '% of the points in the dataset fall below the k^{th} percentile and the remaining ' $(100-k)$ '% of the points in the dataset fall above the k^{th} percentile.

Quantiles

The 25th, 50th, 75th and the 100th percentiles are called the Quantiles.

The below example was discussed at the timestamp 0:25 where the concept of percentiles and quantiles was explained.



Below is the code snippet discussed at the timestamp 4:25 which explains how to compute the percentiles and quantiles in Python.

```
print("\nQuantiles:")
print(np.percentile(iris_setosa["petal_length"],np.arange(0, 100, 25)))
print(np.percentile(iris_virginica["petal_length"],np.arange(0, 100, 25)))
print(np.percentile(iris_versicolor["petal_length"], np.arange(0, 100, 25)))

print("\n90th Percentiles:")
print(np.percentile(iris_setosa["petal_length"],90))
print(np.percentile(iris_virginica["petal_length"],90))
print(np.percentile(iris_versicolor["petal_length"], 90))
```

```
Quantiles:
[ 1.      1.4     1.5     1.575]
[ 4.5     5.1     5.55    5.875]
[ 3.      4.      4.35    4.6 ]
```

```
90th Percentiles:
1.7
6.31
4.8
```

In this example, we are computing the quantiles and the 90th percentile of the 'petal_length' column for each of the IRIS flower classes separately.

20.11 Interquartile Range (IQR) and Median Absolute Deviation (MAD)

Median Absolute Deviation (MAD)

- Median Absolute Deviation (MA)
- D) is defined as the median of the deviations of the points from their median.
- MAD is also one of the measures of variability of quantitative data.
- As the mean and the standard deviation gets easily affected by the presence of outliers, we use median and MAD in place of mean and standard deviation.
- Median and the MAD are more robust to the outliers when compared to the mean and the standard deviation.

Let 'M' be the median of all the values, then

$$\text{MAD} = \text{median}(|(X_i - M)|)$$

- Median is an alternative to the Mean and MAD is an alternative to the standard deviation.

Below is the code snippet that shows how to compute the MAD in Python and it was discussed at the timestamp 3:10.

```
from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(iris_setosa["petal_length"]))
print(robust.mad(iris_virginica["petal_length"]))
print(robust.mad(iris_versicolor["petal_length"]))
```

```
Median Absolute Deviation
0.148260221851
0.667170998328
0.518910776477
```

We do not have direct functions to compute the MAD metric either in Numpy or in Scipy. Hence we are importing the 'robust' and are using the mad() function in it, to compute the MAD metric.

Interquartile Range(IQR)

- IQR is defined as the difference between the 75th and the 25th percentiles. That is, the difference between the 3rd and the 1st quartiles.

$$\text{IQR} = Q_3 - Q_1$$

- IQR is a rule of thumb used to identify the outliers present in the dataset.

According to the rule of thumb, if we have any points lying outside the interval $[Q_1 - (1.5 \times IQR), Q_3 + (1.5 \times IQR)]$, then those points are termed out as the outliers.

Also there are many other techniques like Local Outlier Factor(LOF), RANSAC, etc which are used for detecting the outliers and these techniques are discussed along with the Machine Learning topics.

Note: LOF, RANSAC are used when the data points are in vector form, whereas the IQR is used if the data points are scalars.

20.12 Boxplot with Whiskers

- So far we have learnt about percentiles, quantiles and the Interquartile range. Now when we look at the Histograms/PDFs while performing the univariate analysis, we could see the frequency distribution of the values and the spread of the values, but we could not obtain the 25th, 50th, 75th percentiles and the Interquartile range from such plots.
- The CDFs can give these values, but not directly. We manually have to check at what value of a numerical feature, is the CDF value equal to 0.25, 0.5 and 0.75. Though the Histograms/PDFs could explain the density of the points, they could not give any information about the quantiles. So in order to obtain the quantiles and the Interquartile range, we do have some plots and boxplot is one among such plots.
- The kind of analysis we can make with a boxplot is checking how many points are having the values below the 25th percentile, how many points have the values above the 75th percentile and how many points have the values in the IQR, etc.
- Regarding the whiskers, there is no standard way of drawing it. Typically, we take $(Q_3 + (1.5 * IQR))$ as the value for the upper whisker and $(Q_1 - (1.5 * IQR))$ as the value for the lower whisker.
- A Boxplot displays the summary of a large amount of data in five numbers. These numbers include the Median, the Upper Quantile(Q_3), the Lower Quantile(Q_1), the minimum and the maximum data values.

Advantages of Boxplot

1) Handles Large Data Easily

Organizing the data in a boxplot by using five concepts is an efficient way of dealing with large data, which is unmanageable for the other plots like a line plot or a stem plot or a leaf plot.

2) Exact values are not retained

The boxplot doesn't keep the exact values and the details of the distribution results. A box plot shows only a simple summary of the distribution of results, so that we can quickly view it and compare it with the other data.

We have to use a box plot in combination with other statistical graph methods like histogram for more thorough and detailed analysis of the data.

3) Clear Summary

A box plot is a highly visually effective way of viewing a clear summary of one or more sets of data. It is particularly useful for quickly summarizing and comparing different sets of results from different experiments. A box plot allows a

graphical display of the distribution of results and provides indications of symmetry within the data.

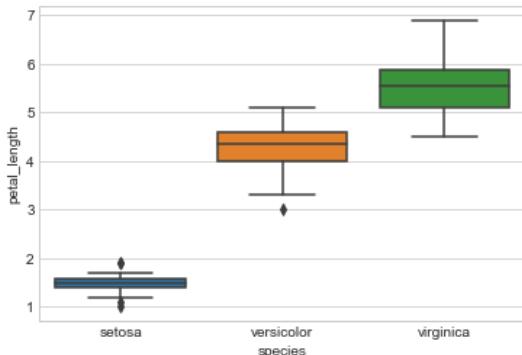
4) Displays Outliers

A box plot is one of very few statistical graph methods that show outliers. There might be one outlier or multiple outliers within a set of data, which occurs both below and above the minimum and maximum data values.

By extending the minimum and the maximum data values by 1.5 times the Interquartile range, the box plot delivers outliers. Any values that fall outside of the extended minimum and the maximum values, are known as outliers and these outliers can be easily determined using a box plot.

Below is an example of the code snippet and the graph of a boxplot that was discussed starting from the timestamp 1:20.

```
#Box-plot with whiskers: another method of visualizing the 1-D scatter plot more intuitively.  
# The Concept of median, percentile, quantile.  
# How to draw the box in the box-plot?  
# How to draw whiskers: [no standard way] Could use min and max or use other complex statistical techniques.  
# IQR Like idea.  
  
#NOTE: IN the plot below, a technique call inter-quartile range is used in plotting the whiskers.  
#Whiskers in the plot below donot correposnd to the min and max values.  
  
#Box-plot can be visualized as a PDF on the side-ways.  
  
sns.boxplot(x='species',y='petal_length', data=iris)  
plt.show()
```



In this video, we are plotting a boxplot graph for the 'petal_length' column values for different classes of the 'Iris' flowers.

```

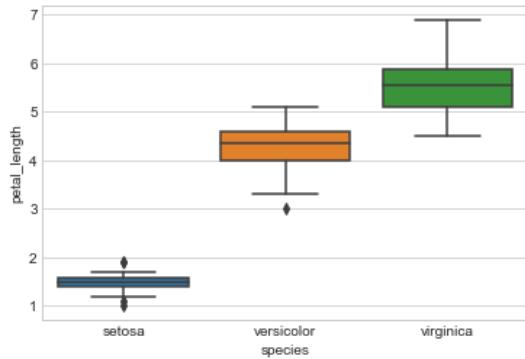
#Box-plot with whiskers: another method of visualizing the 1-D scatter plot more intuitively.
# The Concept of median, percentile, quantile.
# How to draw the box in the box-plot?
# How to draw whiskers: [no standard way] Could use min and max or use other complex statistical techniques.
# IQR like idea.

#NOTE: IN the plot below, a technique call inter-quartile range is used in plotting the whiskers.
#Whiskers in the plot below donot correposnd to the min and max values.

#Box-plot can be visualized as a PDF on the side-ways.

sns.boxplot(x='species',y='petal_length', data=iris)
plt.show()

```



Here the ‘x’ parameter in **sns.boxplot()** denotes the class column with class values. A separate boxplot gets created for each of the classes.

The ‘y’ parameter takes the numerical column, which we use in plotting the boxplot graph. The ‘data’ parameter takes the dataframe as an input.

20.13 Violin Plot

Violin plots are the plots that have the benefit of both box plot and Probability Density Function (PDF). It is a combination of both these plots.

Advantages of Violin Plot

1) Violin plot is like a box plot but better

Box plots show Median, ranges, variations effectively. They allow comparing groups of different sizes. But the box plots are misleading. They aren't affected by the data's distribution. Whenever the data changes, the statistical summaries (like the median and the range) might change, but the box plots remain the same.

This is when we have to use a violin plot. A violin plot carries all the information that a box plot would carry and it literally has a box plot inside the violin, but doesn't fall into the distribution trap.

2) Violin graph is like a density plot, but better

The violin shape of the violin plot comes from the data's density plot. We can just turn that density plot sideway and put it on both sides of the box plot, mirroring each other.

Reading the violin shape is exactly how we read a density plot. The thicker part indicates the values in that section of the violin have higher frequency and the thinner part implies lower frequencies.

3) Violin graph is visually intuitive and attractive

To compare different sets, their violin plots are placed side by side. They don't sit on top of one another. Violin plots are easy to read. The dot in the middle represents the median. The box represents the interquartile range. The shape of the violin displays the frequencies of the values.

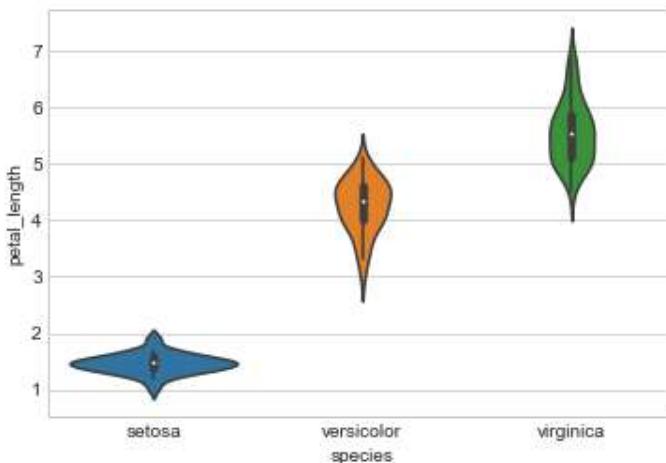
Different violins are the different sets. We don't even have to fill the chart with colors and patterns to distinguish sets, which actually makes it less distracting and easier to read.

4) Violin graph is non-parametric

Unlike the bar graphs with the mean and the error bars, violin plots contain all data points. This make them an excellent tool to visualize samples of small sizes. Violin plots are perfectly appropriate even if the data doesn't come from the normal distribution. They work well to distinguish both the qualitative and the quantitative data.

Below is an example of the violin plot and it's code snippet that were discussed starting from the timestamp 1:20.

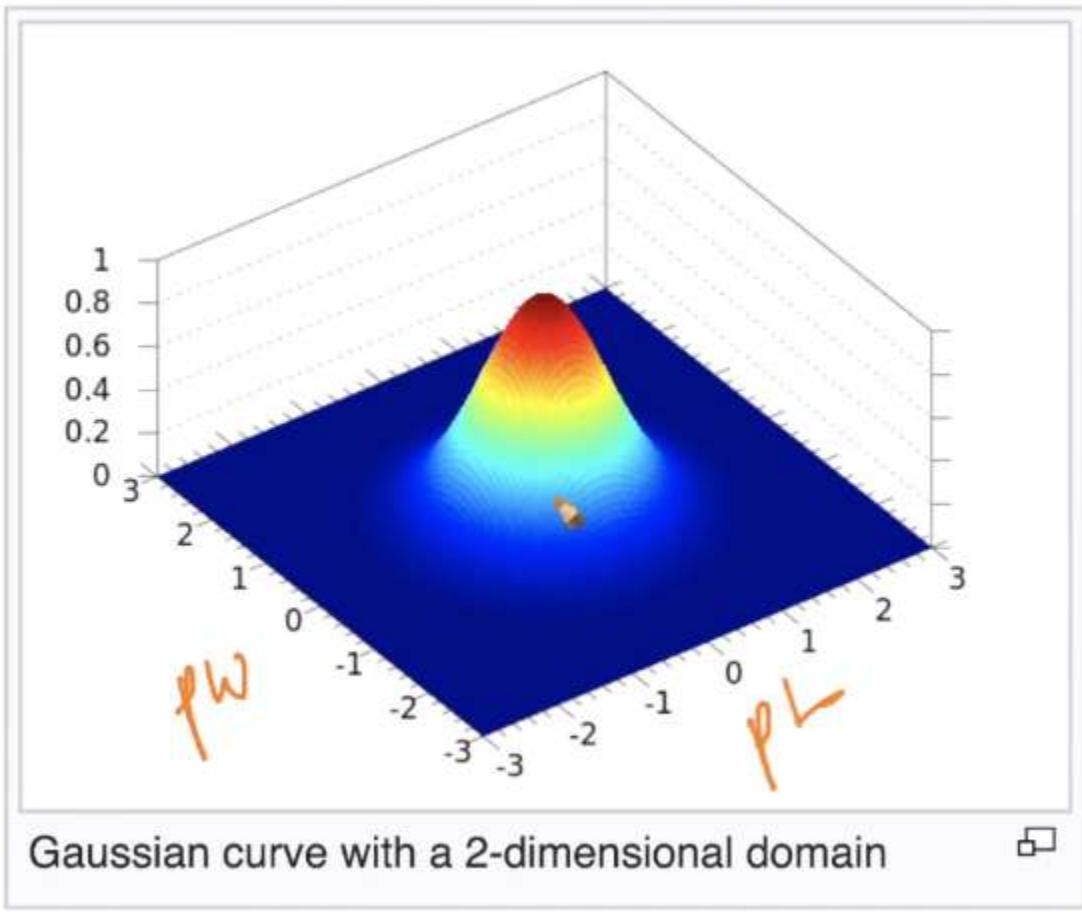
```
: # A violin plot combines the benefits of the previous two plots  
# and simplifies them  
  
# Denser regions of the data are fatter, and sparser ones thinner  
# in a violin plot  
  
sns.violinplot(x="species", y="petal_length", data=iris, size=8)  
plt.show()
```

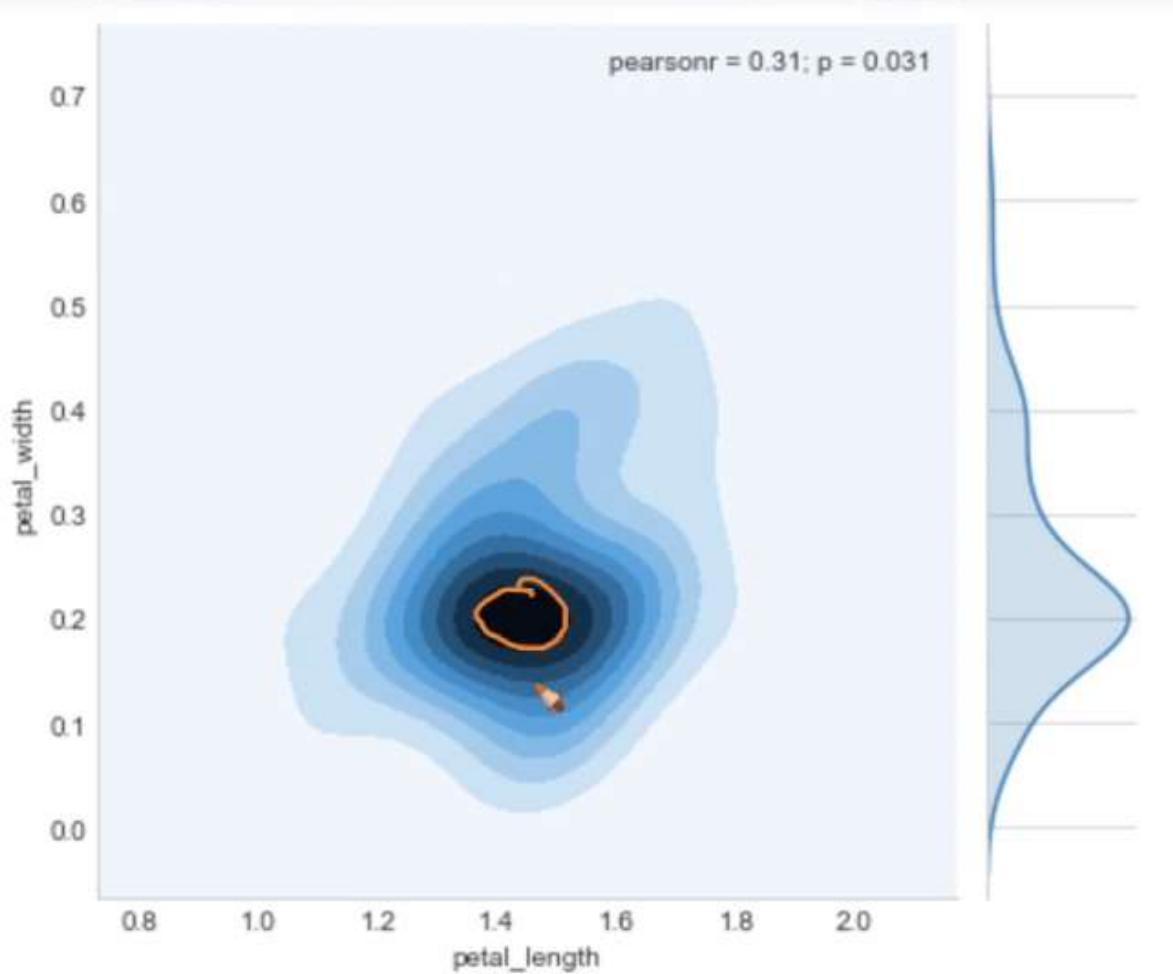


20.15 Multivariate Probability Density, Contour Plot

So far we have learnt about 2D density plots with the density distribution of only one variable. Now the Contour plot here deals with plotting the density functions using 2 variables at a time. This plot is now a 3D plot, with values of one variable on the 'X' axis and the other variable values on the 'Y' axis and the PDF values on the 'Z' axis. Below is an example of code snippet and the graph of a contour plot.

```
#2D Density plot, contours-plot  
sns.jointplot(x="petal_length", y="petal_width", data=iris_setosa, kind="kde");  
plt.show();
```





In this plot, we can see the darker region has more density of points when compared to the lighter region. Also all the points present in a particular region with the same color are at the same height. The PDFs present at the top and the right side of this contour plots are the PDFs of the individual features ‘petal_length’ and ‘petal_width’.

21.2 Introduction to Probability and Statistics

Timestamp 3.41



1. As shown above if we consider a query datapoint X_q , using probability we will be able to describe there is 0.8 probability for it to be versicolor and 0.2 probability for it to be virginica and 0 probability for setosa . Lets understand the probability and statistics with respect to Machine Learning

dice : six-sides $\rightarrow \{1, 2, 3, 4, 5, 6\}$
 (roll a fair dice) \uparrow equally likely
 $\text{Q.V. } X = \{1, 2, 3, 4, 5, 6\}$

tossing of a coin
 $\text{Q.V. } Y = \{H, T\}$

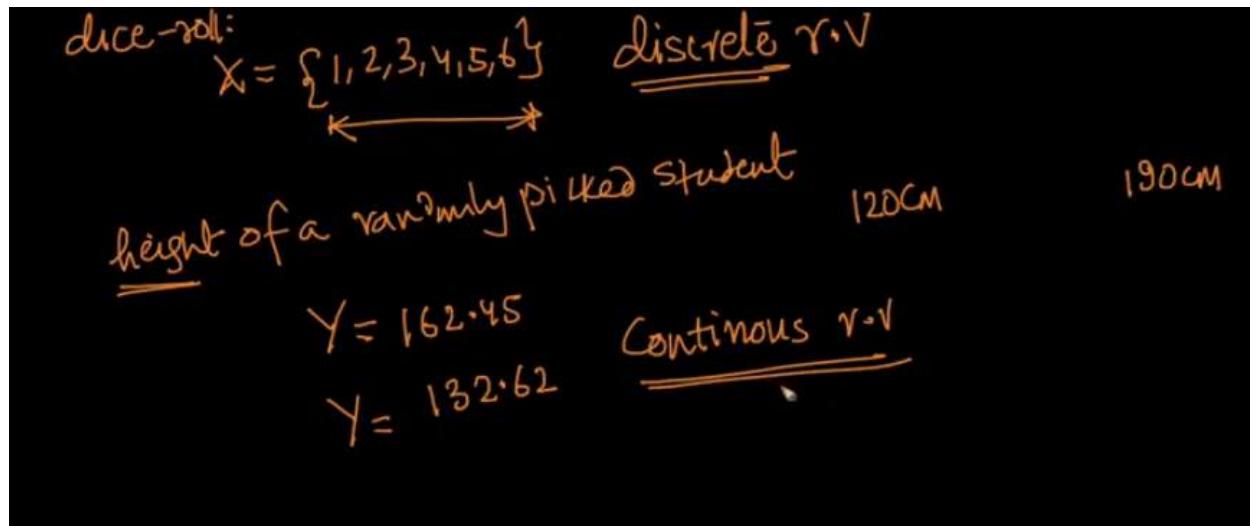
1. In an experiment of rolling a fair dice, random variable X can take any of the values from set $\{1, 2, 3, 4, 5, 6\}$ and is equally likely.
2. Similarly tossing a fair coin random variable y can take any of values from set $\{H, T\}$ and is equally likely.

Timestamp 11.12

$$\begin{aligned}
 & \text{dice-roll: } X = \{1, 2, 3, 4, 5, 6\} \\
 & P(X=1) = \frac{1}{6} = P(X=2) \\
 & P(X \text{ is even}) = \frac{1}{2} = P(X=2) + P(X=4) + P(X=6) \\
 & P(X \text{ is odd}) = \frac{1}{2} \\
 & P(X=x_i) \rightarrow \boxed{P(x_i)}
 \end{aligned}$$

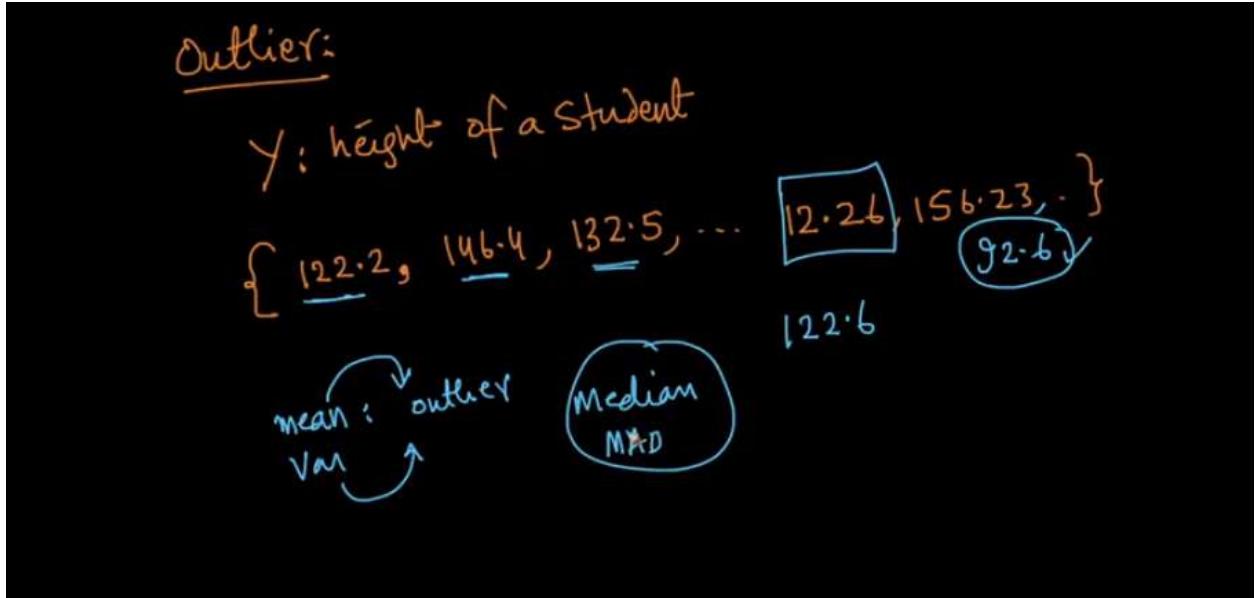
1. Probability of random variable x taking a value 1 when a dice is rolled is $\frac{1}{6}$ and is same as probability of x taking value 2
2. When a fair dice is rolled the probability of random variable x taking an even number is $\frac{1}{2}$ as shown above which is the same as x taking an odd number.

Timestamp 12.11



1. There are two types of random variables, consider an experiment where a dice is rolled a random variable x can take any value in set $\{1,2,3,4,5,6\}$. Another experiment where the height of a randomly picked student lies between 120-160, random variable y can take a real value in the range.
2. A random variable x which can take one value in a finite set of outcomes(x can take a value from a discrete set) is a discrete random variable.
3. A random variable y which can take real values is called a continuous random variable

Timestamp 14.4



1. Let's say we have collected the observations of random variable y (heights of students) as shown above.
2. If there is an observation which is too small or too large out of all other observations such as observation is called an outlier. Outlier can happen because of any reason it may be a human error, observation error or it could be genuine outlier, but outliers can corrupt our analysis as mean and variance are very much impacted by outliers.

21.3 Population and Sample

Timestamp 4.21

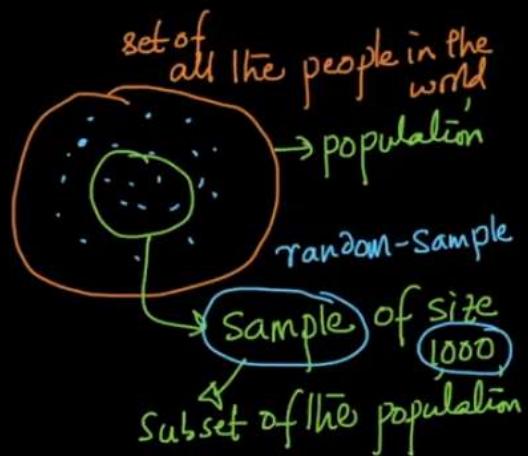
Population & Sample:

→ estimate height of a human
 mean of a pop $\mu = \frac{1}{7B} + \sum_{i=1}^{7B} h_i$

Mean of sample \bar{x}

$$\bar{h} = \frac{1}{1000} + \sum_{i=1}^{1000} h_i$$

↑ heights in my sample



As sample size increases

$$\bar{x} = \mu$$

↑ sample-mean ← pop-mean

1. Let's assume set of all people in the world and it's the population, now we want to estimate the average /mean height of the humans as shown above. We cannot find the heights of all the people in the population inorder to find the mean height of the population (μ) so we are going to estimate the mean height using a sample(say a sample of size 1000).
2. Sample is a subset of population(if sample is collected randomly such a sample is called random sample), now we can find the mean height of the sample as shown.
3. As the size of sample increases, sample will be equal to population mean.
4. Population is set of all events or observations whereas a sample is a subset of population which we use to estimate some statistic or some property of population .Here height is the property of population we have estimated.

21.4 Probability from a set theoretic view

Timestamp 9,45

Sample Space (S): Set of all the possible outcomes of an exp

Expt flipping 2 coins (distinct)

Outcomes: $\{(H,H), (H,T), (T,H), (T,T)\} = S$

Expt: conduct a 7-way horse race

Outcomes: ordering of 7 horses

$2, 1, 3, 6, 5, 4, 7$

$7!$

$S = \{7! \text{ possible orderings}\}$

horse-race $\{1, 2, 3, \dots, 7\}$

Expt: 2 dice (distinct) $1, 2, 3, \dots, 6$

Outcomes: $\{(1,1), (1,2), (1,3), (1,4), \dots\} = S = \{36 \text{ possible outcomes}\}$

$6 + 6 = 36$

Expt: light bulb, measuring the $\#$ hrs the light bulb works.

Outcomes: $S = \{x : 0 \leq x < \infty\}$

$x \in \mathbb{R}$

0.1

0.2

0.11

1. Let's study probability from set theoretic view
2. Let's say an experiment of flipping 2 coins the possible outcomes are as shown above. Set of all possible outcomes if an experiment is called the sample space(S).

3. Imagine another experiment where we conduct a seven horse race ,the set of all possible outcomes are the ordering of horses after the race is sample space i.e) $7!$ possible orderings is the sample space
4. In experiment if throwing two distinct dice the set of all possible outcomes are 36 (6×6)which is sample space.
5. Consider another experiment where we have a light bulb and we will be measuring the number of hours the light bulb works .In this case our possible outcomes lie in range $0 \leq X < \infty$ which is sample space.

The possible outcomes need not be discrete finite sets; it can also be an infinite set of outcomes. Sample space is the space of all the possible outcomes that we care about as experiment is concerned.

Timestamp 16.57

Event (E) Any Subset of S is an event

(e.g) $E = \{x : 0 \leq x \leq 5\} \Rightarrow$ light bulb works for 0 to 5 hrs

$E \subseteq S$ $\left\{ \begin{array}{c} l_1 \\ l_2 \\ l_3 \\ \vdots \\ l_m \end{array} \right\} \rightarrow \underline{\leq 5 \text{ hrs}}$

Expt: 2 dice (distinct) $1, 2, 3, \dots \rightarrow$

Outcomes: $\{(1,2), (1,3), (1,4), \dots\} = S \quad \{36 \text{ possible outcomes}\}$

$6 + 6 = 36 \quad S \ni E = \{(3,3), (1,5), (5,1), (4,2), (2,4)\}$

1. An event is any subset of S (Sample space)including null set.
2. Consider the experiment of the light bulb discussed above and we can have an event E such that the light bulb works for 5 hours as shown . $E: 0 \leq X \leq 5$
3. In the experiment of throwing two dice lets define an event E such that the sum of the dice is 6 $E:\{(3,3),(2,4),(4,2),(1,5),(5,1)\}$
4. Above are all possible ways in which the event(getting sum 6 when two dice are rolled)can occur.

Similarly we can define events on experiments where $E \subseteq S$.

21.5 Axioms of Probability, Properties and Examples-

Timestamp 4.37

✓ Set - Theory ✓

expt S : sample space \cup
 E : event : set of outcomes

$E \subseteq S$ (e.g) throwing 2 coins (distinct)

$E = \{(T,T), (T, H)\}$ $S = \{(H,H), (H,T), (T,H), (T,T)\}$

$E \cap F = \{(H,H)\}$ $E = \text{all outcomes where first coin is } H$

$E \cup F = \{(H,H), (T,T), (H,T)\}$ $F = \text{all outcomes with both coins of same value}$

$S \setminus \{(H,H), (T,T)\} = F$



APPLIED

Lets study probability in set theoretic perspective

1. If we consider an experiment we define a sample space (S) which is a set of all outcomes. An event E which is a set of outcomes (subset of S).
2. Let's say we are flipping two coins and we have our sample space as $s=\{HH, HF, FH, HH\}$
3. Consider an event E of getting outcomes where first coin is H ($E=\{HH, HT\}$)
4. Also consider another event F of getting outcomes where both coins are of same value
5. Now we can think of these events E and F as sets and can perform all operations which we can perform on sets. i.e) set union, intersection, complement etc as shown above.

Timestamp 12.48

Frequentist vs Bayesian ↗ Bayes Theorem

① conduct the expt n times

② $n(E) = \text{number of times the outcome of the expt } E$

(e.g.) fair coin $\rightarrow H \quad T$
 $S = \{H, T\}$
 $E = \{H\}$

$P(E) = \lim_{n \rightarrow \infty} \frac{n(E)}{n}$

expt: Toss a fair coin once
 $E = \{H\}$

frequency \rightarrow
 $P(H) = \lim_{n \rightarrow \infty} \frac{n(H)}{n}$

6/10 times $\rightarrow 6H$
10 times $\rightarrow 4T$
1000 $\rightarrow 500H$
10000 $\rightarrow 494T$
(0.5)

APPLIED COURSE

Let's understand the Frequentist and Bayesian approach of probability

From a frequentist approach probability of an event $p(E)$ can be defined as

1. let's say we are conducting an experiment n times (assume the experiment of tossing two fair coins)
2. $n(E)$ be the number of times the outcome of the experiment belongs to event E

Then $p(E) = \lim_{n \rightarrow \infty} n(E)/n$

$n(E)$ is the frequency of occurrence of the event and n is total number of times we are conducting the experiment as n tends to infinity (we conduct the experiment infinite times)

Set Theory Axiomatics (Self-evident)

Geometry: parallel lines don't meet

Physics
[3 Newton's laws]

↳ Mechanics

S

① $0 \leq p(E) \leq 1$

② $p(S) = 1$

③ If E_1, E_2, E_3, \dots are mutually exclusive
then $E_i \cap E_j = \emptyset$

$p(E_1 \cup E_2 \cup E_3 \cup E_4 \dots) = \sum_i p(E_i)$

$\sqrt{\{ |E_1 \cup E_2 \cup E_3 \cup \dots|} = |E_1| + |E_2| + |E_3| + \dots$

APPLIED
SCIENCE

Axiomatic(self evident) approach of probability says

1. The probability of any event lies between 0 and 1 i.e) $0 \leq p(E) \leq 1$
2. The probability of sample space is $p(S)=1$
3. If there are events E_1, E_2, E_3, \dots etc which are mutually exclusive ($E_1 \cap E_2 = \emptyset$) then $p(E_1 \cup E_2 \cup E_3 \cup E_4 \dots) = \sum_{i=1}^n p(E_i)$

Using the above axioms and properties in set theory we can actually prove all theorems in probability

Timestamp 12.42

Dice

$$S = \{1, 2, 3, 4, 5, 6\}$$

mutually exclusive

$$\begin{aligned} E_1 &= \{2\} \neq \frac{1}{6} \\ E_2 &= \{4\} \neq \frac{1}{6} \\ E_3 &= \{6\} \neq \frac{1}{6} \end{aligned}$$

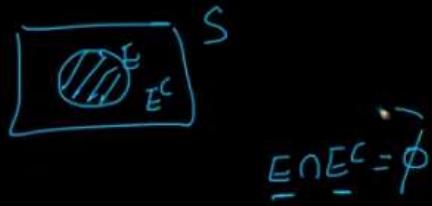
$$\begin{aligned} p(E_1 \cup E_2 \cup E_3) &= p(\{2, 4, 6\}) = p(E_1) + p(E_2) + p(E_3) \\ &= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2} \end{aligned}$$

Above is an example where we have three mutually exclusive events and their union is equal to the sum of the events .

Timestamp 23.35

Results:

$$\textcircled{1} \quad P(E^c) = 1 - P(E)$$



$$\textcircled{2} \quad \checkmark P(S) = 1$$

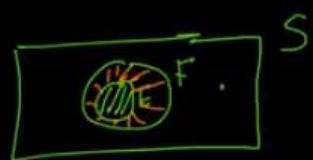
$$\textcircled{1} \quad P(E \cup E^c) = P(S) = 1$$

$$P(E) + P(E^c) = 1$$

$$P(E^c) = 1 - P(E)$$

$$E \cap E^c = \emptyset$$

$$\textcircled{2} \quad \text{if } E \subseteq F \quad P(E) \leq P(F)$$



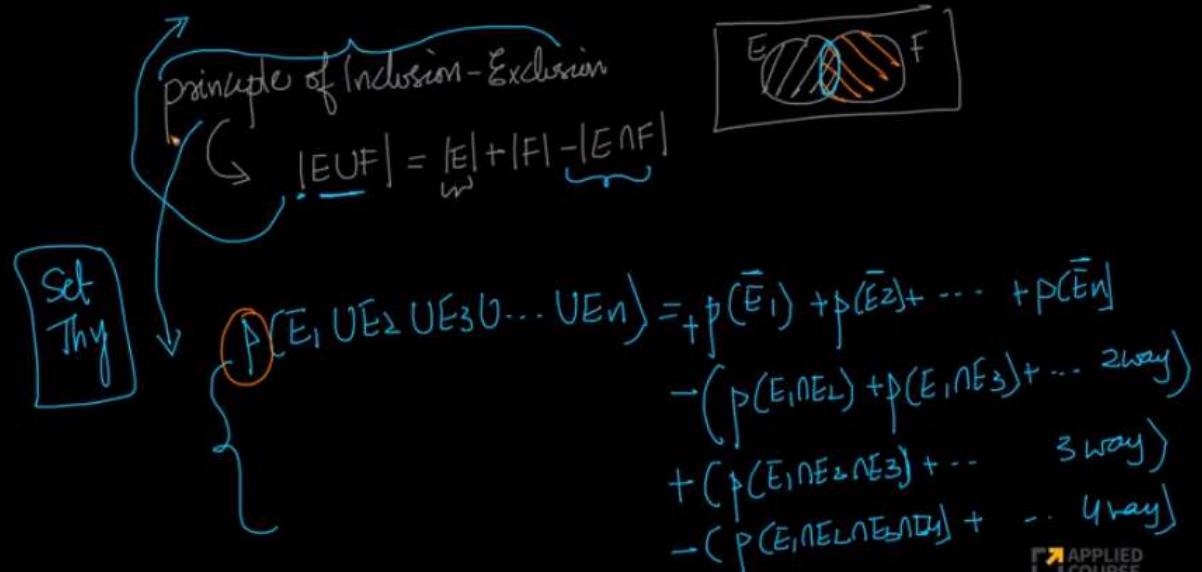
$$F = E \cup (F \cap E^c)$$

$$\checkmark P(F) = P(E \cup (F \cap E^c))$$

$$P(F) = P(E) + P(F \cap E^c)$$

$$\boxed{P(F) \geq P(E)}$$

$$③ \checkmark P(E \cup F) = P(E) + P(F) - P(E \cap F)$$



- Above are few examples of proof that we can solve all theorems in probability using these axioms and properties in set theory

Timestamp 34.43

Sample Space with equally likely outcomes

(dice) $S = \{1, 2, 3, 4, 5, 6\}$

$P(E) = \frac{|E|}{|S|}$

$P(\{1\}) = P(\{2\}) = P(\{3\}) = P(\{4\}) = \dots = P(\{6\}) \checkmark \text{ given}$

$\checkmark \left\{ P(S) = 1 \rightarrow \text{2nd axiom } P(\{1, 2, 3, \dots, 6\}) = \sum_{i=1}^6 P(\{i\}) = P(S) = 6 \right.$

$P(\{i\}) = \frac{1}{6}$

- In an experiment of rolling a dice, we have a sample space with equally likely outcomes as shown above.

$$S = \{1, 2, 3, 4, 5, 6\}$$

$p(\{1\}) = p(\{2\}) = p(\{3\}) = p(\{4\}) = p(\{5\}) = p(\{6\}) = \frac{1}{6}$ and We know that probability of sample space is $p(S) = 1$

- Hence, $p(E) = |E|/|S|$ this is true for sample space with equally likely events

Timestamp

(e.g.)

$$\frac{6C_1 \cdot 5C_2}{11C_3} = \frac{4}{11}$$

$|S| = 11C_3$

$p(E) = \frac{|E|}{|S|}$

$\checkmark \left\{ |E| = 6C_1 \cdot 5C_2 \right\} \checkmark$

- In the above example we have to find probability that we have to choose 3 balls out of 11 balls such that we need to choose exactly 1 white and 2 black. Out of 11 balls we have 6 are white and 5 are black.
- As shown above using combinatorics we can find the probability is 4/11

(e.g) n balls

One special

expt: K balls drawn one at a time randomly

E : special ball is picked

$$\frac{|E|}{|S|} = P(E)$$

$$|S| = {}^n C_K$$

$$= \frac{{}^{n-1} C_{K-1}}{^n C_K}$$

$$|E| = {}^1 C_1 \cdot \frac{{}^{n-1} C_{K-1}}{K-1}$$

$$\left\{ \frac{K}{n} \right\} \checkmark$$

1. Above example we have n balls out of which one ball is special. Let's say in an experiment of drawing k balls out of n balls randomly.
2. Let say event E where special ball is picked
3. $P(E)$ can be calculated using combinatorics as shown above

Matching Problem:

{ 'N' men → ✓ Throw

pick a hat randomly



$P(\text{No one has picked their hat}) = ?$

WINT: Inclusion-Excl

Counting

- In a party where N men throw their hats into the bag and then each of them picks a hat randomly. We have to find probability that No one picks their own hat. It can be solved using principle of inclusion and exclusion as shown below

$E_i = i^{\text{th}}$ person has picked the correct hat

$$P(E_1 \cup E_2 \cup \dots \cup E_n) = \text{prob. of at least one person picking their hat}$$

$$1 - P(E_1 \cup E_2 \cup \dots \cup E_n) = \text{prob. of no person picking their hat}$$

$$\rightarrow \frac{1}{n} + \frac{1}{n} + \dots + \frac{1}{n} = \frac{n}{n} = 1/1!$$

$$\rightarrow P(E_1) + P(E_2) + \dots + P(E_n) \rightarrow \left(\frac{1}{n} \cdot \frac{1}{n-1}\right)^n C_2 = \frac{n \times n-1}{1!} \times \frac{1}{n} \cdot \frac{1}{n-1}$$

$$- [P(E_1 \cap E_2) + P(E_1 \cap E_3) + \dots] \rightarrow \frac{1}{2!}$$

$$+ [P(E_1 \cap E_2 \cap E_3) + \dots] \rightarrow \frac{1}{n} \cdot \frac{1}{n-1} \cdot \frac{1}{n-2} \cdot \frac{n \times n-1 \times n-2}{3!} = \frac{1}{3!}$$

P 3

$$P\left(\bigcup_{i=1}^n E_i\right) = \frac{1}{1!} - \frac{1}{2!} + \frac{1}{3!} - \frac{1}{4!} + \dots$$

$$1 - P\left(\bigcup_{i=1}^n E_i\right) = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!} + \dots + \frac{1}{\infty!}$$

if $n = \infty$

$$e^{-1} = 0.36788$$

21.6 Conditional Probability & Examples

Timestamp

The notes illustrate the concept of conditional probability using the example of throwing two dice. The sample space S consists of 36 outcomes: $\{(1,1), (1,2), \dots, (6,6)\}$. Event F is defined as the first die showing a 3, which includes outcomes $\{(3,1), (3,2), (3,3), (3,4), (3,5), (3,6)\}$. Event E is defined as the sum of the two dice being 7, which includes outcomes $\{(1,6), (2,5), (3,4), (4,3), (5,2), (6,1)\}$. The intersection of events E and F is $\{(3,4), (3,5), (3,6)\}$. The probability of event E given that event F has occurred is calculated as $P(E|F) = \frac{P(E \cap F)}{P(F)}$, where $P(F) = \frac{6}{36} = \frac{1}{6}$ and $P(E \cap F) = \frac{3}{36} = \frac{1}{12}$.

Let's try to understand conditional probability using an example

- Consider an experiment of throwing 2 dice, we have Sample space $S=\{(1,1),(1,2),\dots,(6,6)\}$
- Let's say
Event F where we get 3 on the first dice - $\{(3,1),(3,2),(3,4),(3,3),(3,5),(3,6)\}$
Event E where sum of two dice is 7 - $\{(1,6)(2,5),(3,4),(4,3),(5,2),(6,1)\}$

Now we can find probability of event E given that event F has already occurred using conditional probability

$$P(E|F) = P(E \cap f)/P(F) \text{ as shown above}$$

Conditioned on the fact that F has already occurred the probability of event E occurring will be given by $P(E \cap f)/P(F)$ and probability holds only when $P(F) \neq 0$.

Timestamp 13.30

(Q) Student taking one-hr exam

$\checkmark p(\text{student finishes the exam in under } \underline{x} \text{ hrs}) = \frac{x}{2} \rightarrow \textcircled{1}$

{ Given a student is working at 0.75 hrs what is the prob that the student uses the full 1hr $\rightarrow 0.8$

$\checkmark \begin{cases} F = \text{student uses the full one hr} \\ F^c = \text{student finishes exam in under 1hr} \end{cases}$

$P(F^c) = \frac{1}{2} \quad P(F) = 1 - \frac{1}{2} = \frac{1}{2} \rightarrow \textcircled{2}$

$L_x = \text{student finishes in } x \text{ hrs}$

$L_x^c = \text{student is still working at } x \text{ hrs}$

$\checkmark L_{0.75}^c$

$P(F | L_{0.75}^c) = \frac{P(F \cap L_{0.75}^c)}{P(L_{0.75}^c)} = \frac{P(F)}{1 - P(L_{0.75})} = \frac{\frac{1}{2}}{1 - \frac{1}{2}} = \frac{1}{2}$

$P(L_x) = \frac{x}{2} \quad 0 \leq x \leq 1$

$\boxed{0.8}$

Consider the above example where a student is talking one hour exam.

- Given probability that student finishes the exam in under hrs=x/2
- Given a student is working at 0.75 hrs We need to find the probability that the student uses full hour.

We can solve the problem as shown above using conditional probability.

21.7 Multiplication theorem

Timestamp

(e.g) Select \underline{n} balls
sequentially & randomly chosen ✓
without replacement
urn

urn → r red, b blue balls

Given that $n \leq r+b$
 K out of n balls chosen are blue,
what's the prob of the 1st ball picked is blue } cond. prob

- For understanding conditional probability let's consider the above example
- There are r red balls and b blue balls in an urn ,we have to select n balls sequentially and randomly chosen without replacement.
- Given that out of n balls chosen K balls are blue.We need to find the probability of the 1 st ball being picked is blue.

B = event that 1st picked ball is blue
 B_k = event that k out of n balls picked are blue

$$P(B|B_k) = \frac{P(B \cap B_k)}{P(B_k)}$$

$$\checkmark P(B_k) = \frac{\frac{b}{c_n} \cdot \frac{r}{n-k}}{\frac{b+r}{c_n}} = \frac{n(B_k)}{n(S)}$$

We define events as shown above and we need to find

$$P(B|BK) = P(B \cap BK)/P(BK)$$

We calculate the probability of event BK (k out of n balls picked are blue) as shown above.

- We are choosing n balls out of $b+r$ balls, this is our sample space.
- We select k balls out of b blue balls and $n-k$ balls out of r red balls

Timestamp 10.19

$\checkmark B$ = event that 1st picked ball is blue
 B_k = event that k out of n balls picked are blue

$$P(B|B_k) = \frac{P(B \cap B_k)}{P(B_k)} = \frac{P(B_k|B) \cdot P(B)}{P(B_k)} = \frac{b}{r+b}$$

Using conditional probability we can rewrite $P(B \cap BK)$. Now we find $P(B)$, the probability that first picked ball is blue i.e $b/r+b$ as shown above.

$\checkmark B = \text{event that 1st picked ball is blue}$
 $B_k = \text{event that } k \text{ out of } n \text{ balls picked are blue } (b-1)$

$$P(B|B_k) = \frac{P(B \cap B_k)}{P(B_k)} = \frac{P(B_k|B) P(B)}{P(B_k)} \xrightarrow{\substack{b \\ r+b}} \binom{b-1}{k-1} \binom{r}{n-k} \xrightarrow{\substack{b \\ r+b-1}} \binom{b-1}{n-1}$$

Timestamp 16.17

$$P(B_k|B) = \frac{n \left(\begin{array}{c} k-1 \text{ blue balls \& } n-k \text{ red balls} \\ (b-1) \end{array} \right)}{n \left(\begin{array}{c} n-1 \text{ balls from } (r+b-1) \end{array} \right)}$$

$$= \frac{\binom{b-1}{k-1} \cdot \binom{r}{n-k}}{\binom{r+b-1}{n-1}}$$

- Given that we have already picked first ball blue, now we have to find the probability that we pick k blue balls out of n balls i.e. $P(B_k|B)$
- Since we have already picked first ball blue now we choose $k-1$ balls out of $b-1$ blue balls and $n-k$ balls out of r red balls as shown above

Now we have our numerator and denominator, we just substitute the values and we get

$$P(B|B_k) = P(B \cap B_k)/P(B_k) = k/n$$

$$P(E_1 | E_2) = \frac{P(E_1 \cap E_2)}{P(E_2)} \quad \text{if } P(E_2) \neq 0$$

Above is the definition of conditional probability, We can think of multiplication rule as a generalization of conditional probability

Timestamp 22.44

$$\begin{aligned} \Rightarrow P(\underbrace{E_1 \cap E_2}) &= P(E_1 | E_2) P(E_2) \\ P(E_1 E_2) &= P(E_1 | E_2) P(E_2) \end{aligned}$$

$$\text{Gen: } P(\underbrace{E_1 E_2 E_3 \dots E_n}) = P(E_1) \cdot P(E_2 | E_1) \cdot P(E_3 | E_2 E_1) \dots P(E_n | E_{n-1} E_{n-2} \dots E_1)$$

By repeatedly applying conditional probability we obtain the multiplication rule as shown above.

(e.g) Matching problem:

{

n persons $\rightarrow n$ hats
✓ randomly pick hats back

$$\checkmark P(\text{no one correctly their hat}) = \sum_{i=0}^n (-1)^i / i!$$



Ind-End

(Q) $P(\text{exactly } k \text{ persons have picked correctly}) = ?$

$\checkmark A = \{k \text{ persons who would pick correctly}\}$

$E = \text{everyone in } A \text{ has picked correctly}$
 $G = \text{every one other than people in set } A \text{ have picked incor}$

APPLIED COURSE

Timestamp 34.42

$$P(G \cap E) = P(G|E) = P(G|E) \cdot P(E)$$

\downarrow AND \downarrow

$F_1 = \text{event that 1st per in } A \text{ has picked corr.}$

$F_2 = \text{", 2nd "}$

$F_3 = \text{", 3rd "}$

$P(E) = P(F_1 F_2 F_3 \dots F_K)$

$= P(F_1) P(F_2 | F_1) P(F_3 | F_2, F_1) \dots P(F_K | F_1, F_2, \dots, F_{K-1})$

$\frac{n!}{(n-k)!} \frac{(n-k)!}{n!} = \frac{1}{n} \cdot \frac{1}{n-1} \cdot \frac{1}{n-2} \dots \frac{1}{n-k+1}$

- Above is a problem where n persons randomly pick their hats will be solved as shown above

21.8 Independent events

Timestamp 4.54

✓ Independence of events:

(e.g) Expt: Toss a coin & throw a dice

$\begin{cases} E: \text{the coin is } H \\ F: \text{the dice is } 3 \end{cases}$

$p(E \cap F) = \overbrace{p(E|F)}^{\text{def}} p(F) \rightarrow \text{cond. prob.}$

$= p(E) \cdot \frac{1}{6}$

$= \frac{1}{2} \cdot \frac{1}{6} = \frac{1}{12}$



Independent events

$\begin{cases} p(E|F) = p(E) \rightarrow \textcircled{1} \\ p(E \cap F) = p(E) p(F) \end{cases}$

1. Consider an experiment of tossing a coin and throwing a dice. As shown above define two events E and F.
 E = probability of getting a head
 F = probability of getting 3
2. The probability of both events happening $E \cap F$ can be obtained using conditional probability.
 $p(E \cap F) = p(E|F) \cdot p(F)$
3. The fact that F has already happened has no impact on E happening. Event E happening doesn't depend on event F (E is independent if F) so $p(E|F) = p(E)$.
Then
 $p(E \cap F) = p(E) \cdot p(F)$
4. Two events are said to be independent events when their intersection is equal to the product of the product of events.

Timestamp 11.53

<u>NOTE:</u> $\rightarrow P(E \cap F) = P(E) P(F)$ $P(E F) = P(E)$	<u>Independence</u> vs <u>Mutually Exclusive</u> $E \cap F = \emptyset$ $P(E \cap F) = P(\emptyset) = 0$
--	--

- For mutually exclusive events the intersection of events is null , independent events when their intersection is equal to the product of the product of events.

(e.g) 52 Cards

expt: { a } pick a Card randomly
 { pick the 1st Card randomly
 replace the Card
 { pick the 2nd Card randomly

$P(\text{jack and 8}) = P(\text{jack} \cap 8)$
 $= P(\text{jack}) P(8)$
 $= \frac{4}{52} \cdot \frac{4}{52}$

- In the above example probability of picking a second card doesn't depend on the event of picking the first card since we are replacing the first card. They both are independent events .

(eg) (2) Coins (distinct)

$\checkmark E$: 1st coin is H
 $\checkmark F$: 2nd coin is T

$$P(EF) = P(E) P(F)$$

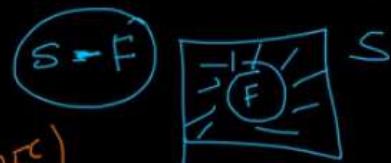
$$= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

Gen: $P(E_1 E_2 \dots E_n) = \underbrace{P(E_1) P(E_2) P(E_3) \dots}_{\text{indep. of each other}} = \prod_{i=1}^n P(E_i)$

- We can extend the concept of independent events and generalize it for n events as shown above. Its nothing but the product of all the events which are independent.

Timestamp 17.9

NOTE: if $E \& F$ are indep then $E \& F^c$ are also indep



$$P(E) = P(E \cap F) + P(E \cap F^c)$$



$$\begin{aligned} P(E) &= P(E \cap F) + P(E \cap F^c) \\ &= P(E) P(F) + P(E \cap F^c) \\ P(E) \{1 - P(F)\} &= P(E \cap F^c) \end{aligned}$$

$$\boxed{P(E) P(F^c) = P(E \cap F^c)}$$

- As shown above if two events E and F are independent then $E \& F^c$ are also independent.

Timestamp 24.30

(e.g) { An infinite seq. of tails | exp (indep) is performed }
 a success prob = $\frac{p}{1-p}$

{ trail: throwing a dice }
 success: outcome is $\underline{1}$
 failure outcome is not $\underline{1}$
 ($p=1/6$)
 ($1-p$)

(a) atleast one success in n tails
 $= 1 - P(\text{no success in } n \text{ tails})$
 $= 1 - P(F_1 F_2 F_3 F_4 \dots F_n)$
 $= 1 - P(F_1) P(F_2) \dots P(F_n) = 1 - \underline{(1-p)}^n$

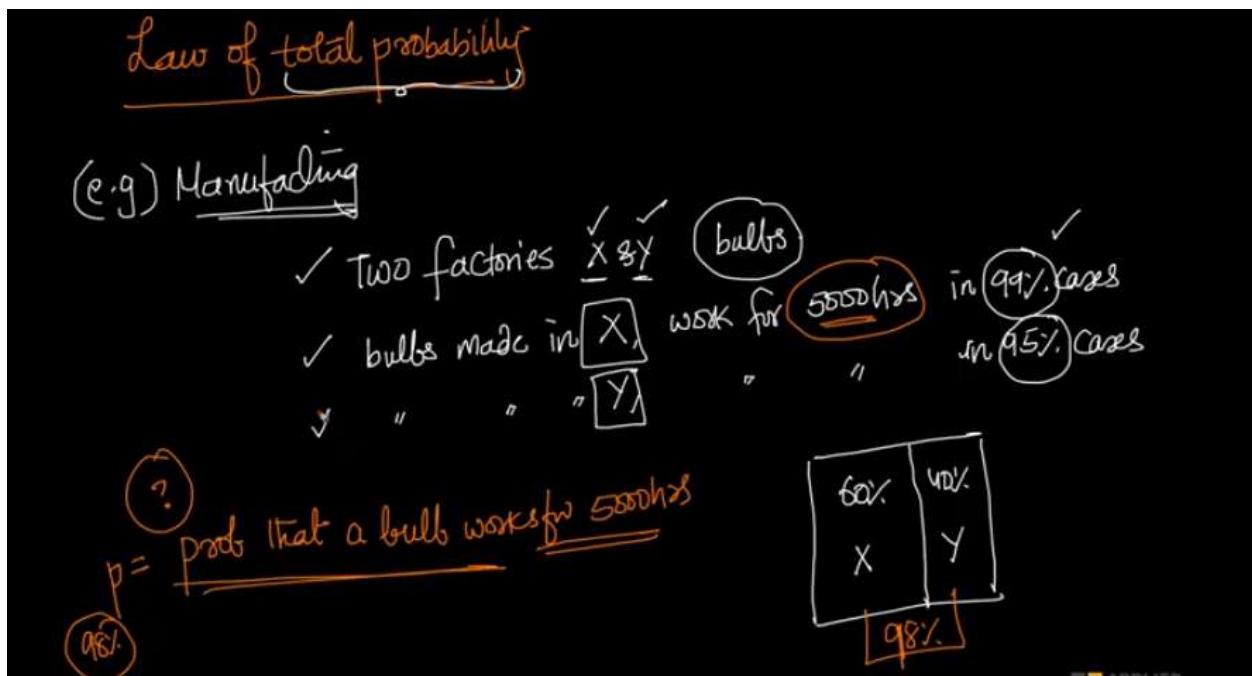
(b) exactly K successes in n trials
 $= {}_n C_K \frac{P^K}{\checkmark} \frac{(1-p)^{n-K}}{\checkmark}$

(c) all n trials are succ.
 $= p \cdot p \cdot \dots \underset{n \text{ times}}{=} p^n$

- Above is an example of using independent events to solve problems.

21.9 Law of total Probability

Timestamp 3.00



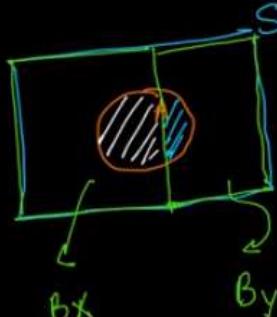
1. Above is an example of use case where we use Law of total probability
2. We have two manufacturing factories X and Y. As shown above if we have a package where we have 60 % of bulbs from factory X and rest 40 % from factory Y. Now we need to find the probability that the bulb works for 5000 hours.

Timestamp 10.6

✓ A: event that a bulb works for 5000 hrs
 B_x: event that a made at X
 B_y: " " " " Y

$$P(A) = P(A \cap B_x) + P(A \cap B_y)$$

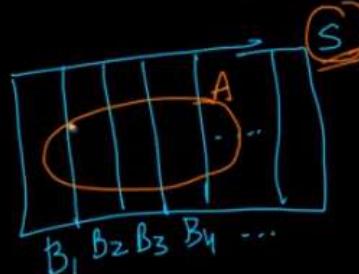
$$= P(A|B_x) P(B_x) + P(A|B_y) P(B_y)$$

$$(0.99 + 0.6) + (0.95 + 0.4)$$


$$\left. \begin{array}{l} S = B_x \cup B_y \\ B_x \cap B_y = \emptyset \end{array} \right\}$$

- We can solve the problem as shown above using total probability.

if B_1, B_2, B_3, \dots

$$\left\{ \begin{array}{l} B_i \cap B_j = \emptyset \rightarrow \text{mutually exclusive} \\ B_1 \cup B_2 \cup B_3 \cup \dots = S \end{array} \right.$$


then,

$$P(A) = P(A \cap B_1) + P(A \cap B_2) + P(A \cap B_3) + \dots$$

$$= P(A|B_1) P(B_1) + P(A|B_2) P(B_2) + \dots$$

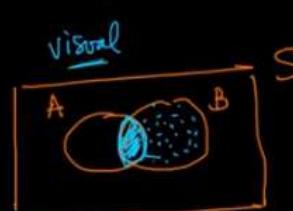
$$P(A) = \sum_i P(A|B_i) P(B_i)$$

- If you have several events $B_1, B_2, B_3, B_4, \dots$ etc. such that all are mutually exclusive events and union of all the events is sample space S (mutually exhaustive events). Then probability of any event A can be written as shown above using law of total probability.

21.10 Bayes Theorem

Timestamp 4.35

Bayes Theorem: \rightarrow Bayesian Stats

$$\checkmark p(A|B) = \frac{p(A \cap B)}{p(B)} \text{ if } p(B) \neq 0$$


$$p(A|B) = \frac{p(A \cap B)}{p(B)}$$

$$p(B) = 0 \Leftrightarrow B = \emptyset$$

- We can obtain Bayes theorem using conditional probability. By assuming that $p(B)$ is not 0 and B is not a null set we can write conditional probability as shown above.

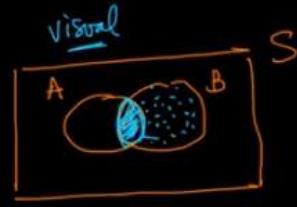
$$p(A|B) = \frac{p(B \cap A)}{p(B)} \xrightarrow{\text{Cond}} = \frac{p(B|A) p(A)}{p(B)}$$

↑
Cond-proba

- We apply conditional probability again as shown above to get Bayes' theorem.

✓ Bayes Theorem: → Bayesian Stats

$$\checkmark P(A|B) = \frac{P(A \cap B)}{P(B)} \text{ if } P(B) \neq 0$$



$$\textcircled{1} \quad P(A|B) = \frac{P(B \cap A)}{P(B)} = \frac{P(B|A) P(A)}{P(B)}$$

Cond-prob Cond-prob Marginal

$$\textcircled{2} \quad P(A|B) = \frac{P(B|A) P(A)}{P(B)} \quad \text{if } P(B) \neq 0$$

Cond-prob Marginal

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

$P(B) = 0 \Leftrightarrow B = \emptyset$

APPLIED COURSE

- The probability of events A and B called marginal probabilities.
- Bayes theorem can also be thought of as an equation that connects the conditional probabilities two events and their marginal probabilities

Timestamp 10.58

Act

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

$$= \frac{P(B|A) P(A)}{P(B \cap A) + P(B \cap A^c)}$$

$$B = (B \cap A) \cup (B \cap A^c)$$

$$\checkmark \left\{ P(A|B) = \frac{P(B|A) P(A)}{P(B|A) P(A) + P(B|A^c) P(A^c)} \right\}$$

- Alternative definition for Bayes theorem can be derived as shown above. Which can be easily derived using set theory.

Medical Diagnosis:

Approx 1% of women in 40-50 have breast cancer

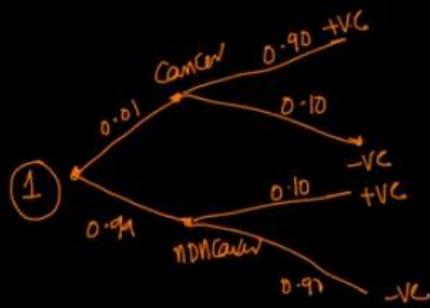
✓ Mammogram (X-ray) → cheapest (not a perfect)

✓ if a woman has breast cancer, the test will result in
+ve value 90% of time

✓ if a woman does not have breast cancer, then the test
+ve → 10% of time

- Above is an example solved using Bayes theorem.
- We have to solve the probability that a woman actually has cancer given the test is positive.

Timestamp 21.17



$$\begin{aligned}
 p(+ve) &= p(+ve \cap \text{Cancer}) \\
 &\quad + p(+ve \cap \text{NOT Cancer}) \\
 &= \cancel{p(+ve | \text{Cancer})} p(\text{Cancer}) \\
 &\quad + p(+ve | \text{No Cancer}) \\
 &\quad \quad \quad p(\text{No Cancer}) \\
 &= (0.9)(0.01) \\
 &\quad + (0.1)(0.99)
 \end{aligned}$$

Timestamp 23.17

✓ prob. that the woman has cancer given the $\underline{\underline{+ve}}$

$$p(\text{cancer} | +ve) = \frac{p(+ve | \text{cancer}) p(\text{cancer})}{p(+ve)}$$

$$= \frac{0.9 * 0.01}{p(+ve)} = \frac{9}{108} \underline{\underline{-8.3\%}}$$

The problem can be solved as shown above

Timestamp 29.21

\checkmark odds \rightarrow belting $\underline{\underline{=}}$

$$\text{odds of } A \text{ happening} = \frac{P(A)}{P(A^c)} = \frac{P(A)}{1-P(A)}$$

$$\checkmark P(A) = \frac{2}{3}, P(A^c) = \frac{1}{3}$$

2

$$\left\{ \begin{array}{l} \text{The odds in favour of } A \text{ are } 2 \text{ to } 1 \\ \text{in } \left[\begin{array}{l} \text{to } 1 \\ \text{in } 2 \end{array} \right] \end{array} \right\} \frac{P(A)}{P(A^c)} = \frac{2}{1}$$

$$P(A) =$$

- There is a concept called odds .The Probability of Event A happening is $\frac{2}{3}$ and not happening is $\frac{1}{3}$ then we can say that the odds in favour if A are 2 to 1 which means there's $\frac{2}{3}$ probability that A ill happen and $\frac{1}{3}$ probability that it won't happen.

21.12 Random variables with examples

Timestamp 5.25

A random variable can be thought of as a mapping or function from an event to a real number. It is also called as a stochastic variable

✓ Random variables:

→ set - Theory (axioms) & counting

→ original to prob

→ Random var: mapping from an event to a R

• i. Toss 3 distinct coins || $8 = 2^3 = |S|$ {H,H,T} → 2

$P(X=1) = {}^3C_1 \frac{1}{2} \cdot \frac{1}{2^2}$ $\boxed{X} = \text{number of heads}$ $P(X=0) = P(\text{all tails}) = \frac{1}{2^3}$

$P(X=2) = {}^3C_2 \frac{1}{2^2} \cdot \frac{1}{2}$ $P(X=3) = \frac{1}{2^3}$

APPLIED

- In the above example of tossing 3 distinct coins let X be a random variable where X=number of heads .Here X is a discrete random variable as it can only take only real values 0,1,2,3.

(2) ✓ $X = \text{amount of rainfall on a given day}$

$\checkmark \underbrace{P(X \geq 2\text{cm})}_{\text{farmer}} = \underline{\underline{0.95}}$

$\checkmark \underbrace{P(X \leq 1\text{cm})}_{\text{farmer}} = \underline{\underline{0.99}}$

$\checkmark P(X \geq 2\text{cm}) = \underline{\underline{0.999}}$

$\checkmark X \in [0, \infty)$

$2\text{cm}, 1.067\text{cm}$

(3) $X = \text{height of students}$

$P(X \geq 180\text{cm}) = \underline{\underline{1\%}}$

buckets | chairs

(4) $X = \text{time spent on a website}$

Let $P(X \geq 10\text{min}) = \underline{\underline{80\%}}$ ✓

Let $P(X \leq 1\text{min}) = \underline{\underline{90\%}}$ (wrong) $\rightarrow \checkmark$

$X = [0, \infty)$

Continuous
 $\begin{cases} 2.1\text{ min} \\ 1.067\text{ min} \end{cases}$

(5) $X = \# \text{ visitors to a website on a given day}$ $\rightarrow \text{discrete}$

$\checkmark P(X \geq 100) = \underline{\underline{0.1\%}}$

$\frac{1}{100} = \underline{\underline{\frac{1}{3000}}}$

S/W & H/W $X = \{0, 1, 2, 3, \dots\}$
 servers

$\begin{cases} 1.5X \\ 2.1X \end{cases}$

(6) $X = \# \text{ children in a family}$

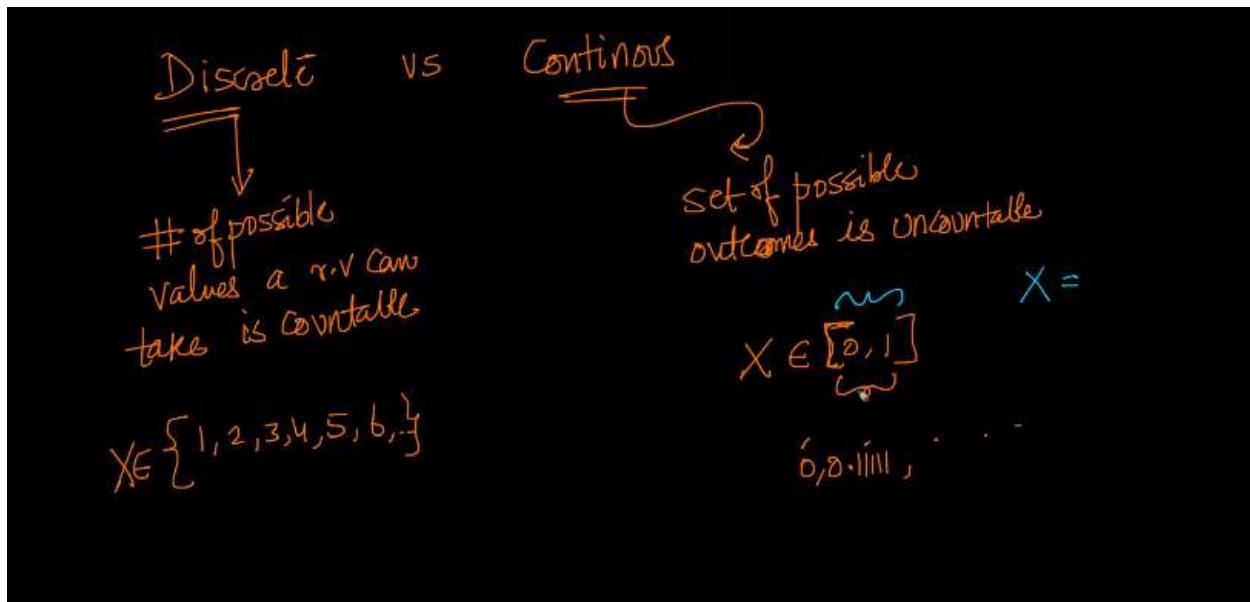
$\checkmark P(X \leq 2) = \underline{\underline{95\%}}$ $\rightarrow \text{discrete}$

$\checkmark P(X = 0) = \underline{\underline{90\%}}$ $\rightarrow \text{Japan} \checkmark$

$X \in \{0, 1, 2, 3, \dots\}$

1. The random variable describing the amount of rainfall on a given day is continuous random variable
2. The random variable describing the height of students is continuous random variable
3. The random variable describing the time spent on a website is a continuous random variable as it can take any value between 0 to infinity.
4. The random variable describing number of visitors to a website on a given day is discrete random variable
5. The random variable describing number of children in a family is discrete random variable

Timestamp 22.20

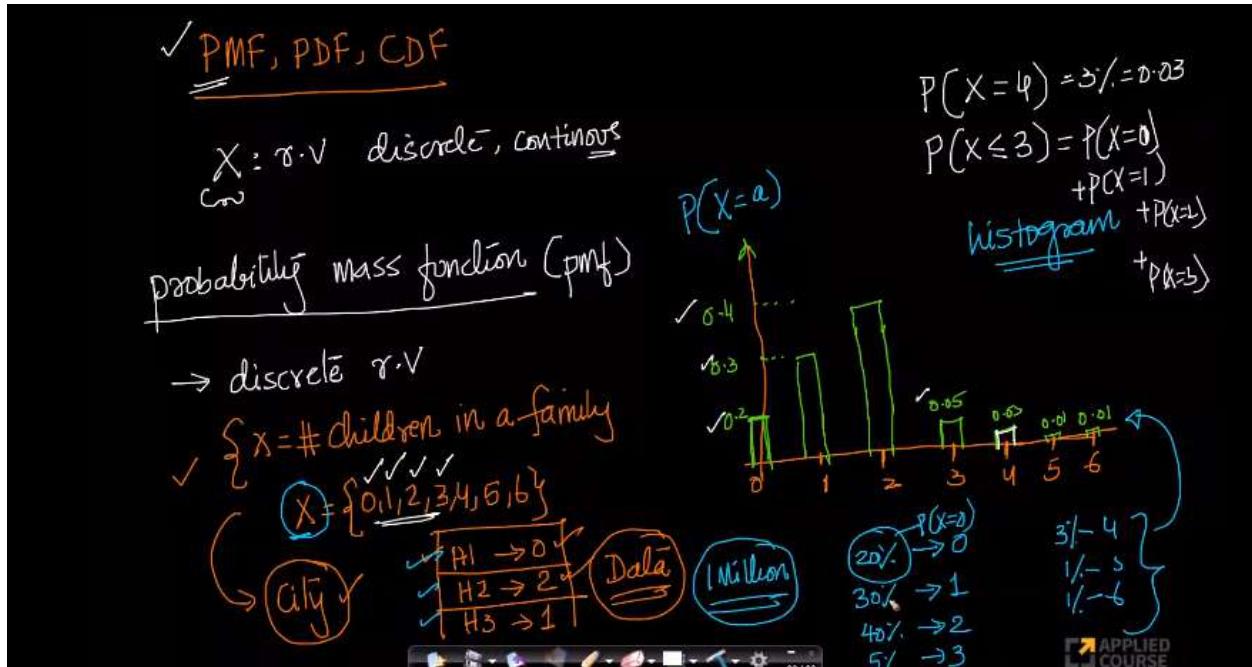


Discrete and continuous random variables are defined as shown above

21.13 PMF, CDF and PDF of random variables

There are three important functions that are defined on random variables PDF,CDF,PMF,lets discuss

Timestamp 8.20



- PMF is described for discrete random variables. Let x be a random variable describing the number of children in a family which is discrete.
- We are plotting $P(x=a)$ on y axis and values of x in x-axis and we draw the histogram as shown above . Using the histogram we can answer questions as shown.

Timestamp 12.25

 function: ✓ $P(X=a) = p(a)$

(e.g.) X : discrete r.v
 $X \in \{0, 1, 2, 3, 4, \dots\}$

 ✓ $\sum P(X=i) = \sum p(i) = C \frac{\lambda^i}{i!}$ for some λ : r.v value

① $P(X=0) = p(0) = C \frac{\lambda^0}{0!} = C$

② $P(X \leq z) = P(X=0) + P(X=1) + P(X=2) + \dots$
 $= C \frac{\lambda^0}{0!} + C \frac{\lambda^1}{1!} + C \frac{\lambda^2}{2!} + \dots = C + \lambda C + \frac{\lambda^2}{2} C$

$X \in \{0, 1, 2, 3, \dots\}$

✓ $P(0) + P(1) + P(2) + P(3) + \dots = 1$

✓ $\sum_{i=0}^{\infty} C \frac{\lambda^i}{i!} = C \underbrace{\sum_{i=0}^{\infty} \frac{\lambda^i}{i!}}_{= e^\lambda} = C e^\lambda = 1 \Rightarrow C = \frac{1}{e^\lambda} = e^{-\lambda}$

- PMF is a function which gives the probability that random variable X takes value a i.e) $P(X=a)$ or $P(a)$
- As shown above if we have the PMF function we can easily find the probability of random variable X taking value a .
- The sum of all the probabilities that variable x can take will be equal to 1.

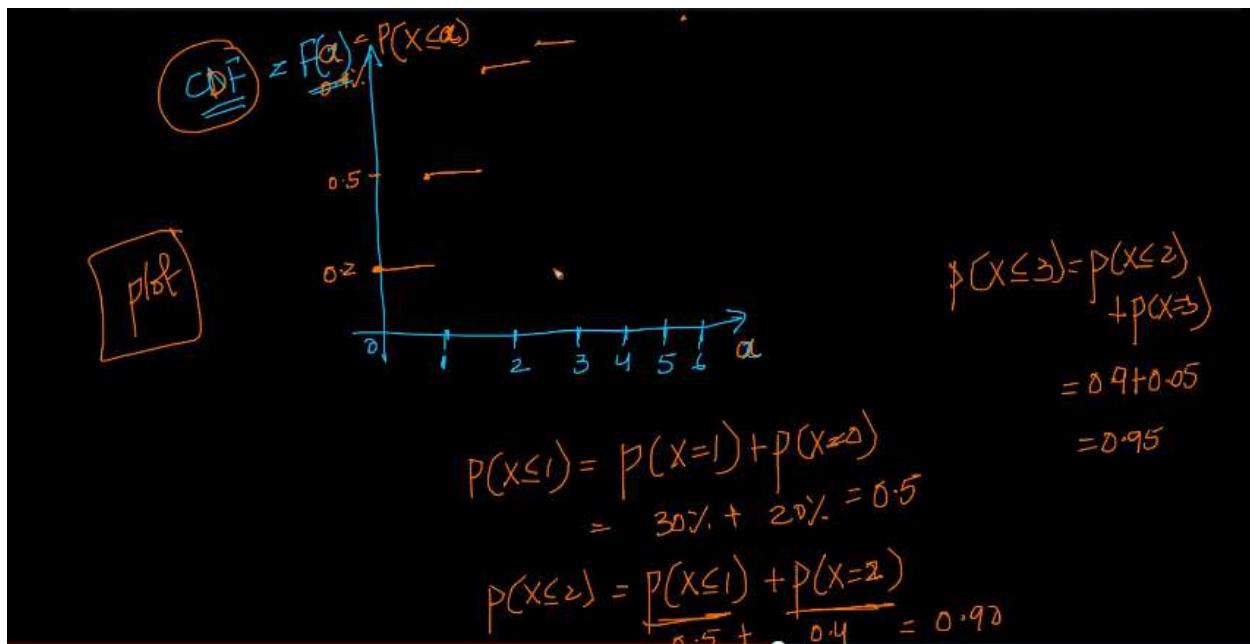
Cumulative distribution Function (CDF)

X : discrete RV

$$X \in \{0, 1, 2, \dots\}$$

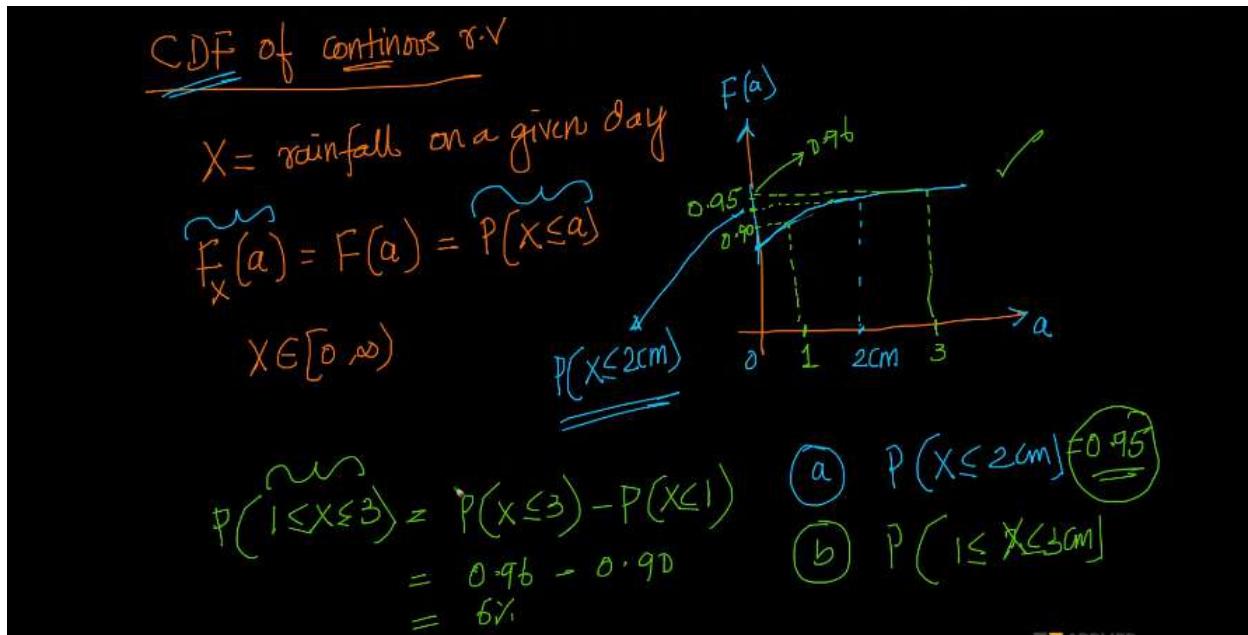
$$\begin{aligned} F_X(a) &= \sum_{x \leq a} p(x) = p(X=a) + p(X=a-1) + p(X=a-2) + \dots + p(X=0) \\ &\checkmark F(a) = \sum_{x \leq a} p(x) = \sum_{x \leq a} p(x) \end{aligned}$$

- Cumulative distribution function of a Random variable X at a point a , where a is one of the values random variable X can take is probability that X can take value which is less than or equal to a as shown above



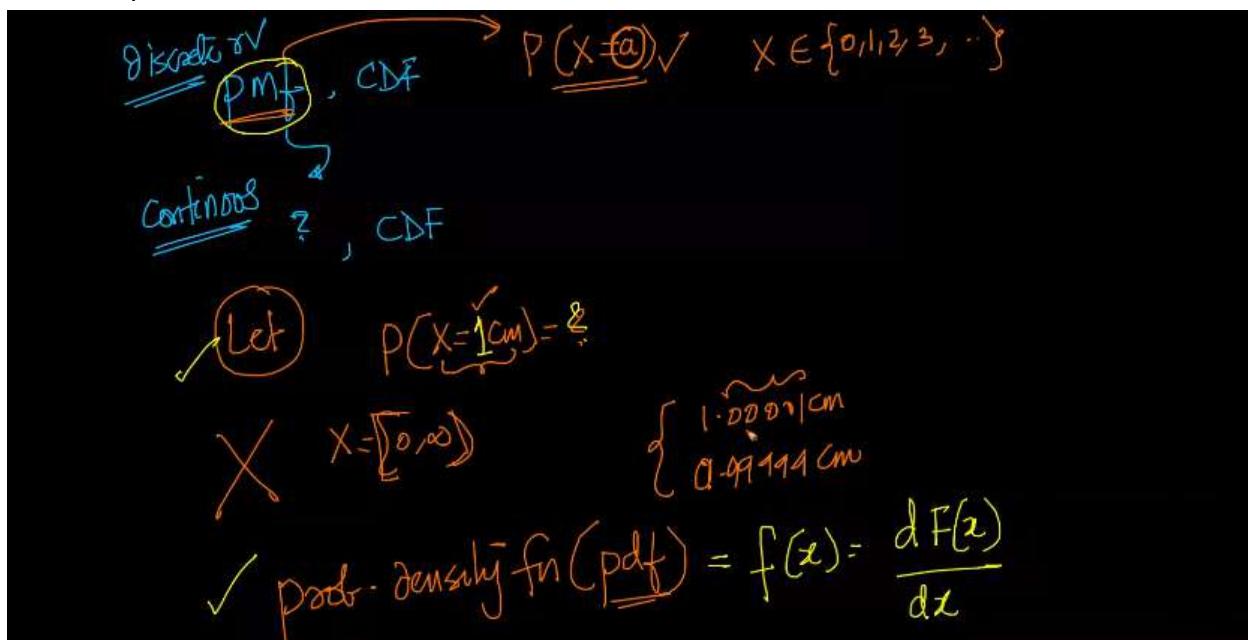
- CDF of a discrete random variable can be plotted as shown above .

Timestamp 30.34

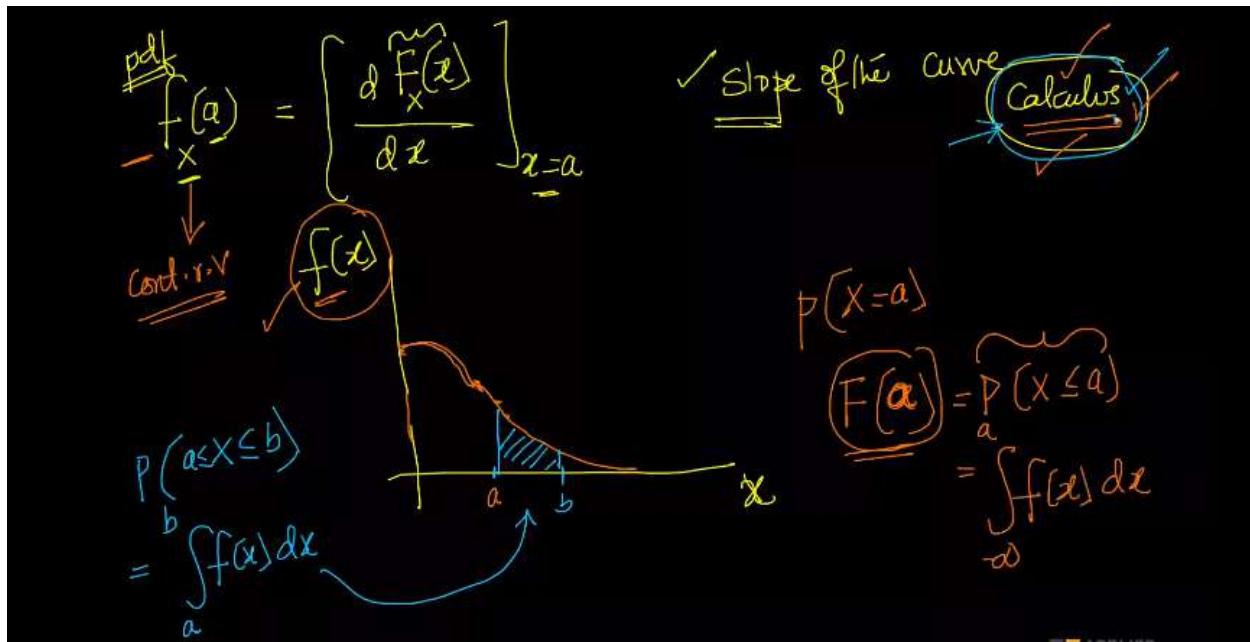


- CDF of a continuous random variable is explained above. We can draw a plot for cdf function and using it we can easily find probabilities as shown.

Timestamp 30.17



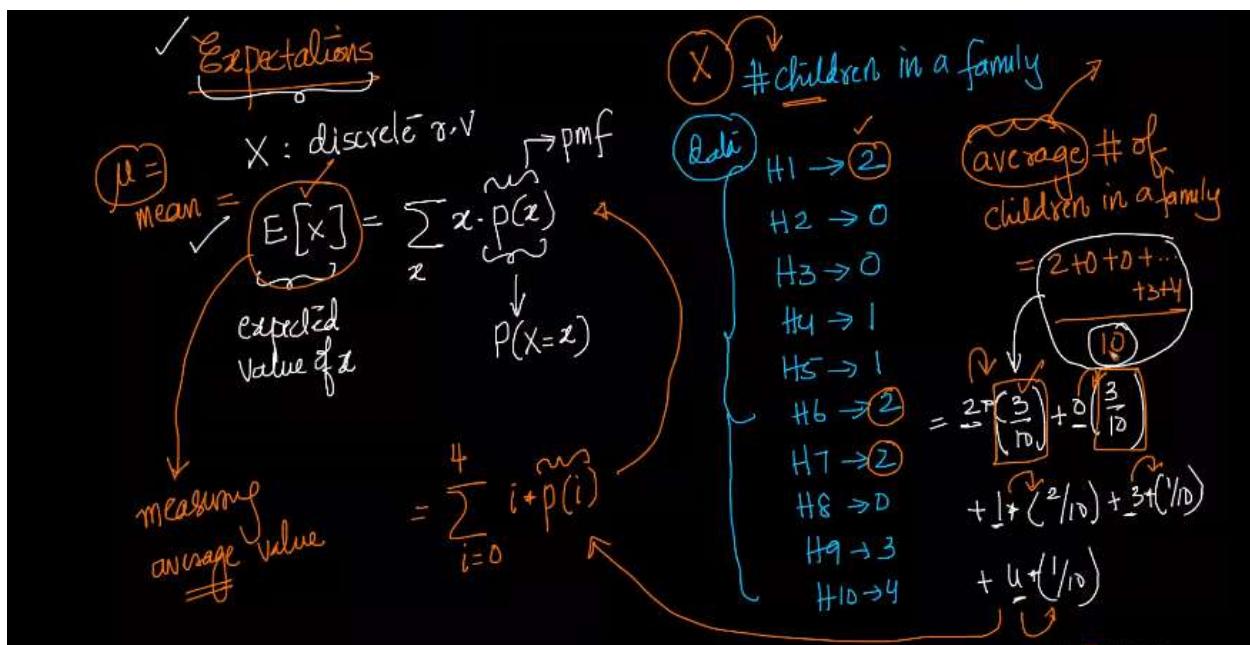
- We have CDF defined for both discrete (curve is not smooth) and continuous random variables (has smooth curve) but PMF is defined for only discrete random variables and for continuous random variables we have PDF (Probability density function).



- We define the PDF as shown above .

21.14 Expectation

Timestamp



- Expectation can be defined for a discrete random variable. Expectation of x can be written as shown above ,we can understand it using an example shown above.
- Intuitively expectation is measuring the average value of random variable x based on some data.

Timestamp 11.21

$\mu = \mu_x = E[X] = \sum_x x \cdot p(x)$ ← discrete r.v

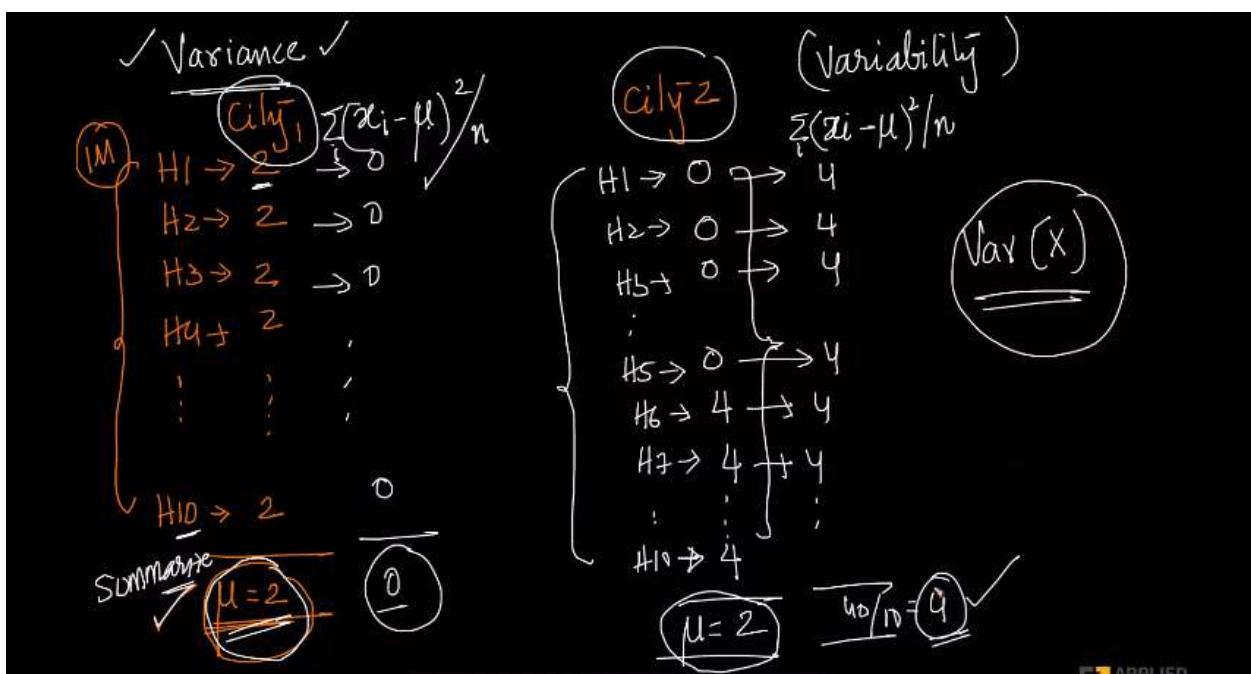
$\mu_x = E[X] = \int x \cdot f(x) dx$ ← continuous r.v

↳ $X = \text{rainfall on a day}$

$\boxed{\text{City}}$ $\boxed{\text{Baghdad}}$ $\boxed{\text{Seattle}}$

avg. rainfall
high → wet

- Expectation for a continuous random variable can be obtained as shown above .



- Variance can be explained using the example shown above.
- If we are given average of number of children in household in two different cities μ, μ^* . we can clearly notice there is significant variability between number of children in the households in two cities, only when we look into the data we can notice the variability.
- Clearly only mean cannot describe the data perfectly ,so to describe such a phenomenon in the data we use variance as a measure.
- As shown variance of the data of City1 is 0 and the variance of City 2 is 4 Which makes us understand that there is a huge variability in the data of city 2.
- Variance measures variability in the data

The image shows a handwritten derivation of the variance formula. On the left, the formula for variance is derived from the definition of expectation:

$$\text{Var}(X) = \frac{\sum_i (x_i - \mu)^2}{n}$$

An arrow points from this to the formula for the expected value of a function:

$$E[g(x)] = \int x f(x) dx$$

Below these, the formula for variance is expanded:

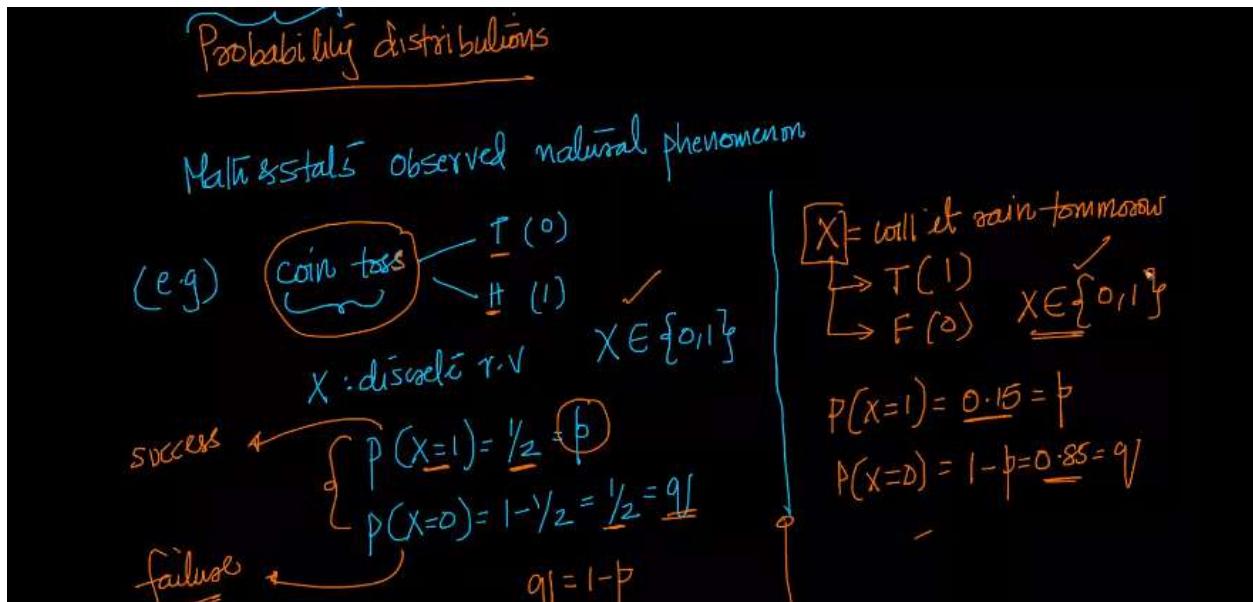
$$\begin{aligned} ? &= E[X^2] - 2\mu E[X] + \mu^2 \\ &= E[X^2] - 2\mu \tilde{E}[X] + \mu^2 \\ &= E[X^2] - 2\mu^2 + \mu^2 = E[X^2] - \mu^2 \end{aligned}$$

On the right, three equivalent ways to express the expected value of a function are shown:

$$\begin{aligned} E[g(x)] &= \sum_x g(x) p(x) \\ g(x) f(x) & \quad \int g(x) f(x) dx \end{aligned}$$

21.15 Probability Distributions: Bernoulli and Binomial

Timestamp 2.05



- Let's consider the above examples and understand the probability distributions
- As shown in the experiment of coin toss X is a random variable and probability of x taking a value 1(getting head) is p and probability of x taking a value 0(getting tail) is q . Here $q=1-p$
- Similarly we can understand the example of random variable x where it describes whether it will rain tomorrow as p . Probability of not raining is $q=1-p$
- Intuitively p,q are nothing but probability of success and failure.
- Similar to the above examples we can notice the same pattern in the below example as well.

Timestamp 7.14

(e.g) $X = \text{will a customer purchase a product}$ (ecommerce)

$$\begin{cases} \rightarrow Y(1) \\ \rightarrow N(0) \end{cases} \quad X \in \{0, 1\}$$

$$\begin{aligned} P(X=1) &= 0.05 \Rightarrow \\ P(X=0) &= 0.95 \Rightarrow \end{aligned}$$

(e.g) gender of a newborn $\begin{cases} M(0) \\ F(1) \end{cases}$

$$\begin{aligned} X \in \{0, 1\} \quad P(X=1) &= 0.5 \Rightarrow \\ P(X=0) &= 0.5 \Rightarrow \end{aligned}$$

Timestamp 11.18

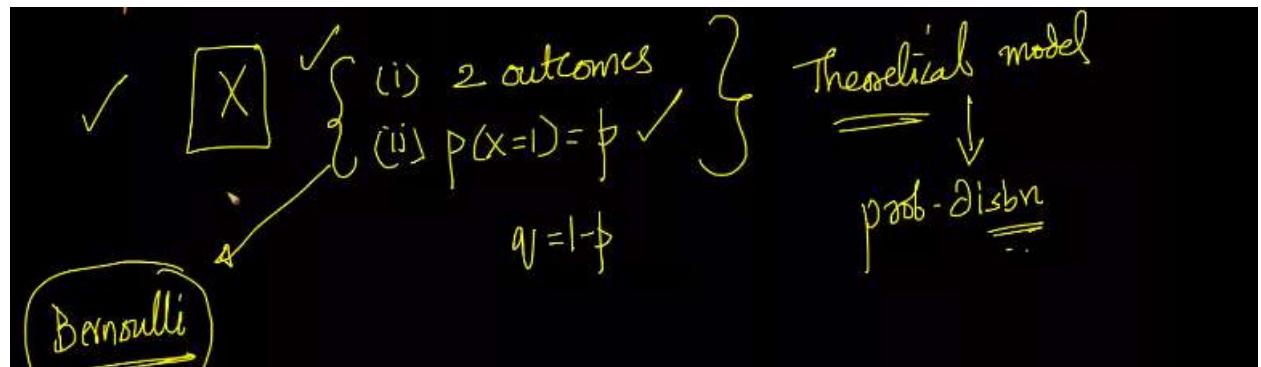
$$X \in \{0, 1\}$$

$$\begin{cases} P(X=1) = p = \text{success prob} \\ P(X=0) = 1-p = q = \text{failure prob} \end{cases}$$

⑤ will a person have accident $\begin{cases} Y(1) \\ N(0) \end{cases}$
 $P = 0.0001$

✓ \boxed{X} ✓ $\begin{cases} (i) 2 \text{ outcomes} \\ (ii) P(X=1) = p \quad p = 1-q \end{cases}$ ✓ $\begin{cases} \text{Theoretical model} \\ \text{prob-distrn} \end{cases}$

- The pattern we have observed in all the above examples is put together as shown above
- Statisticians thought that if we can study the patterns i.e)probability distributions and come to certain conclusions(ideas and outcomes),we can apply all those conclusions to any of the examples we have discussed.



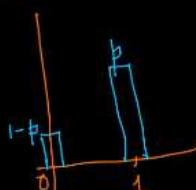
- If the random variable follows above two properties then it belongs to Bernoulli distribution.
- When we say a random variable belongs to the Bernoulli distribution it will have two outcomes and if we get probability of success P then we can say everything about this random variable

Timestamp 15.26

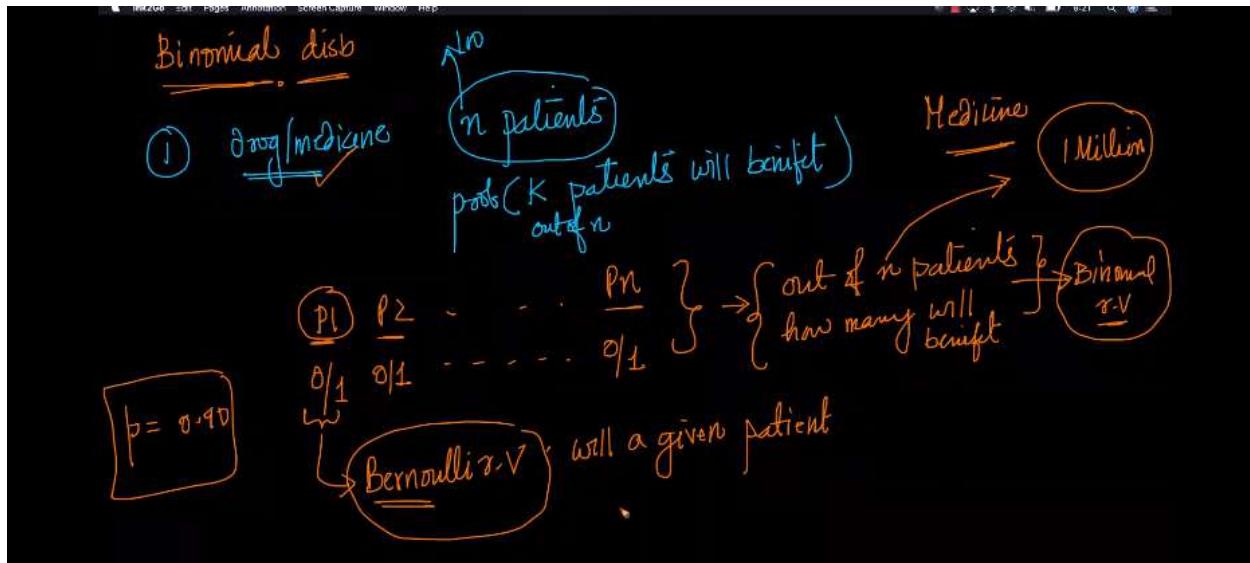
$$\begin{aligned}
 \text{Var}(X) &= E[X^2] - \mu^2 \\
 &= \sum_x x^2 p(x) - \mu^2 \\
 &= 0 \cdot q + 1 \cdot p - \mu^2 \\
 &= p - \mu^2 \\
 &= p(1-p) \\
 &= pq
 \end{aligned}$$

$\checkmark X \sim \text{Bernoulli}(p)$ p : param of the dist.
 $X \in \{0, 1\}$ X : discrete RV

$$\begin{aligned}
 p &= P(X=1) \\
 \text{mean}(X) &= E(X) = \mu \\
 &= \sum_x x \cdot p(x) \\
 &= 0 \cdot q + 1 \cdot p \\
 &= p
 \end{aligned}$$

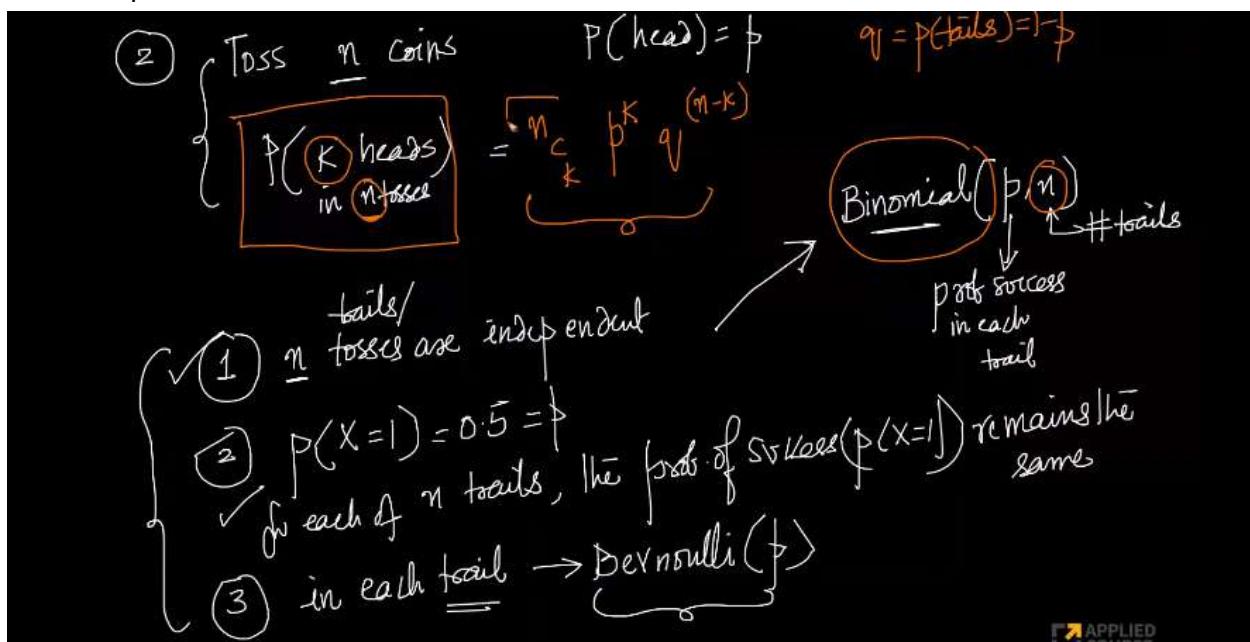
$\checkmark \text{pmf:}$ 
 $\checkmark \text{cdf:}$ 

- The PMF,CDF, mean,variance of a random variable belonging to Bernoulli distribution as shown above.



- Let's understand Binomial distribution using the above example
- Let's say we have manufactured a drug and we want to find the probability that how many patients will benefit out of n patients. We know that the outcome of drug on patient 1 is independent of the outcome of drugs on patient 1. For each patient there are two outcomes 0/1 (p/q) drug didn't help the patient and drug did help the patient this is nothing but Bernoulli distribution.

Timestamp 31.46



- If a real world phenomenon satisfies above three conditions then it is said to belong to Binomial Distribution as shown above.

Timestamp 33.41

$$X \in \{0, 1, 2, \dots, n\}$$

$$P(X=k) = P(X=k) = \binom{n}{k} p^k (1-p)^{n-k} \xrightarrow{\text{PDF}}$$

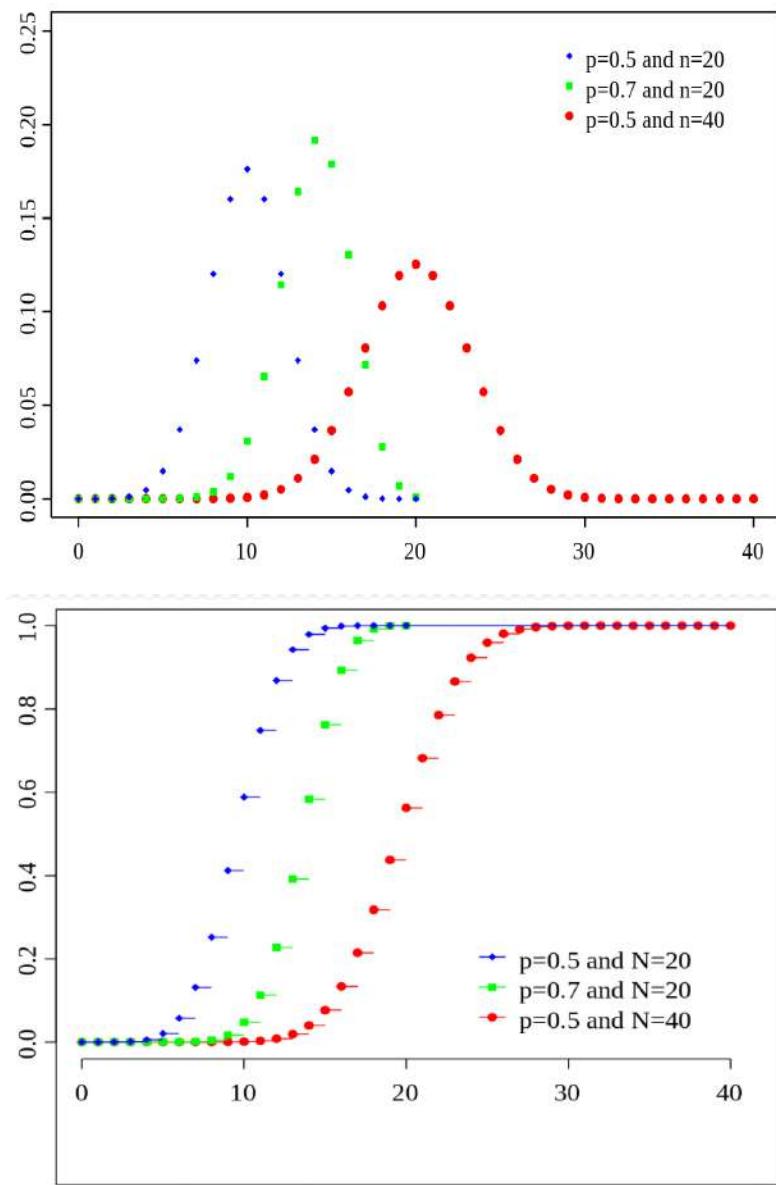
$$P(X \leq k) = \sum_{i=0}^k \binom{n}{i} p^i (1-p)^{n-i} \xrightarrow{\text{CDF}}$$

$$\begin{aligned} \mu &= E(X) = E(X_1 + X_2 + \dots + X_n) \\ &= E(X_1) + E(X_2) + \dots + E(X_n) \\ &= p + p + \dots \underset{n \text{ times}}{+} \\ &= \underbrace{n}_{\text{number of trials}} p \end{aligned}$$

$$\mu = E(X) = \sum_{k=0}^n k \cdot P(X=k) = \sum_{k=0}^n k \cdot \binom{n}{k} p^k (1-p)^{n-k}$$

$$\begin{aligned} \text{Var}(X) &= \text{Var}(\underbrace{X_1 + X_2 + X_3 + \dots + X_n}_{\text{independent}}) \\ &= \text{Var}(X_1) + \text{Var}(X_2) + \dots + \text{Var}(X_n) \\ &= pq + pq + \dots \underset{n \text{ times}}{+} \\ &= npq = \underline{n} \underline{p} \underline{(1-p)} \end{aligned}$$

- PDF,CDF,mean,variance of a Binomial distribution can be derived as shown above



- Above are the pdf and cdf curves of a binomial distribution ,the curves are not continuous ,they are discrete.

21.16 Poisson Distribution

Timestamp

Poisson Distribution

(e.g.) # calls received at a call-center per hour

→ # events occurred per unit time

(e.g.) # patients in emergency between 10PM-11PM

(e.g.) # customers at the counter per hour

(e.g.) # insurance claims in a year

(e.g.) # goals in sport between 2 teams

APPLIED

Occurrence [edit]

Applications of the Poisson distribution can be found in many fields related to counting:^[26]

- Telecommunication example: telephone calls arriving in a system.
- Astronomy example: photons arriving at a telescope.
- Chemistry example: the molar mass distribution of a living polymerization^[27]
- Biology example: the number of mutations on a strand of DNA per unit length.
- Management example: customers arriving at a counter or call centre.
- Finance and insurance example: number of losses or claims occurring in a given period of time.
- Earthquake seismology example: an asymptotic Poisson model of seismic risk for large earthquakes.^[28]
- Radioactivity example: number of decays in a given time interval in a radioactive sample.

- We can understand poisson distribution using above examples
- Poisson distribution measures the number of events occurred per unit time

$$X \sim \text{Poisson}(\lambda) \quad X = \{0, 1, 2, 3, 4, \dots\}$$

rate

pmf: $P(X=k) = \frac{\lambda^k e^{-\lambda}}{k!}$ $\sum_{x=0}^{\infty} \frac{\lambda^x e^{-\lambda}}{x!} = e^{-\lambda} e^{\lambda} = 1$

CDF: $P(X \leq k) = \sum_{x=0}^k e^{-\lambda} \frac{\lambda^x}{x!}$

- Given λ the rate parameter of the poisson distribution PMF,CDF can be derived as shown.(the derivations are obtained from lot of observations empirically)

Timestamp 18.32

$$\begin{aligned}
 \mu = E[X] &= \overline{x} = \sum_{x=0}^{\infty} x \cdot \frac{e^{-\lambda} \lambda^x}{x!} = \lambda e^{-\lambda} \sum_{x=1}^{\infty} \frac{\lambda^{x-1}}{(x-1)!} \\
 &= \lambda e^{-\lambda} \underbrace{\sum_{x=1}^{\infty} \frac{\lambda^{x-1}}{(x-1)!}}_{= e^{\lambda}} \\
 &= \lambda e^{-\lambda} \left\{ \frac{\lambda^0}{0!} + \frac{\lambda^1}{1!} + \dots \right\} \\
 &= \lambda e^{-\lambda} e^{\lambda} = (\lambda)
 \end{aligned}$$

$$\begin{aligned}
 \text{Var}(X) &= E(X^2) - \mu^2 \\
 &= E(X(X_1 + X)) - \mu^2 \\
 &= E(X(X_1)) + E(X) - \mu^2 \\
 &= \underbrace{\sum_{x=0}^{\infty} x^2 e^{-\lambda} \frac{\lambda^x}{x!}}_{x^2} + \lambda - \mu^2 \\
 &= \sum_{x=0}^{\infty} x(x_1) \frac{e^{-\lambda} \lambda^x}{x!} = x^2 e^{-\lambda} \sum_{x=2}^{\infty} \frac{\lambda^{x-2}}{(x-2)!} \\
 &\quad - \lambda^2 e^{-\lambda} e^\lambda = \lambda^2
 \end{aligned}$$

The mean and variance of poisson distribution is λ

Timestamp 21.59

$$X_1 \sim \text{Poisson}(\lambda_1)$$

$$X_2 \sim \text{Poisson}(\lambda_2)$$

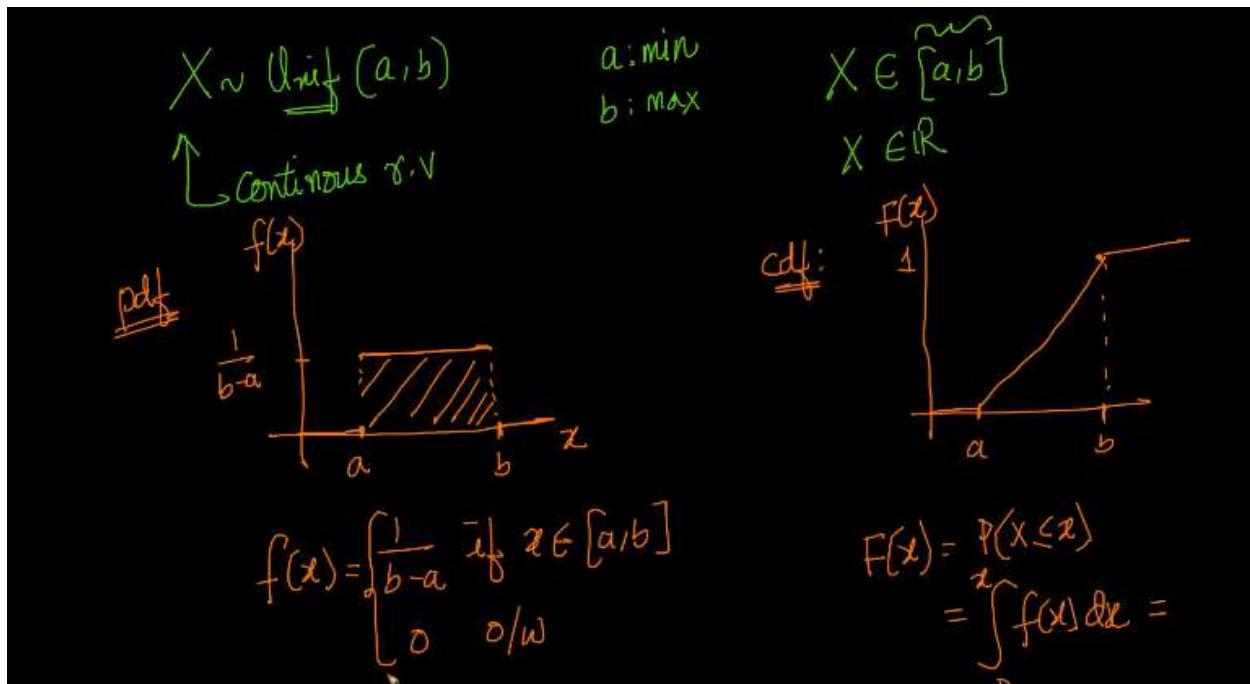
X_1 & X_2 are indep

$$(X_1 + X_2) \sim \text{Poisson}(\lambda_1 + \lambda_2)$$

- Interesting property of a poisson variable is If X_1 X_2 are two random variables belonging to poisson distribution with λ_1 and λ_2 as rates and X_1 and X_2 are independent then the random variable $X_1 + X_2$ also belongs to poisson distribution with rate $\lambda_1 + \lambda_2$

21.17 Uniform (continuous) distribution

Timestamp 9.17



$$F(x) = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } x \in [a, b] \\ 1 & \text{if } x > b \end{cases}$$

$$\mu = E[X] = \int_{-\infty}^{\infty} x \cdot f(x) dx = \int_{-\infty}^a x \cdot 0 dx + \underbrace{\int_a^b x \cdot \frac{1}{b-a} dx}_{0} + \int_b^{\infty} x \cdot 0 dx$$

$$= \left[\frac{x^2}{2(b-a)} \right]_a^b = \frac{b^2 - a^2}{2(b-a)} = \left(\frac{b+a}{2} \right)$$

$$\text{Var}(X) = E(X^2) - \mu^2$$

estimation

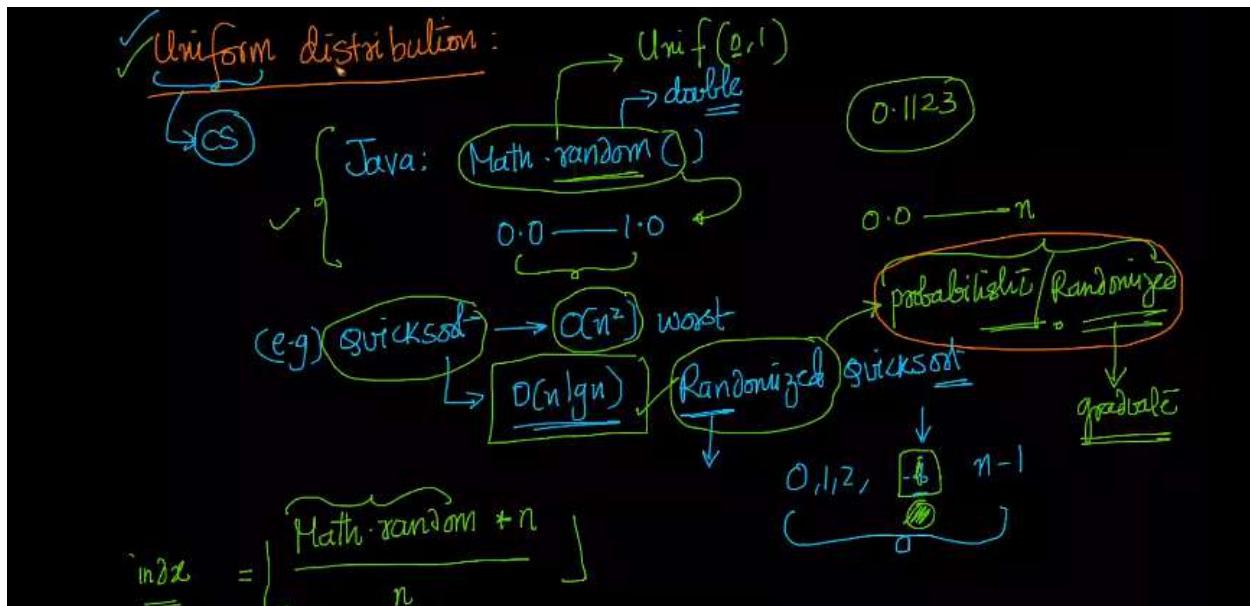
$$= \int_a^b x^2 \frac{1}{b-a} dx - \frac{(a+b)^2}{4}$$

$$= \left[\frac{x^3}{3(b-a)} \right]_a^b - \frac{(a+b)^2}{4}$$

$$= \frac{b^3 - a^3}{3(b-a)} - \frac{(a+b)^2}{4}$$

$$= \frac{1}{3} (b^2 + ab + a^2) - \frac{(a+b)^2}{4} \rightarrow \underline{\underline{\frac{1}{12}(b-a)^2}}$$

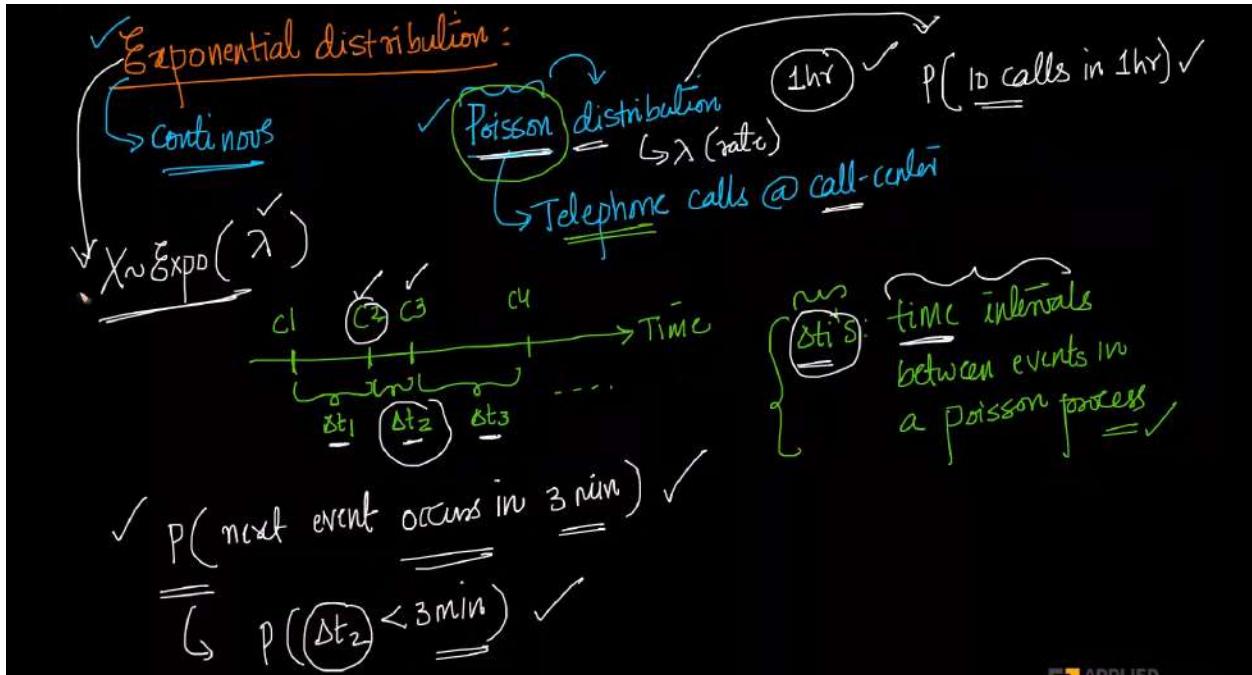
- A uniform random variable X can take value in between a,b including a,b .The PDF,CDF,mean,variance can be defined as shown above



- There are lots of applications for uniform random variables in computer science especially in probabilistic and randomized algorithms.

21.18 Exponential Distribution

Timestamp 6.22



- In the example of Telephone calls in a call center ,using poisson distribution we can answer questions like probability of getting 10 calls in one hour if we know λ ,but we cannot answer questions like probability that the next event occurs in three minutes.
- Δt 's are the time intervals between poisson process.Exponential distribution knows about Δt so we'll be able to answer such questions.

Timestamp 8.20

$$\begin{aligned}
 \text{pdf} \quad f(x) &= \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} & \int e^{\lambda t} dt = \frac{1}{\lambda} e^{\lambda t} \\
 \text{CDF} \quad F(x) &= P(X \leq x) = \int_0^x \lambda e^{-\lambda t} dt = \left[-\frac{1}{\lambda} e^{-\lambda t} \right]_0^x \\
 &= \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}
 \end{aligned}$$

- The PDF and CDF of Exponential distribution are as shown above

Timestamp 14.20

$$\mu = E(X) = \int_0^{\infty} x \cdot (\lambda e^{-\lambda x}) dx$$

→ substitution ✓
Integration by parts

$u = x$
 $du = dx$
 $dv = \lambda e^{-\lambda x}$
 $v = -e^{-\lambda x}$

$\int u dv = uv - \int v du$

$\int_0^{\infty} x \cdot (\lambda e^{-\lambda x}) dx = \left[-x e^{-\lambda x} \right]_0^{\infty} + \int_0^{\infty} e^{-\lambda x} dx$
 $= 0 + \left[\frac{1}{-\lambda} e^{-\lambda x} \right]_0^{\infty} = \left[\frac{1}{\lambda} \right]$ ✓

$$Var(X) = E(X^2) - \mu^2 \quad \mu = \lambda$$

$\int_0^{\infty} x^2 \cdot \lambda e^{-\lambda x} dx$ Integration by parts

$\int_0^{\infty} u v dv = \left[u v \right]_0^{\infty} - \int_0^{\infty} u' v dv$

$= \left[\frac{1}{\lambda^2} \right]$

- Mean and variance of exponential distribution can be derived as shown above

Timestamp 16.52

estimatio $\hat{\lambda}$

under event time

$\text{obs} \left\{ x_1, x_2, x_3, \dots, x_n \right\}$

$$\mu = \frac{\sum_{i=1}^n x_i}{n} \approx \frac{1}{\hat{\lambda}}$$

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^n x_i}$$

- We can estimate lambda using inter event time as shown above

Memoryless property \rightarrow continuous

$$Pr(X > s+t | X > s) = Pr(X > t) \quad \forall s, t \geq 0$$

$$= \frac{Pr(X > s+t \cap X > s)}{Pr(X > s)}$$

$$= \frac{Pr(X > s+t)}{Pr(X > s)} = \frac{1 - Pr(X \leq s+t)}{1 - Pr(X \leq s)} = \frac{e^{-\lambda(s+t)}}{e^{-\lambda s}} = e^{-\lambda t} = Pr(X > t)$$

Event 1 Event 2

3 sec 1 sec

- Above is an important property of the exponential distribution

21.19 Normal/gaussian Distribution

Timestamp 4.00

Normal / Gaussian distribution → continuous

→ widely used / popular

→ e.g.: approx. Gaussian → {heights, length of leaves, weights, ...} → natural phenomenon

Math/Stats: theoretically $X \sim \text{Normal}$

ML & AI

$X \sim N(\mu, \sigma^2)$

mean variance
finite mean & var

$\sigma = \text{Std-Dev} = \sqrt{\text{Var}}$

- Normal distribution is a widely used distribution by statisticians and mathematicians
- For a random variable to be normally distributed it needs to have a finite mean and finite variance

Timestamp 6.55

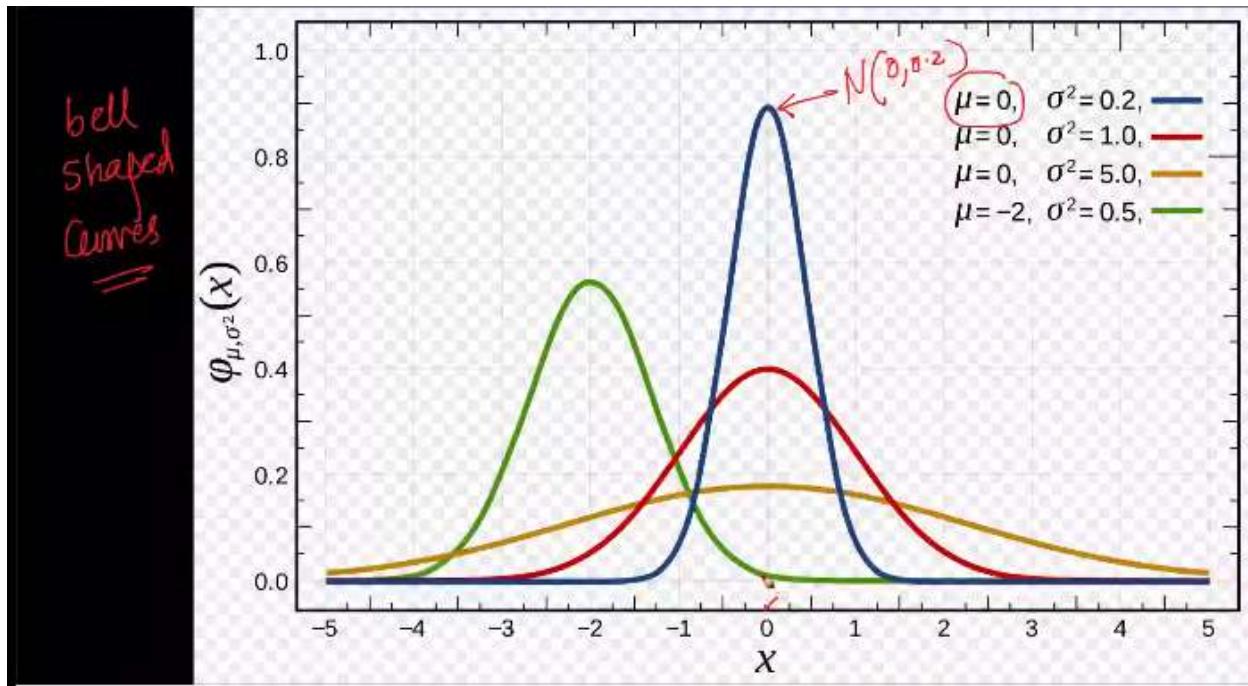
pdf: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$

$\sigma \neq 0$

const

$\int_{-\infty}^{\infty} f(x) dx = 1$

$f(x) > 0$



- We will have mean and variance of Normal Distribution as shown above we can easily calculate the PDF function
- PDF Curve of Normal Distribution are bell-shaped

$$\text{CDF} = \int_{-\infty}^x f(x) dx$$

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

$$P(X \leq x)$$

- CDF of the normal distribution

$$\begin{cases} E(X) = \mu \\ \text{Var}(X) = \sigma^2 \end{cases}$$

estimation

$\underline{x_1, x_2, x_3, \dots, x_n} \leftarrow \text{heights}$

$\hat{X} \sim N(\mu, \sigma^2)$

$\hat{\mu} = \frac{\sum x_i}{n}$ $\hat{\sigma}^2 = \frac{\sum (x_i - \hat{\mu})^2}{n}$

- As shown above you can estimate the mean and variance of the normal distribution

Timestamp 17.02

Standard Normal Dist

The diagram illustrates the transformation of a general normal variable $X \sim N(\mu, \sigma^2)$ into a standard normal variable $Z \sim N(0, 1)$. A curved arrow points from the general normal distribution to the standard normal distribution. Below the general normal distribution, the formula $Z = \frac{X - \mu}{\sigma}$ is written, with arrows indicating the subtraction of the mean and division by the standard deviation. To the right, a box labeled "Properties" contains a small diagram of a bell-shaped curve.

\downarrow
std. normal variable

- We construct a new variable Z using the existing variable by subtracting the variable from mean and dividing it by variance. It is called a standard normal variable belonging to standard normal distribution. Z will have approximately a mean value 0 and variance 1.

22.1 CDF of a Gaussian/Normal Distribution

The CDF value for any distribution always ranges in between 0 and 1. The left bottom tail of the CDF curve has a value of 0 and the right top tail has a value of 1. Let us assume we have a random variable 'X' and the CDF at a point 'x' indicates the probability of 'X' taking the values that are less than or equal to 'x'.

$$\text{CDF}(X=x) = P(X \leq x)$$

A normal distribution is represented as $X \sim N(\mu, \sigma^2)$ where

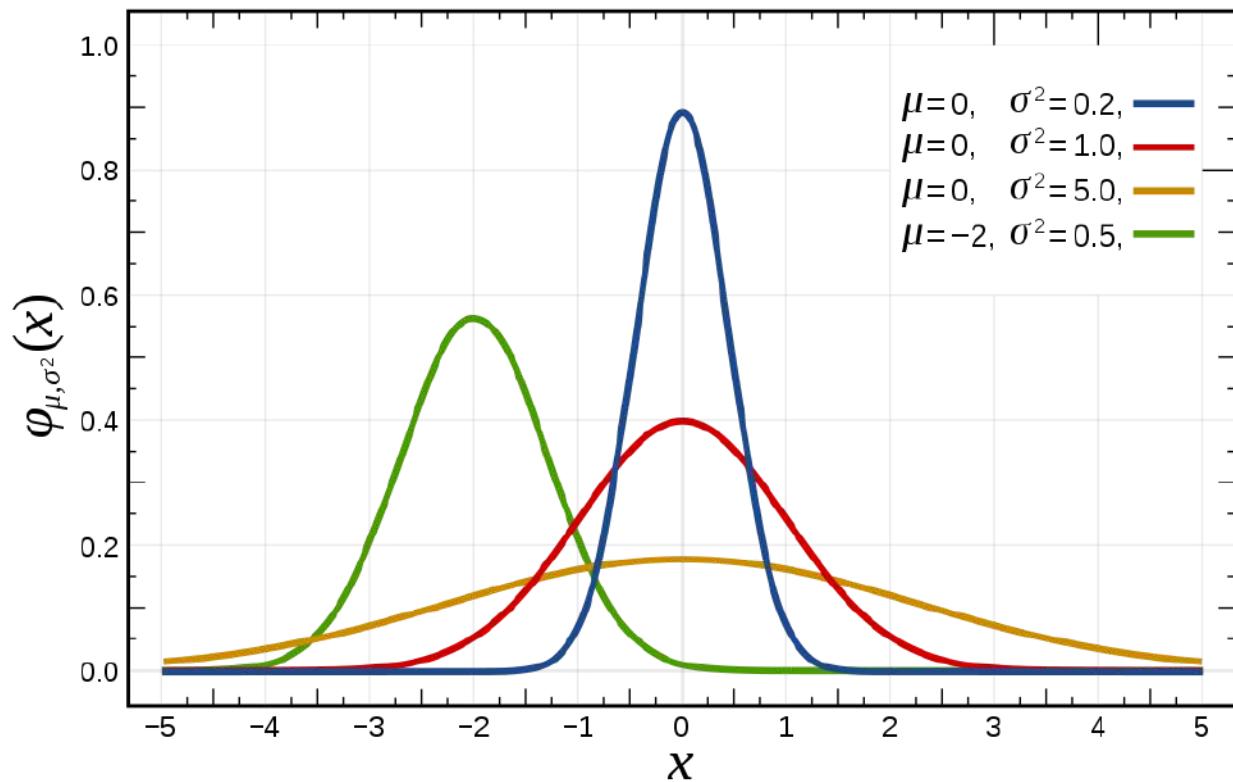
$\mu \rightarrow$ Mean of 'X'

$\sigma^2 \rightarrow$ Variance of 'X'

If the given distribution is symmetric, then 50% of the points lie to the left side of the mean and the remaining 50% of them lie to the right side of the mean. So for such distributions, the CDF at the mean is always equal to 0.5.

Note: This logic is applicable only for the symmetric distributions.

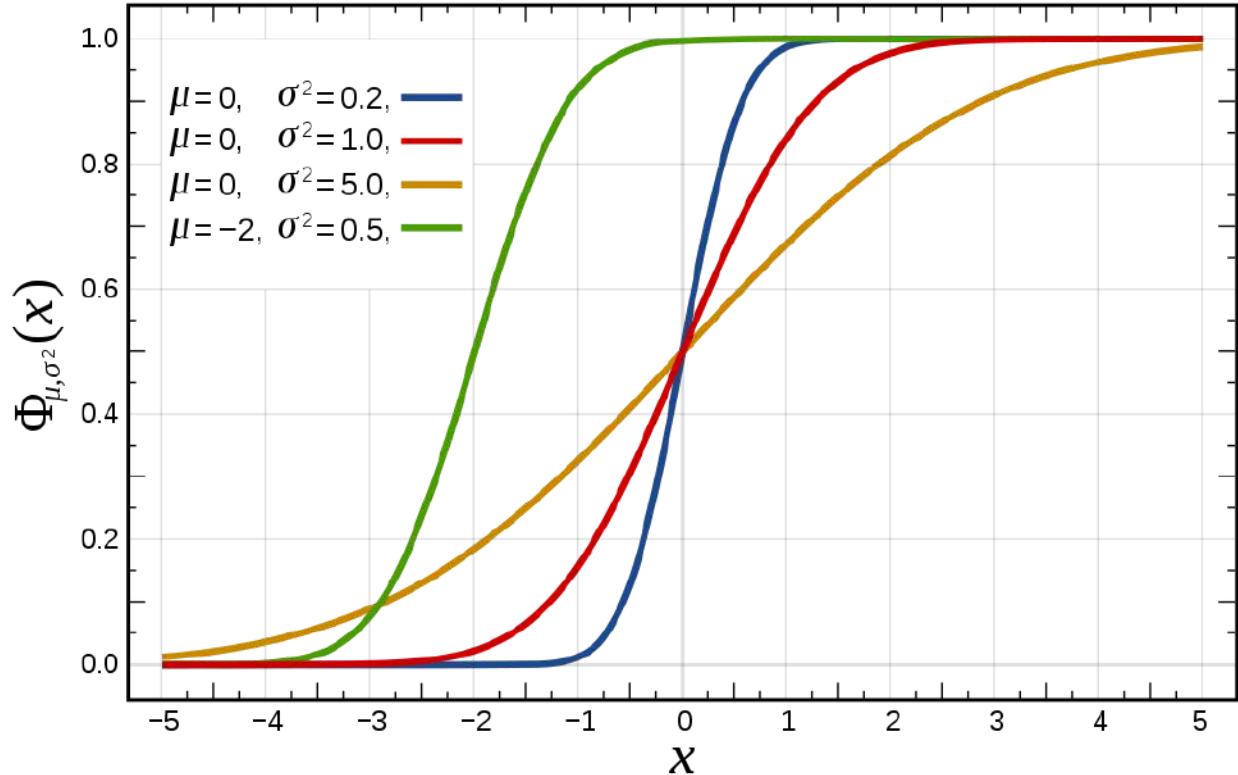
Let us look at the plots of the PDFs and the CDFs of Normal distribution with different mean and variance values, discussed at the timestamp 0.30



Here we can see the PDFs of the gaussian distribution with different mean and variance values. The peak of the curve always lies at the mean and the shape of the curve is symmetric around the mean. Here 50% of the values lie to the left side of the mean and the remaining 50% lie on the right side of the mean.

The curve becomes wider and the peak falls down, if the variance is large. The curve becomes narrower and the peak goes high, if the variance is small.

If the mean increases, then the curve moves towards the right and if the mean decreases, then the curve moves towards the left.



In the above plot, we can see the plot of CDFs of the normal distribution with different mean and the variances. The CDF value for a symmetric distribution at the mean is always equal to 0.5. The alignment of the CDF plot on the axis depends on the position of the mean. If the mean increases, then the CDF curve moves towards the right and if the mean decreases, then the CDF curve moves towards the left. But the value of the CDF always lies in between 0 and 1, irrespective of the position of the mean.

The width of the CDF plot is dependent on the variance. If the variance increases, then the curve moves away from the vertical axis of the mean. If the variance decreases, then the curve moves towards the vertical axis of the mean.

Standard Deviation is a measure that is used to quantify the amount of variation or dispersion of the given set of values. A low standard deviation indicates that the data points tend to be close to the mean. If the standard deviation is high, then it indicates that the data points are spread out over a wide range of values.

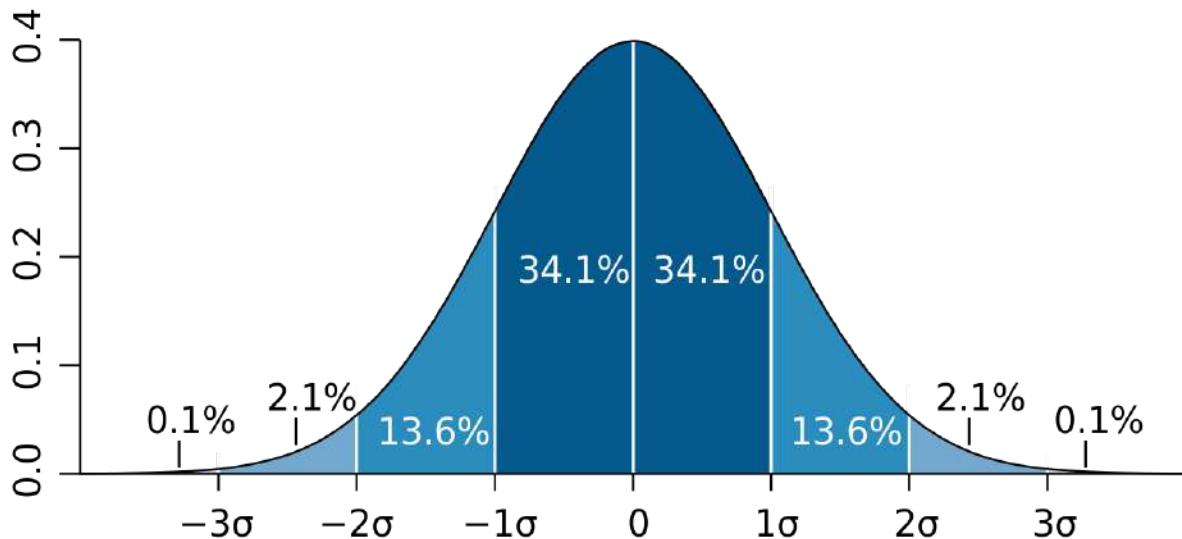
Properties of CDF

- The CDF function should be non decreasing.
- The CDF function has to be right continuous.
- As the value of 'x' keeps tending to negative infinity, the value of CDF should tend to 0.
- As the value of 'x' keeps tending to positive infinity, the value of CDF should tend to 1.
- CDF is never symmetrical in nature.
- CDF is obtained by cumulatively adding the probabilities and as the probabilities can never be negative, the CDF curve never goes down.
- The CDF curve starts at 0 and ends at 1 and usually has 'S' shape.

68-95-99.7 Rule

- The 68-95-99.7 rule or the empirical rule is used to remember the percentage of the values that lie in an interval estimate for a normal distribution.
- 68%, 95% and 99.7% of the values lie in the intervals of first, second and the third standard deviations respectively on both sides.

Below is the representation of the 68-95-99.7 rule, discussed at the timestamp 4:20.



For example, we have a normal distribution with a mean of 150 and a standard deviation of 25, then

$$1\sigma = 25; 2\sigma = 50, 3\sigma = 75$$

68% of the values lie in the interval [150-25, 150+25] (ie., [125,175])

95% of the values lie in the interval [150-50, 150+50] (ie., [100,200])

99.7% of the values lie in the interval [150-75, 150+75] (ie., [75,125])

Note: It is always not mandatory for the mean of a normal distribution to be 0. It can also be a non zero value. But once if the values are standardized, then the mean becomes 0 and the standard deviation becomes 1. The concept of Standardization has been discussed in the next set of lectures in this chapter itself.

This 68-95-99.7 rule is applicable only for the normal distribution.

22.2 Symmetric Distribution, Skewness and Kurtosis

Symmetric Distribution

A probability distribution is said to be symmetric, only if there exists a value x_0 , such that $f(x_0+\delta) = f(x_0-\delta)$ for all real numbers of ' δ ' and 'f' is the PDF. It means if we choose any value ' δ ', then the PDF value at $X=x_0+\delta$ and $X=x_0-\delta$ should be the same.

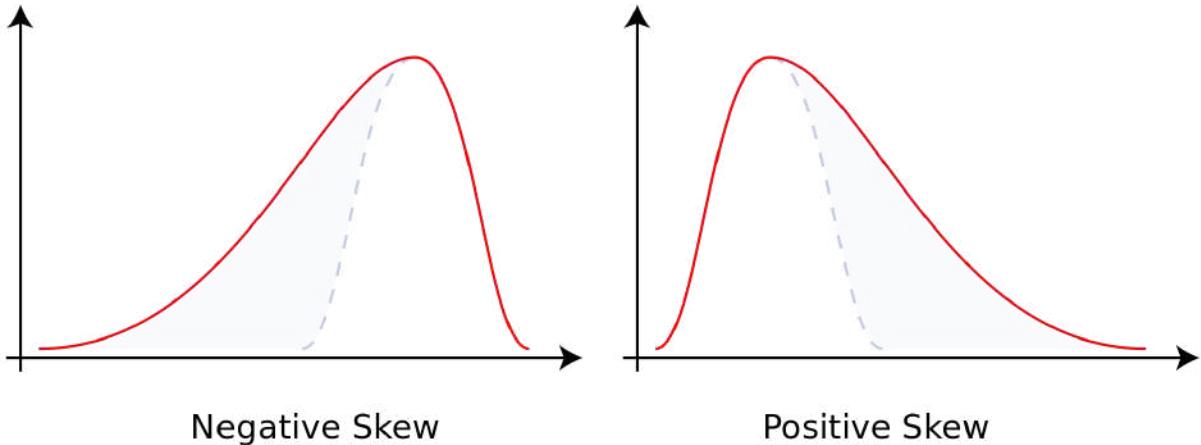
Here if 'f' is symmetric, then the portion of the distribution/curve present to the left side of the point ' x_0 ' is the mirror image of the portion of the distribution/curve present to the right side of the point ' x_0 '.

In case, if there doesn't exist any point ' x_0 ', such that $f(x_0+\delta) = f(x_0-\delta)$, then such a distribution is said to be non-symmetric.

Skewness

- Skewness is a measure of asymmetry of the probability distribution of a real valued random variable, around its mean. It can have either a positive value or a negative value or zero.
- For a symmetric distribution, the skewness value is 0, whereas for a non symmetric distribution, the skewness value could either be negative or positive. For a symmetric distribution, the density of the points on the left side region (ie., before the point ' x_0 ') is same as the density of the points on the right side region (ie., after the point ' x_0 ').
- If a distribution has a longer tail to the left side, then it is called Negatively Skewed Distribution (or) Left Skewed Distribution. The value of skewness for such a distribution is always negative. Here the density of the points on the PDF is more towards the right when compared to the left.
- If a distribution has a longer tail to the right side, then it is called Positively Skewed Distribution (or) Right Skewed Distribution. The value of skewness for such a distribution is always positive. Here the density of the points on the PDF is more towards the left when compared to the right.

Below is the figure that represents the positive and the negative skewed distributions and it was discussed at the timestamp 7:00



In this figure, we could see that in Negative Skewed Distribution, the tail is present to the left side which means the density of the points is lower on the left side and higher on the right side. Similarly, in Positive Skewed Distribution, the tail is present to the right side which means the density of the points is lower on the right side and higher on the left side.

The more negative the skewness measure value is, the more left/negative skewed the distribution would be and the tail would be longer on the left side. The more positive the skewness measure value is, the more right/positive skewed the distribution would be and the tail would be longer on the right side.

The formula for computing the sample skewness was discussed at the timestamp 10:15 and is given below

$$b_1 = \frac{m_3}{s^3} = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^3}{\left[\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \right]^{3/2}}$$

Here ' \bar{x} ' represents the sample mean and 'n' denotes the sample size.

Note: The skewness of a normal distribution is 0.

Advantages of Skewness

- Skewness is used along with the histogram and the QQ plot to characterize the distribution of the data.
- The magnitude(ie., the absolute value) of the skewness indicates how far is the distribution from symmetry. The sign(ie., positive or negative) of skewness indicates the direction in which the distribution is skewed.

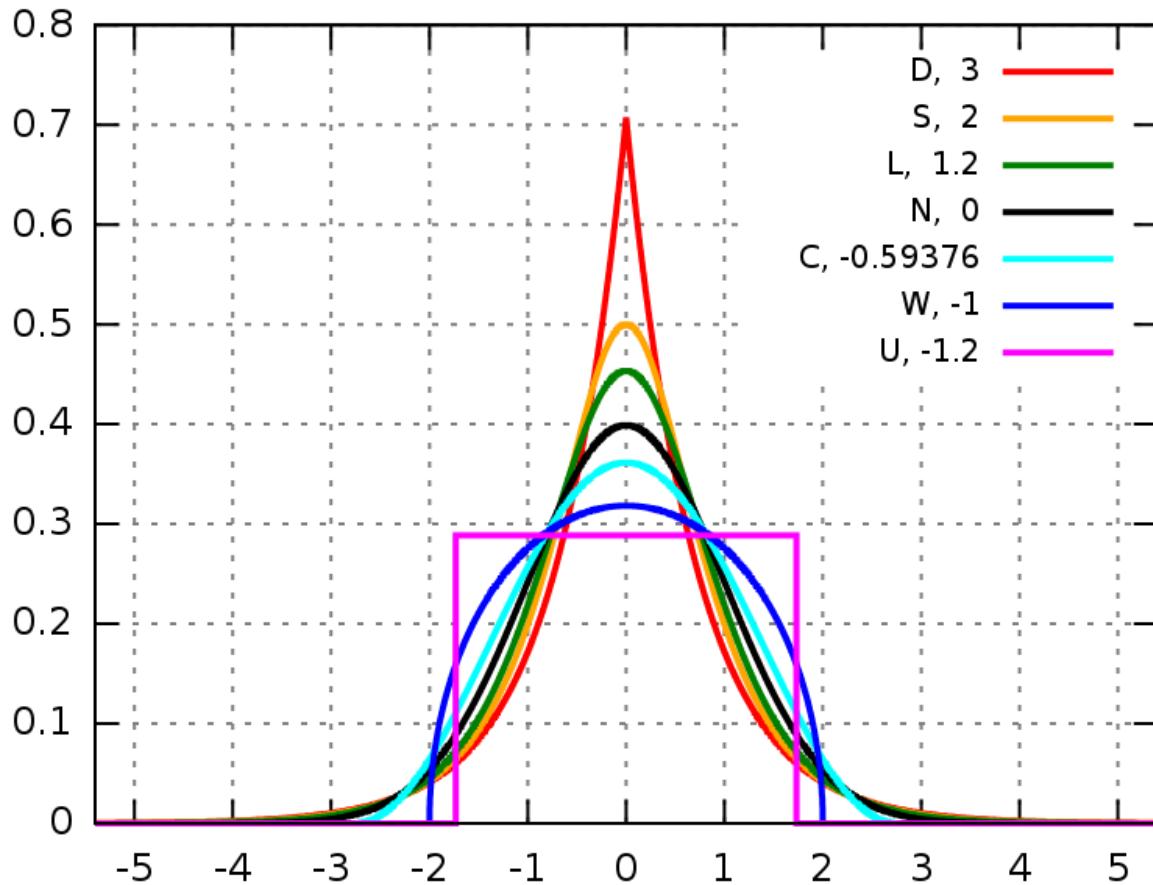
Kurtosis

- Kurtosis is a measure of tailedness of the probability distribution of a real valued random variable.
- Skewness only indicates whether the given distribution is left skewed or right skewed whereas Kurtosis gives additional information about the shape of the distribution.
- Distributions with large Kurtosis have the tails exceeding the tails of normal distribution.
- Distributions with small Kurtosis have the tails smaller when compared to the tails of normal distribution.
- There are different ways of quantifying the kurtosis for a theoretical distribution and corresponding ways of estimating it from a sample or population. Different measures of kurtosis may have different interpretations.

Below is the formula for computing the excess Kurtosis metric and it was discussed at the timestamp 15:20

$$g_2 = \frac{m_4}{m_2^2} - 3 = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^4}{\left[\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \right]^2} - 3$$

In the above formula, the first term denotes the kurtosis value and the excess kurtosis is given as (**excess kurtosis = kurtosis-3**). Below is the graph of how the PDFs look like with different values of the 'excess kurtosis' metric and it has been discussed at the timestamp 17:20.



In this graph of PDFs, we can see the large values of ‘excess kurtosis’ lead to longer tails of PDF on both the sides whereas smaller values of ‘excess kurtosis’ leads to quickly falling tails in the PDF. The ‘excess kurtosis’ value for gaussian distribution is always 0.

The ‘excess kurtosis’ indicates how different is the shape of the distribution when compared to a gaussian distribution with a kurtosis value of 3.

Applications of Kurtosis

One of the major applications of Kurtosis is in finding the outliers from the given set of values.

Note: Symmetry, Skewness and Kurtosis are three parameters used in understanding the shape of the PDF of a distribution.

22.3 Standard Normal Variate (Z) and Standardization

If we have a random variable 'Z' with a mean of 0 and standard deviation of 1 and following normal distribution, the 'Z' is called a Standard Normal Variate.

It is represented as $Z \sim N(0,1)$

Let us assume we have a random variable 'X' which follows normal distribution and has a mean of ' μ ' and variance of ' σ^2 ', then it is represented as $X \sim N(\mu, \sigma^2)$.

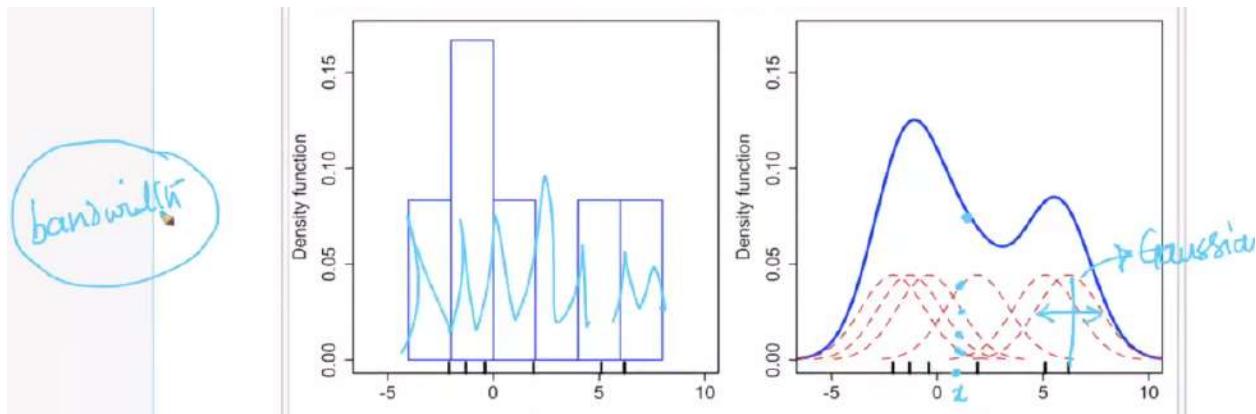
Standardization is the process of transforming a given distribution with a mean ' μ ' and variance of ' σ^2 ', into the same type of distribution with a mean of 0 and standard deviation of 1. (Even the variance also would be 1)

Let the values of 'X' be $[x_1, x_2, x_3, \dots, x_n]$, then after applying standardization $x'_i = (x_i - \mu)/\sigma$ and the transformed distribution is denoted as $X' \sim N(0,1)$

Note: Standardization just transforms the given distribution of values onto a new scale with mean = 0 and standard deviation = 1. The nature of the distribution doesn't change at all.(irrespective of whether the distribution is gaussian or non gaussian)

22.4 Kernel Density Estimation (KDE)

So far we have seen the PDF and histograms in the univariate analysis. The PDF is obtained by drawing a smooth curve on the histogram using a technique called Kernel Density Estimation(KDE). Let us look at the plots that were discussed at the timestamp 0:35.

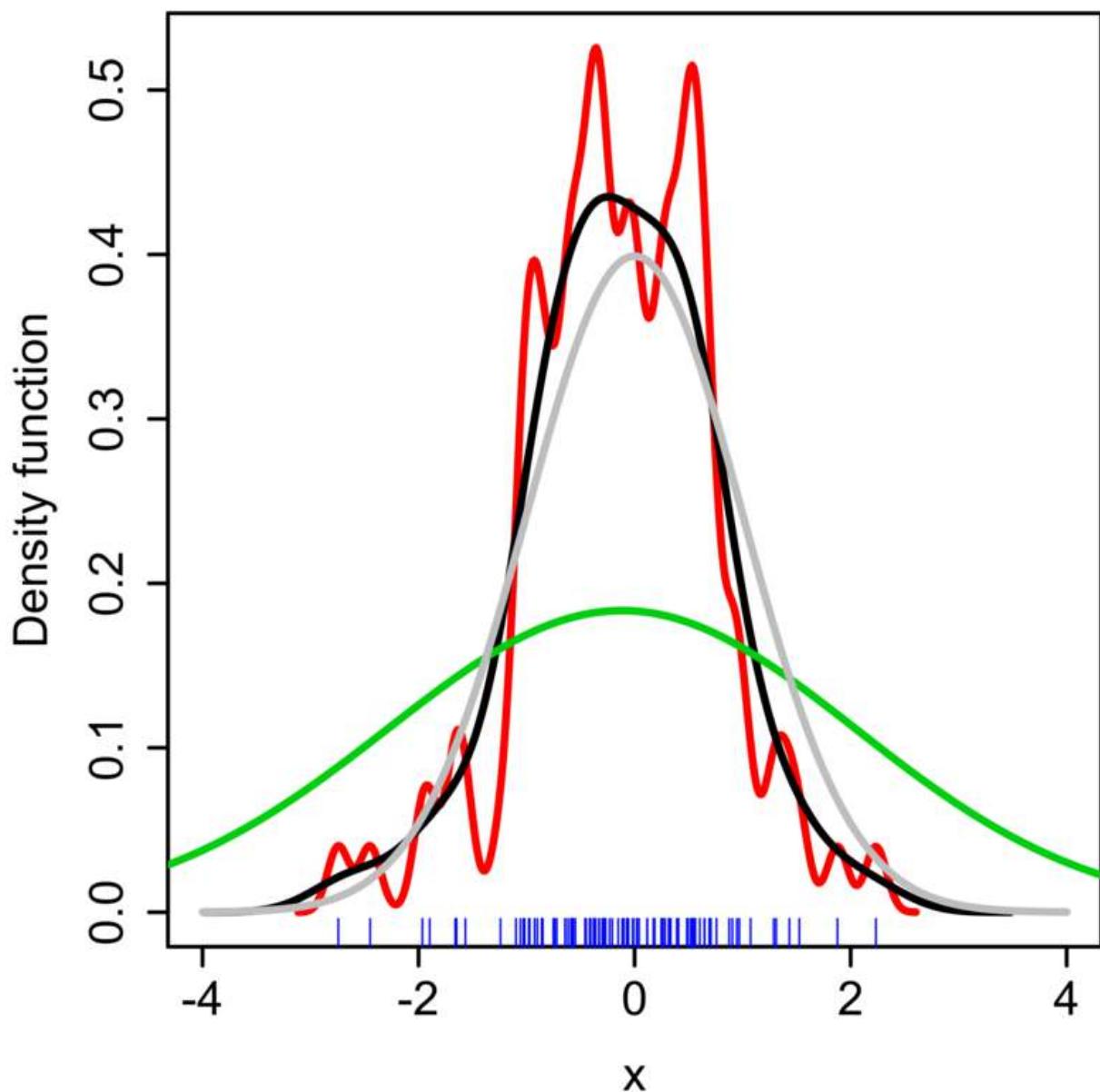


In the right side plot, the number of gaussian kernels drawn is equal to the number of points in the dataset. The kernels used here are the gaussian kernels. Here these gaussian kernels are plotted such that the mean of each of these kernels are the points in the dataset(ie., with each data point as the centre, we plot the gaussian kernels)

For every point on the 'X' axis, we look for how many kernels are passing through that point. The 'Y' axis coordinates of all these kernels are added to get the PDF value at that point.

The bandwidth of these kernels is nothing but the variance of these kernels. If the bandwidth is high, then the kernels will become much wider and thereby the resulting PDFs will be wider. Similarly if the bandwidth is low, then the kernels will become much narrower and the resulting PDF will look jagged. Hence the bandwidth of these kernels should be chosen properly. The standard deviation of these kernels is equal to the bin width. The height of a gaussian kernel should be such that the area under it is equal to 1.

The smoothness of the PDF depends on the bandwidth chosen. So the value of bandwidth has to be chosen properly. Changing the bandwidth changes the shape of the kernel. If the bandwidth is low, only the points that are very nearer are taken into consideration for computing the PDF, and the curve looks more squiggly. If the bandwidth is high, then more points are taken into consideration for computing the PDF at a point, and the PDF curve looks shallow. You can find it out in the below PDF plot.



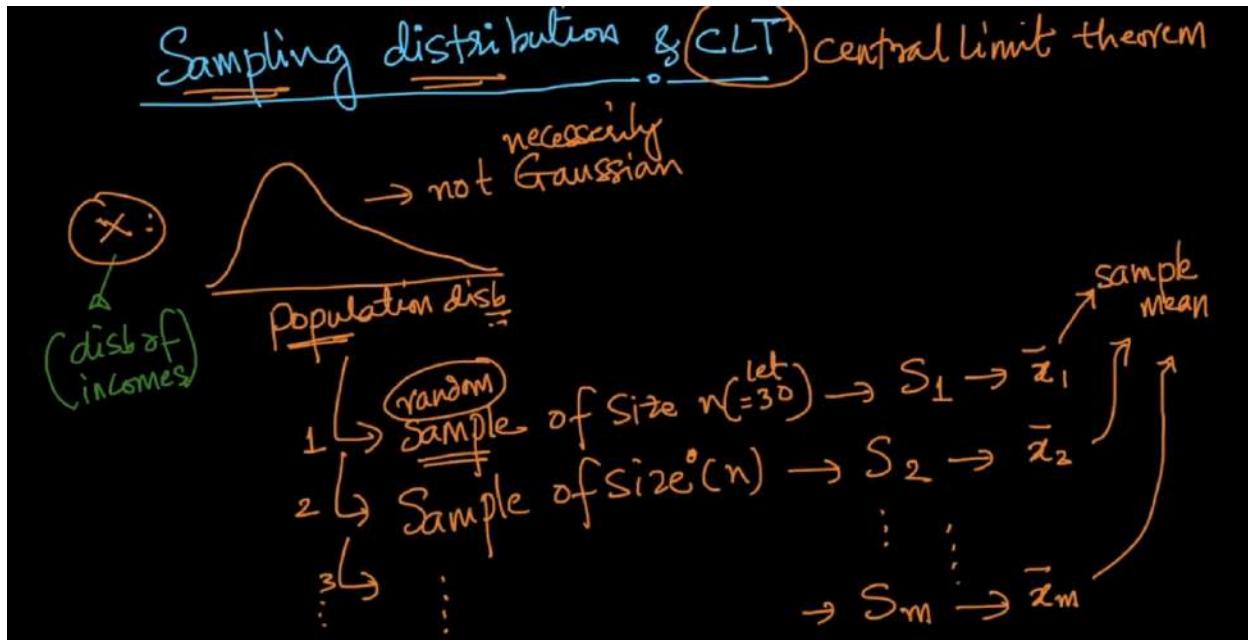
In the above plot, if the bandwidth is high, then the PDF looks like the green curve. If the bandwidth is moderate, then the PDF looks like the black curve. If the bandwidth is low, then the PDF looks like the red curve.

If the density of the points is high, then the height of the PDF increases in that region. If the density of the points is low, then the height of the PDF decreases.

The curves with more variance have more bandwidth and the curves with less variance have lesser bandwidth.

22.5 Sampling Distribution and Central Limit Theorem (CLT)

Sampling Distribution



Let us assume a distribution for the given population. It is not mandatory for this distribution to be gaussian. Let us pick 'm' random samples(which are subsets of the population) from the population and let the size of each of these samples be 'n'. All these samples are independent of each other. Let us calculate the mean of each sample and denote them as

$\bar{x}_1 \rightarrow 1^{\text{st}}$ sample mean

$\bar{x}_2 \rightarrow 2^{\text{nd}}$ sample mean

.

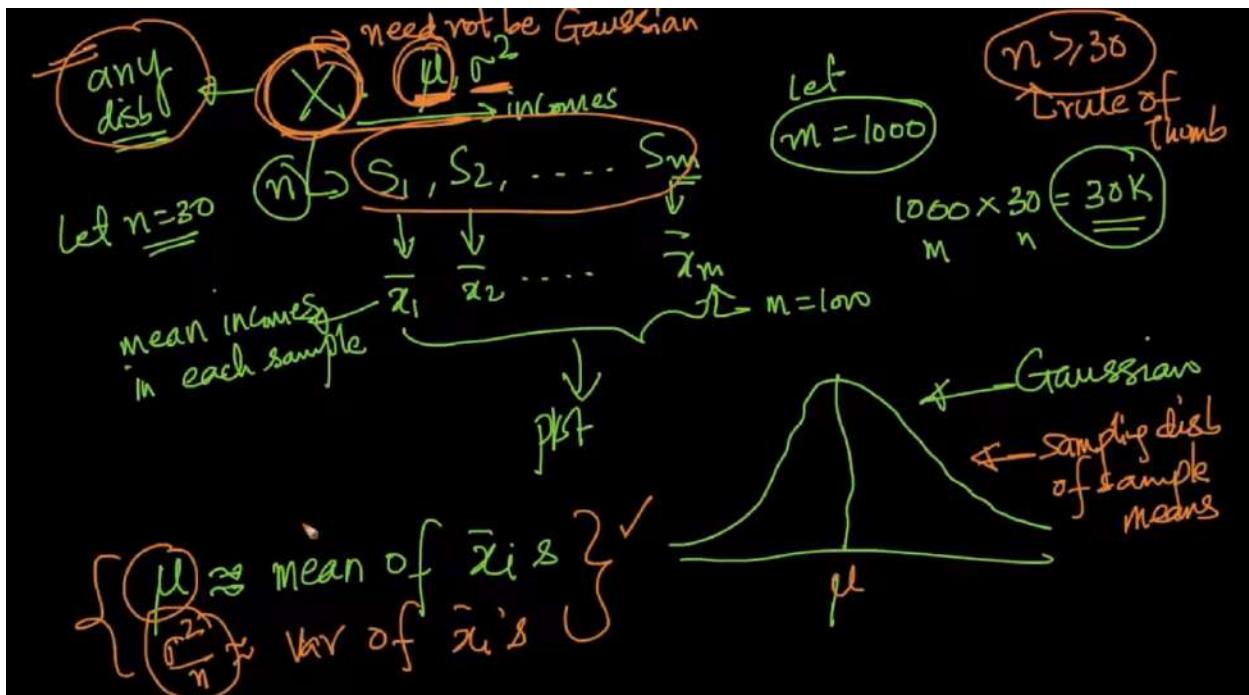
.

$\bar{x}_m \rightarrow m^{\text{th}}$ sample mean

All these samples should be the same size. All the statistical measures of these samples have a probability distribution and it is called **Sampling Distribution**. In this context, as this sampling distribution is obtained from the sample means, we call it **Sampling Distribution of Sample Means**.

We also can have the sampling distributions of sample medians or sample variances as per our problem requirement. Sampling Distributions are very important in statistics as they provide a simplification route to statistical inference.

Central Limit Theorem



Let us assume a random variable 'X' and let it's population distribution have a finite mean and a finite variance. Let us pick 'm' different samples of size 'n' each (ie., $S_1, S_2, S_3, \dots, S_m$) from the population. Let us denote the mean of i^{th} sample as \bar{x}_i . So $\bar{x}_1, \bar{x}_2, \bar{x}_3, \dots, \bar{x}_m$ be the respective means of the samples. The distribution of \bar{x}_i is the sampling distribution of sample means. Now the Central Limit Theorem states that

For any distribution of the random variable 'X' with a finite mean ' μ ' and a finite variance ' σ^2 ', the sampling distribution of the sample means is a gaussian distribution with a mean nearly equal to the population mean ' μ ' and variance equal to ' σ^2/n '.

CLT works always only for the populations with a finite mean and a finite variance. The population distribution could either be gaussian or not, but sample size should be large, all the samples should be of the same size, and all the samples are picked with replacement. Pareto Distribution is an example of a distribution with an infinite mean and an infinite variance. So CLT doesn't work if the population distribution is pareto.

The 68-95-99.7 rule applies for the sampling distribution of the sample means. If the population distribution of 'X' is gaussian, then the 68-95-99.7 rule applies for both the sampling distribution of the sample means and also the population distribution of 'X', whereas if population distribution of 'X' is not gaussian, then the 68-95-99.7 rule applies only for the sampling distribution of the sample means, but not for the population distribution.

22.6 Q-Q plot: How to test if a random variable is normally distributed or not?

Quantile-Quantile (QQ) plot is a probability plot which is a graphical method for comparing two probability distributions by plotting their quantiles against each other.

Procedure for QQ plots

Let ' X ' = $[x_1, x_2, x_3, \dots, x_n]$ be the given random variable for which the probability distribution is unknown.

Let us create a random variable $Y \sim N(0,1)$. So ' Y ' = $[y_1, y_2, y_3, \dots, y_n]$.

Step 1

Sort all the x_i 's in ascending order and compute the percentiles of ' X '.

Notation

$$\begin{aligned} x_i &\rightarrow i^{\text{th}} \text{ point/value of } 'X' \\ x^{(i)} &\rightarrow i^{\text{th}} \text{ percentile of } 'X' \end{aligned}$$

Step 2

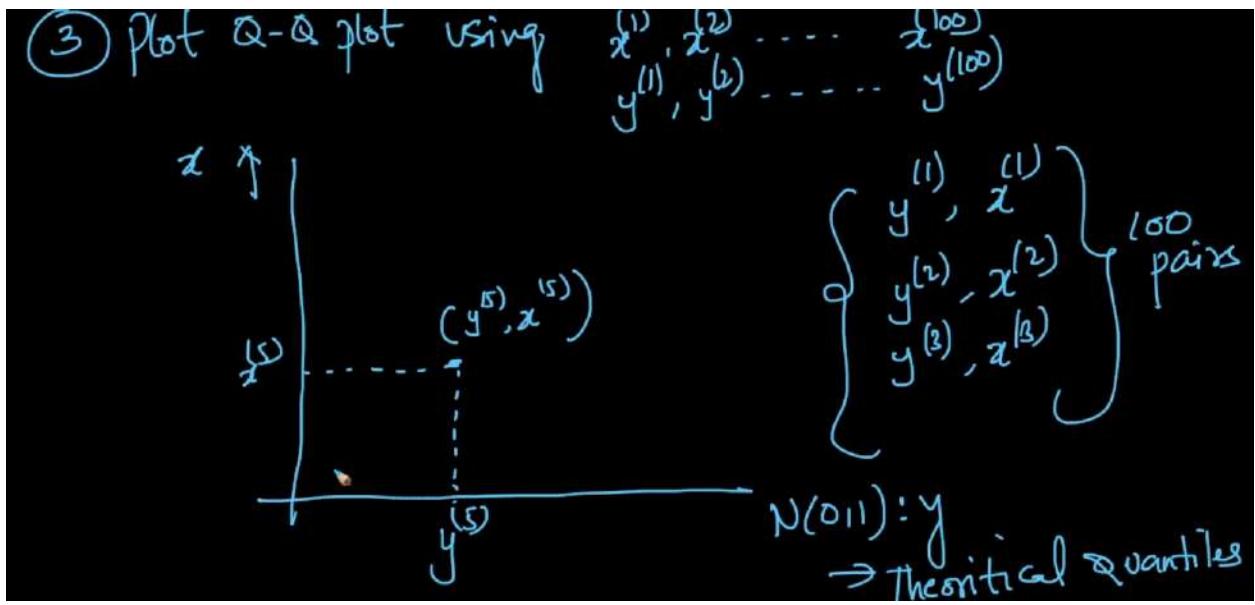
Sort all the y_i 's in ascending order and compute the percentiles of ' Y '.

Step 3

Pick the 100 percentiles of both ' X ' and ' Y ' and form them into pairs as $(y^{(i)}, x^{(i)})$ and plot these points.

After plotting these points, if all these points are on a straight line, then we can say both ' X ' and ' Y ' follow the same distribution.

Below is the QQ plot that was discussed starting from the timestamp 6:05



Below is the code snippet for building the QQ plot that was discussed starting from the timestamp 11:40.

Q-Q Plot

```
#Q-Q plot
import numpy as np
import pylab
import scipy.stats as stats

# N(0,1)
std_normal = np.random.normal(loc = 0, scale = 1, size=1000)

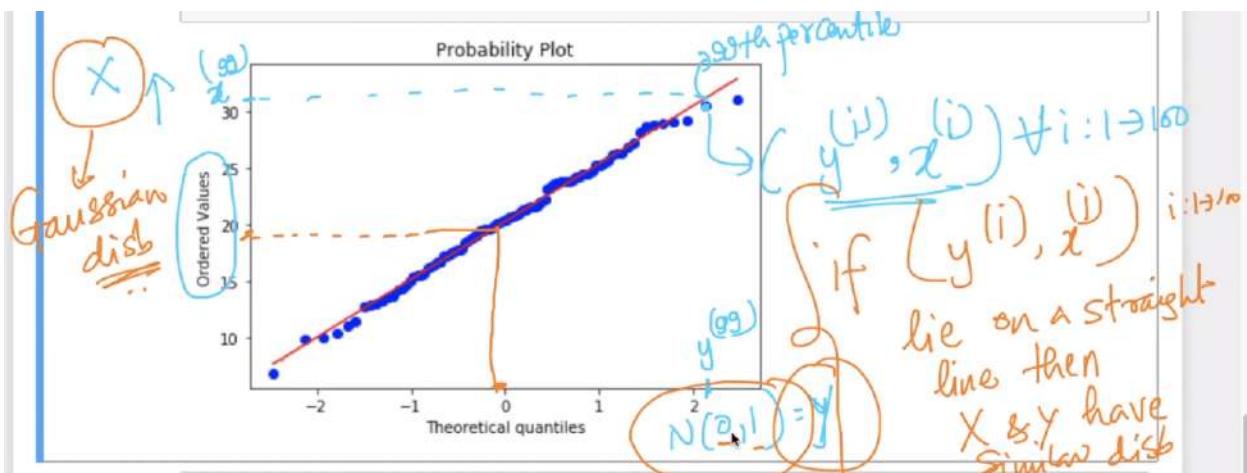
# 0 to 100th percentiles of std-normal
for i in range(0,101):
    print(i, np.percentile(std_normal,i))
```

```
# N(0,1)
std_normal = np.random.normal(loc = 0, scale = 1, size=1000)

# 0 to 100th percentiles of std-normal
for i in range(0,101):
    print(i, np.percentile(std_normal,i))
```

```
# generate 100 samples from N(20,5)
measurements = np.random.normal(loc = 20, scale = 5, size=100)
#try size=1000

stats.probplot(measurements, dist="norm", plot=pylab)
pylab.show()
```



In a QQ plot, if all the points $(y^{(i)}, x^{(i)})$ lie along a straight line, then the distribution of 'X' is the same as the distribution of 'Y'.

i.e., If 'Y' is gaussian, then 'X' also follows Gaussian distribution. If 'Y' is pareto, the 'X' also follows Pareto distribution.

If we increase the number of points in both the sets of the percentiles of 'X' and 'Y', then the points on the QQ plot move closer to each other and the plot becomes more straight. If the number of points in 'X' and 'Y' is less, then the plot looks like a curve, instead of a straight line, even if both 'X' and 'Y' belong to the same distribution.

If 'X' and 'Y' are of different distributions, then the QQ plot is nonlinear. In such cases, even if you increase the number of points in both 'X' and 'Y', the plot becomes more non linear, as they both are from two different distributions.

When we have two random variables 'X' and 'Y' and if the type of distribution of at least one of them is known, then we can check whether they both belong to the same distribution or not, by building the QQ plot.

In case, if we are not known of the distribution of any one of those two random variables, then we should create a random normal or a random uniform distribution 'Z' and build the QQ plots between 'X & Z' and 'Y and Z' combinations and check whether they are the same.

The more the values are taken in 'X' and 'Y', the confidence of describing a particular distribution is higher. But due to the presence of more number of points(i.e., percentiles), the plot looks dense, as if the points are overlapped. But the QQ plots become much more straight.

22.7 Chebyshev's Inequality

Let us assume, we are given a random variable 'X' which follows gaussian distribution. As we know that 68-95-99.7 rule applies for gaussian distribution,

As $X \sim N(\mu, \sigma)$,

68% of the values of 'X' lie in $[\mu-\sigma, \mu+\sigma]$ $\Rightarrow P(\mu-\sigma \leq X \leq \mu+\sigma) = 68\%$

95% of the values of 'X' lie in $[\mu-2\sigma, \mu+2\sigma]$ $\Rightarrow P(\mu-2\sigma \leq X \leq \mu+2\sigma) = 95\%$

99.7% of the values of 'X' lie in $[\mu-3\sigma, \mu+3\sigma]$ $\Rightarrow P(\mu-3\sigma \leq X \leq \mu+3\sigma) = 99.7\%$

If we know that 'X' follows gaussian distribution and has a finite mean and a finite variance, then we can make the above estimations.

But what if we have a finite mean ' μ ' and a finite variance ' σ^2 ' and if we are not aware of the distribution of 'X', the Chebyshev's Inequality states that

$$P(|X-\mu| \geq K\sigma) \leq (1/K^2)$$

$(X-\mu) \geq K\sigma$ means the range $X \geq (\mu+K\sigma)$ and $X \leq (\mu-K\sigma)$

$$P(X \geq (\mu+K\sigma) \text{ and } X \leq (\mu-K\sigma)) \leq (1/K^2)$$

$$\text{Now } P((\mu-K\sigma) \leq X \leq (\mu+K\sigma)) \geq (1-(1/K^2))$$

Example

Let us assume we have a random variable 'X' that denotes the salaries. Let's say the given mean(μ) is 40K and the standard deviation(σ) is 10K.

1) If 'X' follows gaussian distribution

According to the 68-95-99.7 rule,

68% of the salaries lie in the interval $= [\mu-\sigma, \mu+\sigma] = [40K-10K, 40K+10K] = [30K, 50K]$

95% of the salaries lie in the interval $= [\mu-2\sigma, \mu+2\sigma] = [40K-20K, 40K+20K] = [20K, 60K]$

99.7% of the salaries lie in the interval $= [\mu-3\sigma, \mu+3\sigma] = [40K-30K, 40K+30K] = [10K, 70K]$

2) If we are not aware of the distribution of 'X'

Minimum Percentage of salaries that lie in the interval $[\mu-K\sigma, \mu+K\sigma] \geq (1-(1/K^2))$

Minimum Percentage of salaries that lie in the interval $[\mu-\sigma, \mu+\sigma] \geq (1-(1/1^2)) = (1-1) = 0$

(It means minimum 0% of the salaries lie in the interval $[\mu-\sigma, \mu+\sigma]$)

Minimum Percentage of salaries that lie in the interval $[\mu-2\sigma, \mu+2\sigma] \geq (1-(1/2^2)) = (1-(1/4)) = (3/4) = 0.75$

(It means minimum 75% of the salaries lie in the interval $[\mu-2\sigma, \mu+2\sigma]$)

Minimum Percentage of salaries that lie in the interval $[\mu-3\sigma, \mu+3\sigma] \geq (1-(1/3^2)) = (1-(1/9)) = (8/9) = 0.89$

(It means minimum 89% of the salaries lie in the interval $[\mu-3\sigma, \mu+3\sigma]$)

Note:

Chebyshev's Inequality is weaker when compared to the 68-95-99.7 rule which is applied to normal distribution. The 68-95-99.7 rule gives the exact values of the percentage of points lying within a certain number of standard deviations from the mean, whereas Chebyshev's Inequality gives the minimum percentage of the points. The number of deviations 'K' accepts only integer values.

Chebyshev's inequality applies only when the given distribution has a finite mean and a finite variance.

Note: As the video lecture 22.8 is just a doubt session, we are not adding it here. All the information about Chebyshev's inequality is added in the noted for 22.7.

22.9 Discrete and Continuous Uniform Distributions

Uniform Distribution

Uniform distribution is the distribution in which the probability of occurrence of every value(either in a domain or in an interval) is the same.

Types of Uniform Distribution

There are two types of uniform distributions based on the type of random variable used in the distribution. They are

- a) Discrete Uniform Distribution
- b) Continuous Uniform Distribution

Discrete Uniform Distribution

- It is a symmetric probability distribution where in, a finite number of values are equally likely to be observed. Everyone of the 'n' possible outcomes has equal probability ($1/n$)
- A simple example of discrete uniform distribution is throwing a fair die. The possible values are 1,2,3,4,5,6 and each time when the die is thrown, the probability of a given score is $1/6$.
- If two dice are throws and their values are added, then the resulting distribution is no longer uniform because not all sums have equal probability.
- The uniform distribution has 2 parameters and is represented as $U(a,b)$ (where $b>a$) and the number of possible outcomes is denoted as 'n' and is equal to $b-a+1$. (ie., $n=b-a+1$)
- In a uniform distribution, all the values are equi-probable. If there are 'n' possible outcomes, then the probability of occurrence of each value is $1/n$. The possible outcomes of discrete uniform distribution are from a finite set.

Properties of Discrete Uniform Distribution

Notation: $U(a,b)$

Parameters: a,b (where $b>=a$)

PMF: $1/n$

Number of Outcomes(n): $b-a+1$

Mean and Median: $(a+b)/2$

Mode: N/A

Variance: $((b-a+1)^2-1)/12$

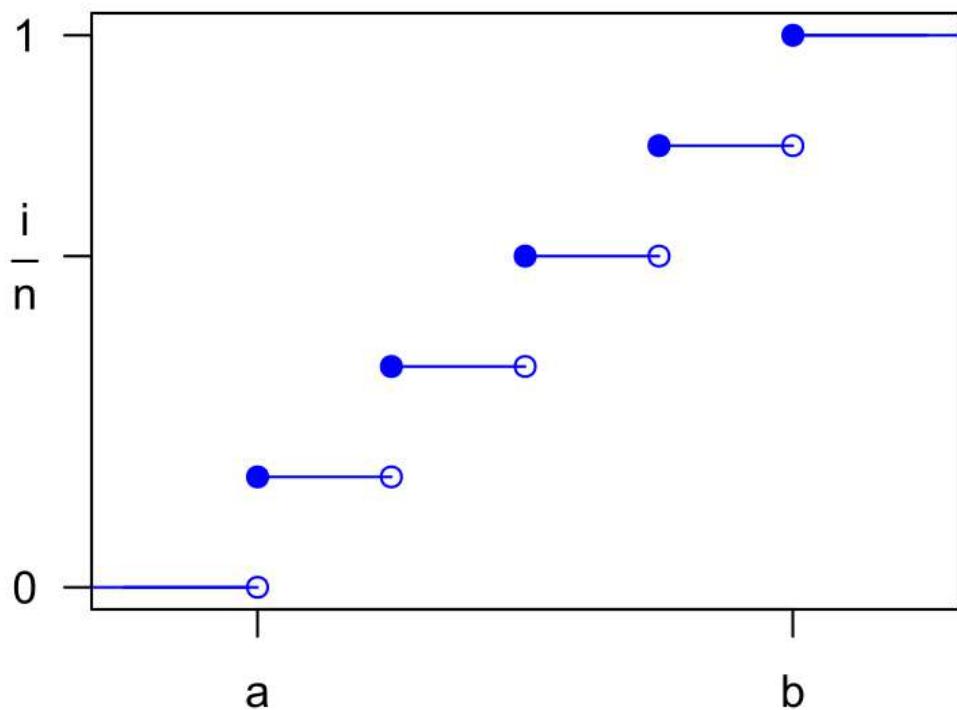
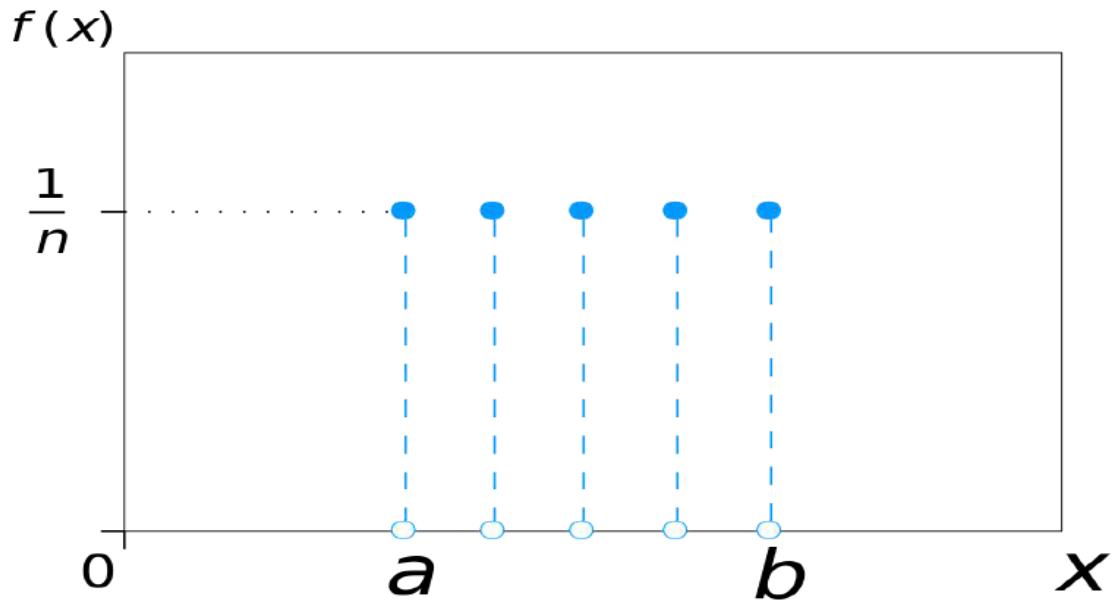
Skewness: 0

Excess Kurtosis: $-(6*(n^2+1))/(5*(n^2-1))$

CDF(at X=k): $(k-a+1)/n$

PMF and CDF plots of Discrete Uniform Distribution

Below are the plots of PMF and CDF of a Discrete Uniform distribution that were discussed starting from the timestamp 1:10



Continuous Uniform Distribution

- If a random variable is continuous and follows uniform distribution, then that distribution is called Continuous Uniform Distribution.
- Let us assume the minimum value of the distribution is 'a' and the maximum value of the distribution is 'b', the probability of occurrence of any value in the interval $[a,b]$ is the same. All the possible outcomes are equally probable.

Properties of Continuous Uniform Distribution

Mean and Median: $(a+b)/2$

Mode: Any value in the interval (a,b)

Variance: $(b-a)^2/12$

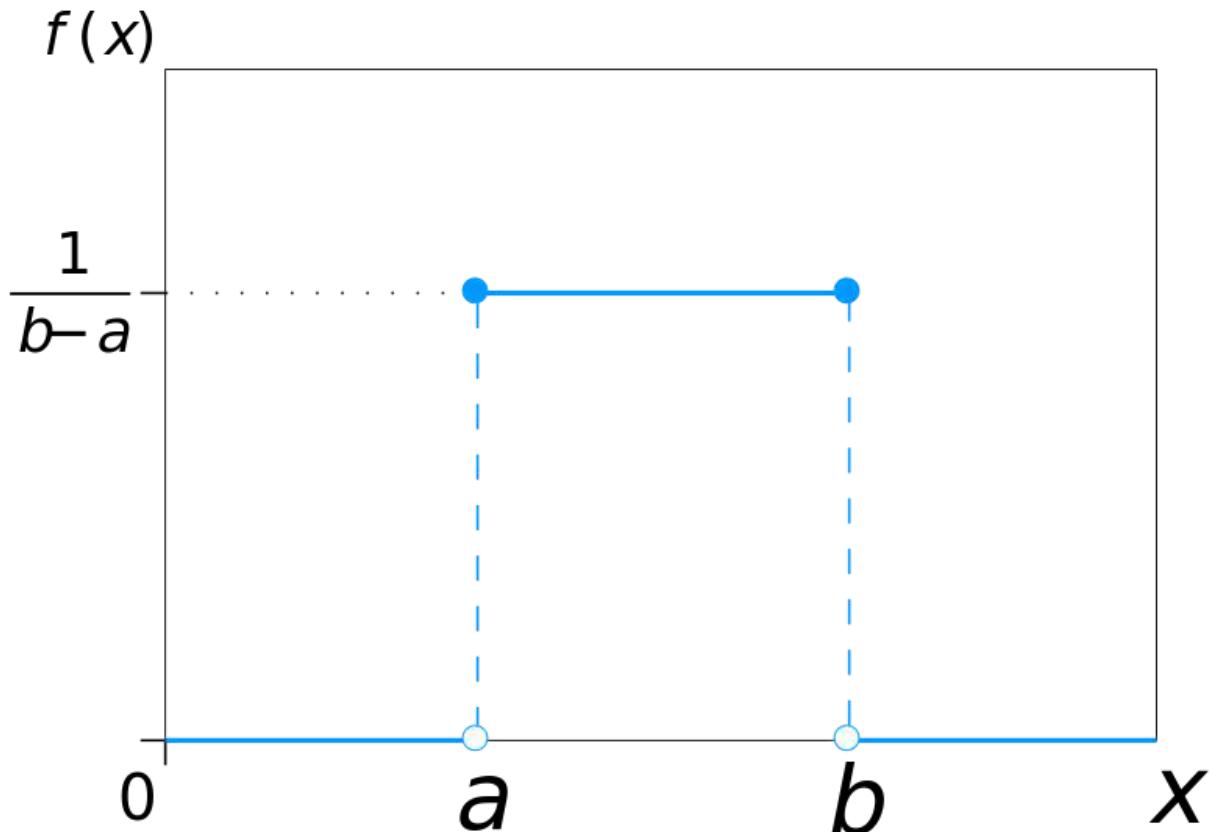
Skewness: 0

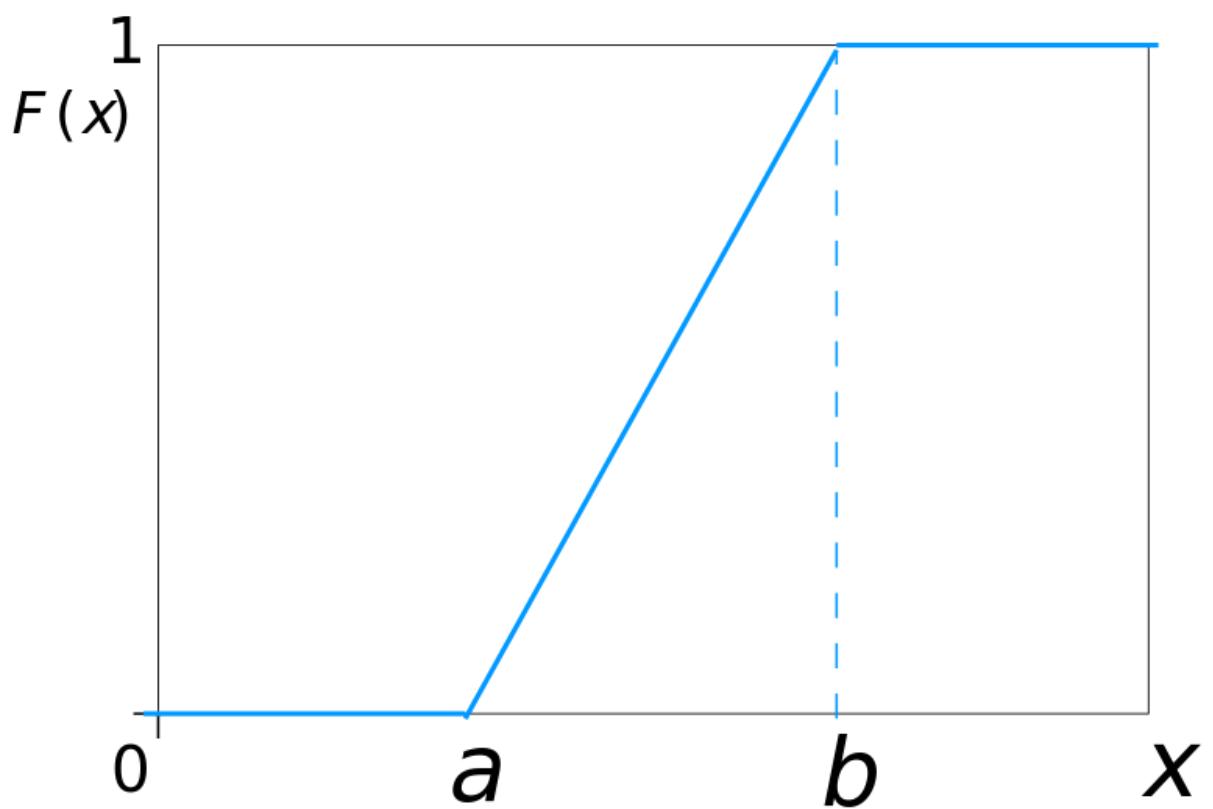
Excess Kurtosis: -6/5

PDF(at $X=x$): $1/(b-a)$ if $x \in (a,b)$. Otherwise 0.

CDF(at $X=x$): 0 if $x < a$. $(x-a)/(b-a)$ if $x \in [a,b]$. 1 if $x > b$

Plots of PDF and CDF of a Continuous Uniform Distribution





22.10 How to randomly sample data points (Uniform distribution)

Using Random Number Generator

In general, most of the random number generators generate the random numbers uniformly, unless specified.

```
import random
```

```
print(random.random())
```

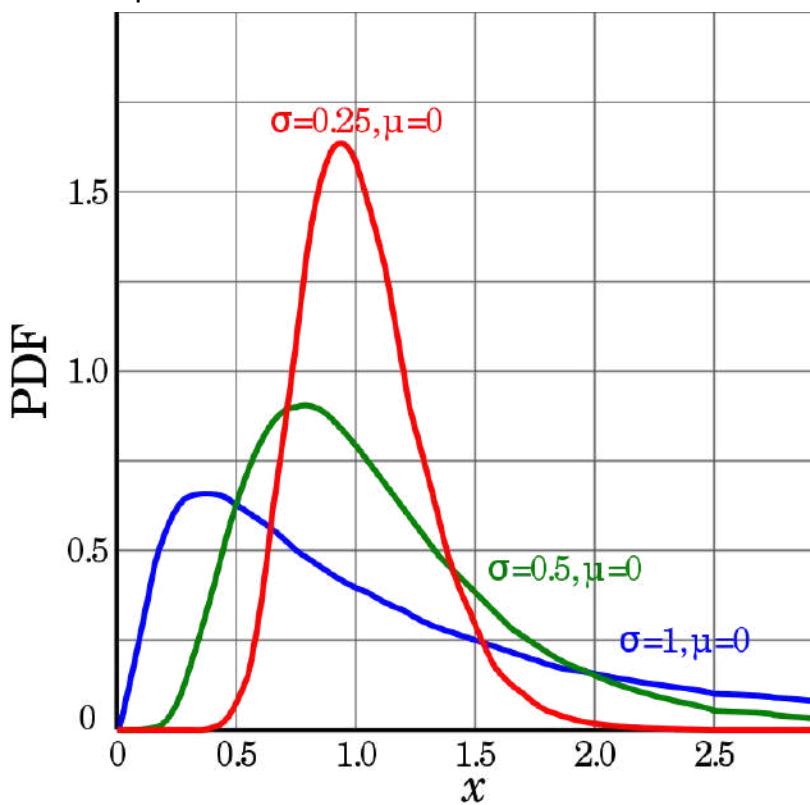
The above statement prints a random value between 0 and 1. These numbers are randomly generated with uniform distribution. Everytime we run this statement, it generates different values in the interval [0,1] with uniform distribution.

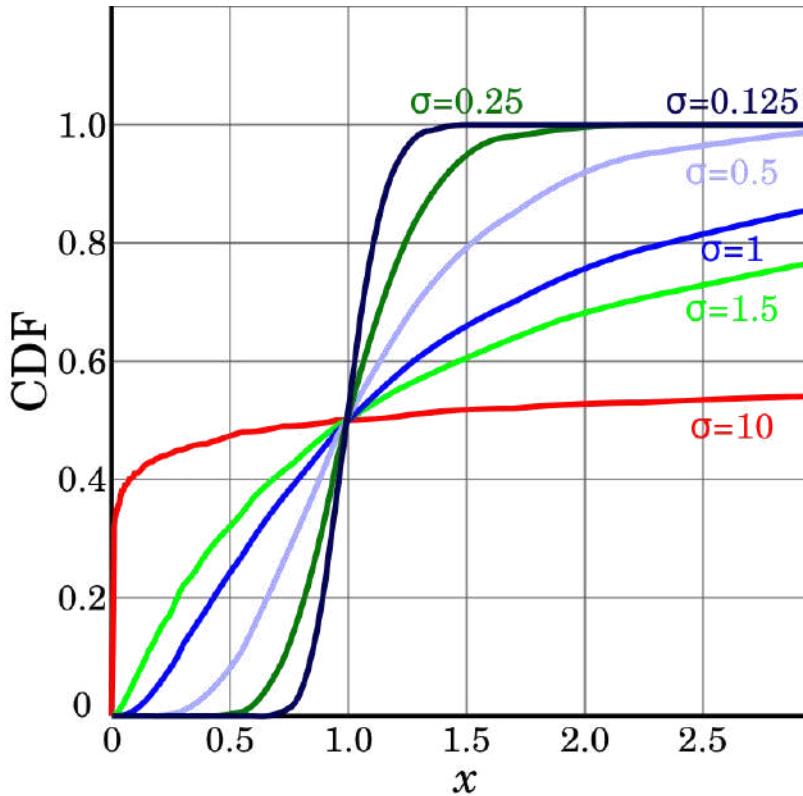
Application of Random Number Generators

Random Number Generators are used in generating continuous random numbers that are uniform in distribution. Random number generators are used in sample selection.

22.11 Log Normal Distribution

- A continuous probability distribution is said to be a log-normal, if its natural logarithm follows a normal distribution.
- If a random variable 'X' is log-normally distributed, then $Y = \ln(X)$ has normal distribution.
- Similarly, if $Y = \ln(X)$ has normal distribution, then $X = e^Y$ has log-normal distribution.
- A random variable which is log-normally distributed, takes only positive and real values. You can see the PDFs and CDFs of a few log normal distributions below with the same mean and different standard deviations, that were discussed at the timestamp 1:20.





- As the standard deviation increases, the tail factor increases and the distribution becomes skewed. (specially in log-normal distribution)
- Log Normal Distribution is always positive skewed. There is no chance of being negative skewed. Every Log Normal Distribution is a skewed distribution, but every skewed distribution is not a log normal distribution.
- The value of PDF at any point can exceed 1(as PDF is computed by adding many gaussian kernels), but the area under the curve should always be equal to 1. PDF height of more than 1, indicates the density of points in that region is heavy.
- If we have a gaussian distribution, 1σ , 2σ help us get a sense of spread of the data. If a distribution is long tailed and not gaussian, we can expect to find some large observations very far away from the mean/median. This type of information helps us get a better sense of what we can expect from our data.
- In a gaussian distribution, the peak of the PDF indicates the mean of the distribution. But in log-normal distribution, the mean is not the same as the peak of the PDF.
- For a log normal distribution, we cannot apply the 68-95-99.7 rule, as it is not a gaussian distribution. But once if we apply a natural logarithm on a log-normal distributed variable, the feature then after the transformation, follows normal distribution and then the 68-95-99.7 rule is applicable.

- In the scenarios where the log-normal distribution has its applications, we see log-normal distribution most of the time being used, whereas the gaussian distribution was less frequently used.
- If 'X' follows log normal distribution, then $Y=\ln(X)$ follows a normal distribution.
 $Y \sim N(\mu, \sigma)$
 According to normal distribution,
 68% of the data lies in $[\mu-\sigma, \mu+\sigma]$
 95% of the data lies in $[\mu-2\sigma, \mu+2\sigma]$
 99.7% of the data lies in $[\mu-3\sigma, \mu+3\sigma]$
 As $Y=\ln(X) \Rightarrow X = e^Y$
 So 68% of the data in 'X' lies in $[e^{\mu-\sigma}, e^{\mu+\sigma}]$
 So 95% of the data in 'X' lies in $[e^{\mu-2\sigma}, e^{\mu+2\sigma}]$
 So 99.7% of the data in 'X' lies in $[e^{\mu-3\sigma}, e^{\mu+3\sigma}]$
 Here μ, σ are the parameters of 'Y'.
- The values in a log normal distribution are positive and hence they are in right skewed form. In case, if we come across any negative values in the log-normal distributed values, we can convert all of them into non negative values, by adding some constant 'a', to all the values. This constant 'a' can be the largest magnitude among all the negative values.

How to check if a given distribution is log-normal

Let 'X' be the given input distribution. Let us compute the natural logarithm for the values of 'X' and let them be denoted as 'Y'. (ie., $Y = \ln(X)$)

Then we should go for the QQ plot with 'Y' values on the 'Y' axis and a randomly generated normal distribution $N(\mu, \sigma^2)$ on the 'X' axis.

If the plot looks like a straight line, then we can confirm that 'Y' is normally distributed and 'X' is log normally distributed.

Why is Log Normal Distribution having skewness

It is because of the exponential function applied on the top of the gaussian distribution. If we generate some data points from a gaussian random variable and apply exponential function on top of them, plot the resultant values, it looks like a log-normal distribution. Log Normal distribution gets more skewed if the standard deviation ' σ ' increases. Larger ' σ ' spreads the points wider in gaussian distribution.

Applications of Log Normal Distribution

- 1) The length of comments posted in internet discussion forums follow log normal distribution
- 2) The time spent by the users on the internet to read articles/blogs, etc also follows log normal distribution.

Why is the peak of a log normal distribution not considered as it's mean?

Normal/Gaussian distribution is symmetric and 50% of the values lie on one side and the remaining 50% of the values lie on the other side. So we can say the highest peak of a gaussian distribution is it's mean.

But in case of a log normal distribution, we can't guarantee that the mean is exactly present at the highest peak, as the curve is asymmetric.

Properties of Log Normal Distribution

Notation: $X \sim \text{lognormal}(\mu, \sigma^2)$

Parameters: $\mu \in (-\infty, \infty)$, $\sigma > 0$

Support (Region in which this distribution is applicable): $x \in (0, \infty)$

PDF(at $X=x$): $(1/(x * \sigma * \sqrt{2\pi})) * \exp(-(ln(x)-\mu)^2/(2\sigma^2))$

CDF(at $X=x$): $\frac{1}{2} + [\frac{1}{2} * \text{erf}((ln(x)-\mu)/(\sigma * \sqrt{2}))]$

Mean: $\exp(\mu + (\sigma^2/2))$

Median: $\exp(\mu)$

Mode: $\exp(\mu - \sigma^2)$

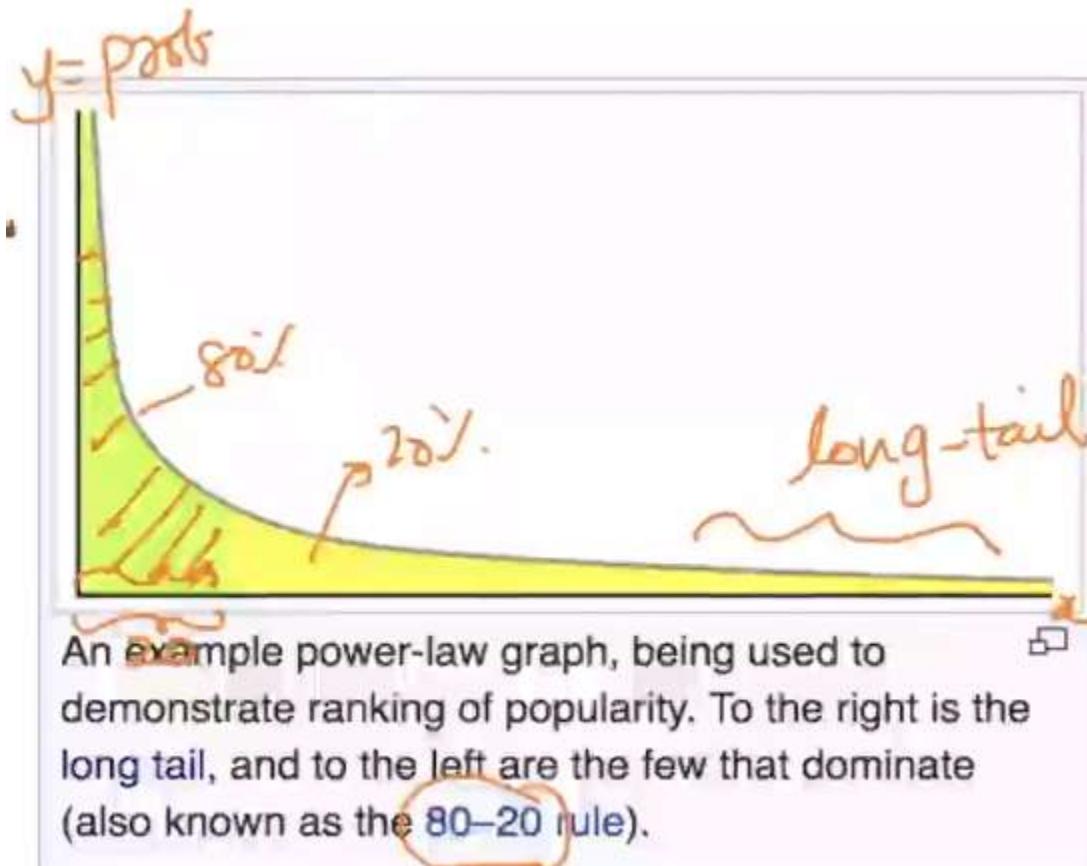
Variance: $[\exp(\sigma^2) - 1] * \exp(2\mu + \sigma^2)$

Skewness: $(\exp(\sigma^2) + 2) * \sqrt{\exp(\sigma^2) - 1}$

22.12 Power Law Distributions

A power law is a functional relationship between two quantities where a relative change in one quantity results in a proportional relative change in the other quantity, independent of the initial size of those quantities. (ie., one quantity varies as the power of the other).

Power Law follows the 80-20 rule. That in the given distribution 'X', 80% of the values of the distribution lie below 20% of the values of 'X'. Whenever a distribution follows Power Law, that distribution is called **Pareto Distribution**. There is a long tail for Power Law functions. Pareto Distributions are for the continuous random variables. You can find the PDF of a Power Law distribution below, which was discussed starting from the timestamp 0:20.



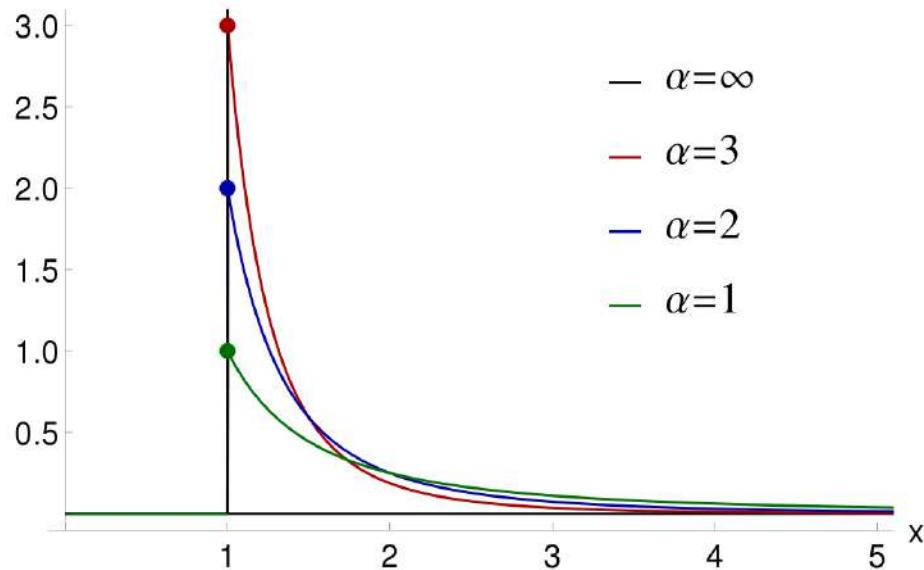
Pareto Distribution

The parameters of the Pareto distribution are

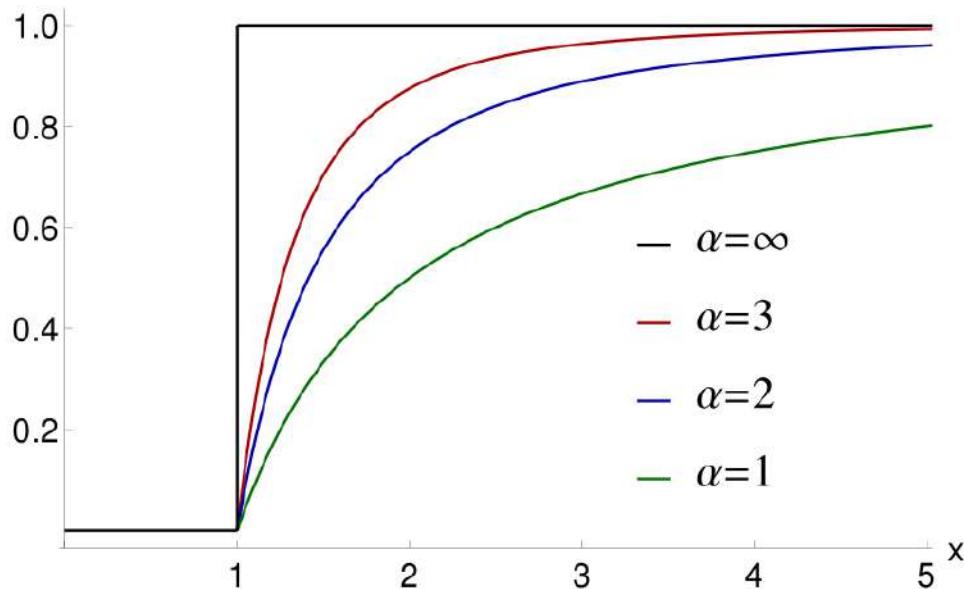
- 1) $x_m > 0$. This parameter is called 'scale' and takes only real values. It is similar to ' μ ' in a gaussian distribution.
- 2) $\alpha > 0$. This parameter is called 'shape' and takes only real values. It is similar to ' σ ' in a gaussian distribution.

Let us look at the PDF and CDF plots of a pareto distribution that were discussed starting from the timestamp 2:35.

$\Pr(X=x)$



$\Pr(X \leq x)$



From the PDF plots, we observe that as the ‘ α ’ value keeps decreasing, the tails become less fatter. For $\alpha \rightarrow \infty$, the PDF becomes a delta function(i.e., a straight vertical line with a single value). Here this delta function has a value only at one point, whereas on the other points, it takes the value as 0. Such a function is called **Dirac Delta Function**.

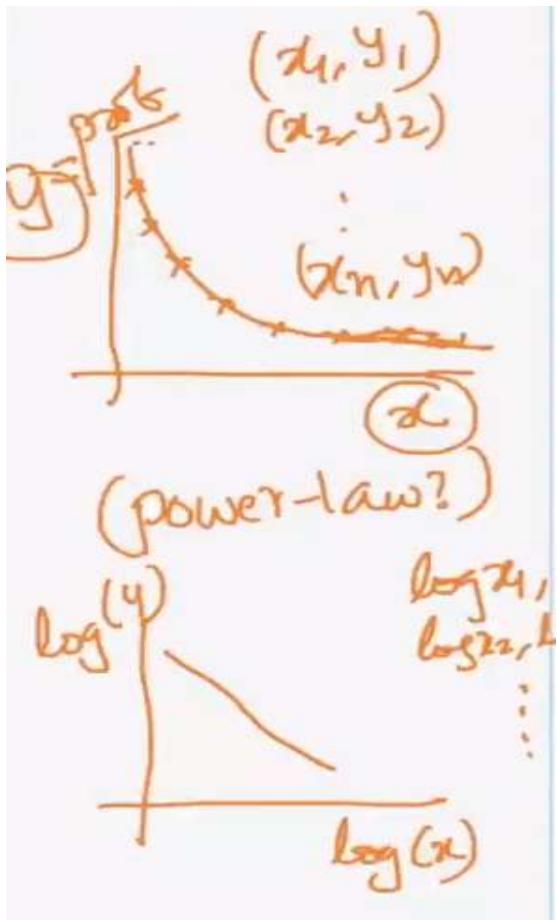
The common point in both Pareto and Log Normal Distributions is “Both the distributions have a small number of larger values and a large number of smaller values. But the major difference is the Pareto Distribution does not have an increasing PDF.

How to check if the two given variables follow Power Law?

The one approach to check if the two given variables follow Power Law is using Log-Log plot.

If ‘X’ and ‘Y’ are two given variables, then if we make a plot with $\text{Log}(X)$ on X-axis and $\text{Log}(Y)$ on Y-axis and if the plot converges to a straight line as shown in the figure, then we can say that the distribution has power law tail. (ie., both the variables follow power law)

A straight line on a log-log plot is a strong evidence for power laws, and the slope of the straight line corresponds to the power law exponent.



Disadvantages of Log-Log Plot

- 1) Log-Log plot is insufficient evidence for a power law relationship as many non power law distributions also appeared like straight lines on log-log plots.
- 2) In order to provide reliable results using a log-log plot, we need huge amounts of data.
- 3) Log-Log plots are appropriate for discrete data.

How to check whether a given distribution is a Pareto Distribution?

Let us assume 'X' is a continuous random variable and we have to check whether it has Pareto Distribution using QQ plot.

We have to take a reference Pareto distribution with continuous random variable 'Y'. We shall calculate the percentile values of 'X' and 'Y' and then we have to plot the points $(y^{(i)}, x^{(i)})$ where

$y^{(i)} \rightarrow i^{\text{th}}$ percentile of 'Y'

$x^{(i)} \rightarrow i^{\text{th}}$ percentile of 'X'

If the result is a straight line, then we can say both follow the same distribution.

20.13 Box Cox Transform

So far we have seen random variables following log-normal distribution and when we apply natural logarithm on it, it turns into gaussian distribution. We generally expect the numerical features to follow gaussian distribution because most of the ML models work on the assumption that all the numerical features in the data follow gaussian distribution.

Power Transform is a data transformation technique used to stabilize variance, make the data more like a normal distribution and improve the validity of measures of association between variables and for other stabilization procedures. One such popular transform is Box Cox Transform.

Box Cox transform is a power transform technique used to convert a non gaussian distribution into a gaussian distribution.

Procedure of Box-Cox Transform

Let us assume, we have a random variable 'X'(say it follows pareto distribution) and we want to transform it into a gaussian distribution denoted by 'Y'.

Let $X = [x_1, x_2, x_3, \dots, x_n]$ is a pareto distribution. The output should be $Y = [y_1, y_2, y_3, \dots, y_n]$ that follows a gaussian distribution. (Here 'X' can be any non gaussian distribution. It is not mandatory for 'X' to be a pareto distribution).

- 1) When we apply box-cox transform on a random variable 'X', then it returns ' λ '.

$$\text{Box-Cox}(X) = \lambda$$

- 2) Compute $y_i = (x_i^{\lambda} - 1)/\lambda$, if $\lambda \neq 0$ (or) $\log_e(x_i)$ if $\lambda = 0$. ($\forall i \rightarrow 1 \text{ to } n$)

If $\lambda = 0$, then the given original distribution is log-normal.

In the conversion process, if we get $\lambda = 0$, then it means 'X' is following log-normal distribution and we have to convert 'X' into gaussian form by applying natural logarithm on every element.

If we get $\lambda \neq 0$, we can go ahead in converting 'X' into gaussian form by the formula

$$y_i = (x_i^{\lambda} - 1)/\lambda$$

Note: Box Cox Transform is not guaranteed to work on all pareto or power law distributed data. It works only on some of them and we need to perform the box-cox transform and observe the QQ plot and confirm if it is working well on our data.

We are finding the best ' λ ' using the data we currently have. If the data we have is a good representative sample of all the possible values, the ' λ ' we computed using the current data would be reasonably good for future unseen data also.

Given any new data, we can apply the transformation using the ' λ ' already obtained to transform the new data into gaussian distributed data. Box Cox Transform is a one-one transform and we can obtain the original data back from the transformed data.

<u>λ value</u>	<u>Transformation used in Box Cox Transform</u>
$\lambda = -1$	Reciprocal Transform
$\lambda = -0.5$	Reciprocal Square Root Transform
$\lambda = 0$	Log Transform
$\lambda = 0.5$	Square Root Transform
$\lambda = 1$	No Transform

Below is the code snippet that is used for applying box-cox transform on a given distribution of the data and it was discussed at the timestamp 7:05.

```
>>> from scipy import stats
>>> import matplotlib.pyplot as plt
```

We generate some random variates from a non-normal distribution and make a probability plot for it, to show it is non-normal in the tails:

```
>>> fig = plt.figure()
>>> ax1 = fig.add_subplot(211)
>>> x = stats.loggamma.rvs(5, size=500) + 5
>>> prob = stats.probplot(x, dist=stats.norm, plot=ax1)
>>> ax1.set_xlabel('')
>>> ax1.set_title('Probplot against normal distribution')
```

We now use **boxcox** to transform the data so it's closest to normal:

```
>>> ax2 = fig.add_subplot(212)
>>> xt, _ = stats.boxcox(x)
>>> prob = stats.probplot(xt, dist=stats.norm, plot=ax2)
>>> ax2.set_title('Probplot after Box-Cox transformation')
>>> plt.show()
```

Note: As the 20.14 video lecture was only about examples, we are not adding any notes for it. You can just go through the video just to get an idea of it.

20.15 Covariance

In our dataset, sometimes there exists relationships between two or more variables(ie., increase/decrease in the value of a variable may decrease/increase the values of another variable). In order to quantify these relationships between the variables, we have 3 measures.

- 1) Covariance
- 2) Pearson Correlation Coefficient
- 3) Spearman Rank Correlation Coefficient

Covariance

Let us assume we have 'N' observations with random variables 'X' and 'Y' in pairs. Then the covariance between them is defined as

$$\text{Covariance}(X,Y) = (1/N) * \sum_{i=1}^N (x_i - \mu_x) * (y_i - \mu_y)$$

Let us recall the variance formula.

$$\text{Variance}(X) = (1/N) * \sum_{i=1}^N (x_i - \mu_x)^2$$

When we compare both the equations, then we can conclude that

$$\text{Covariance}(X,X) = \text{Variance}(X)$$

Covariance measures only the linear relationships between the random variables 'X' and 'Y'. So when we take the covariance between two variables 'X' and 'Y', we have

$$\text{Covariance}(X,Y) = (1/N) * \sum_{i=1}^N (x_i - \mu_x) * (y_i - \mu_y)$$

The covariance between the two random variables 'X' and 'Y' is said to be positive only

- a) If the value of 'X' increases with an increase in the value of 'Y'.
- b) If the value of 'X' decreases with a decrease in the value of 'Y'.

Similarly the covariance between the two random variables 'X' and 'Y' is said to be negative only

- a) If the value of 'X' increases with a decrease in the value of 'Y'.
- b) If the value of 'X' decreases with an increase in the value of 'Y'

Below are the plots that show how the relationship between two variables exists for positive and negative covariances. It was discussed starting from the timestamp 6:20

$$\text{Cov}(X,Y) = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_x) * (y_i - \mu_y)$$

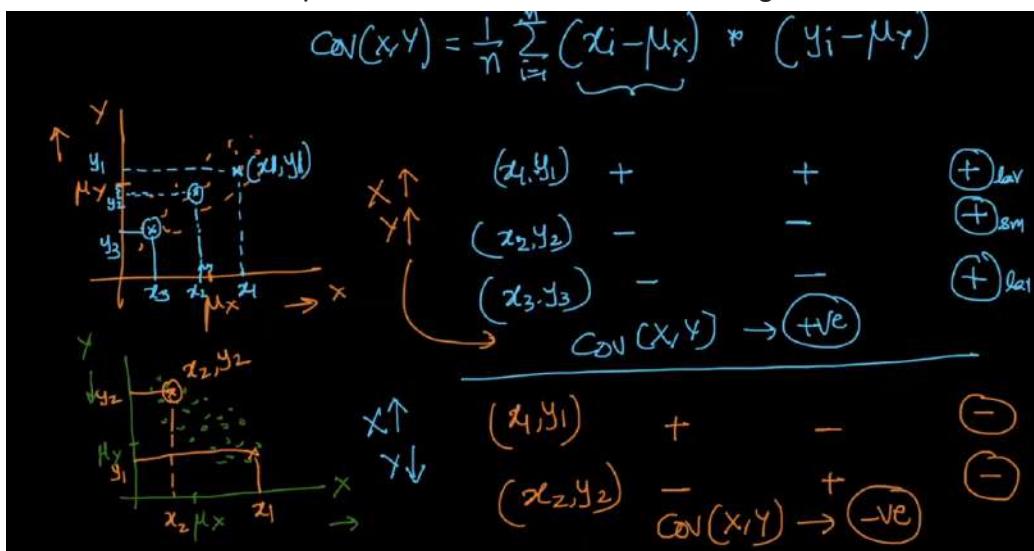


In the above picture, the orange plot shows the existence of the positive covariance and the green plot shows the existence of the negative covariance between the random variables 'X' and 'Y'.

In the orange plot, as the value of 'X' increases, the value of 'Y' also increases. It indicates positive covariance between 'X' and 'Y'. In the green plot, as the value of 'X' increases, the value of 'Y' decreases. It indicates negative covariance between 'X' and 'Y'.

Graphical Representation

Below are the plots that were discussed starting from the timestamp 6:20.



<u>Pairs(X,Y)</u>	<u>Sign of $(x_i - \mu_x)$</u>	<u>Sign of $(y_i - \mu_y)$</u>
(x_1, y_1)	+ve	+ve
(x_2, y_2)	-ve	-ve
(x_3, y_3)	+ve	-ve
(x_4, y_4)	-ve	+ve

In the first two combinations, both the terms(ie., $(x_i - \mu_x)$ and $(y_i - \mu_y)$) are positive, so the covariance becomes positive. In the 3rd and 4th combinations, one of the terms is negative and the other term is positive, so the covariance becomes negative, as the product of these two terms is negative.

Disadvantage of Covariance

When we want to find out the covariance between two variables 'X' and 'Y', if the scale/metric changes, then the covariance score changes drastically. This drawback is overcome by Pearson Correlation coefficient.

Whenever we have 2 variables in different units(ex: kg and pounds), if we want to compute covariance, we first should bring the variables onto the same units, and then compute the covariance.

Whenever the covariance between two random variables is 0, then it doesn't mean that the two random variables are independent of each other. It only means that there doesn't exist any linear relationship between the two random variables. There might exist a non linear relationship between these two random variables.

Significance of Magnitude and Sign of Covariance

The sign of the covariance shows the tendency in the linear relationship between the variables. The magnitude of the covariance is not easy to interpret because it is not standardized and hence it depends on the magnitudes of the variables.

The standardized version of the covariance shows the strength of the linear relationship by its magnitude. If we standardize the data, then the covariance becomes correlation.

If the scale of the variables change, only the magnitude of the covariance changes, but not the sign of the covariance. The sign of the covariance remains the same. When it comes to checking whether the correlation between two random variables is positive or negative, we have to focus only on the sign of the covariance, but not on the magnitude.

When we have to check how strongly/weakly the variables are correlated, we have to consider the magnitude also. For example, if we have 3 random variables 'X', 'Y' and 'Z', then if a small change in 'X', results in a drastic change in 'Y', then we say that 'X' and 'Y' are strongly correlated. If a small change in 'X', results in a small change in 'Z', then we say that 'X' and 'Z' are weakly correlated.

So whether two random variables are strongly correlated or weakly correlated, is decided by the magnitude of the covariance. Having a large magnitude indicates a strong correlation and a small magnitude indicates a weak correlation. In this case, the magnitude of the covariance plays a key role.

Similarly, if we have a positive covariance, then we can say that correlation between two quantities is positive. If we have a negative covariance, then we can say that correlation between two quantities is negative. In this case, the sign of the covariance plays a key role.

20.16 Pearson Correlation Coefficient(ρ)

Pearson Correlation Coefficient is a measure of the linear correlation between two random variables 'X' and 'Y'. It has a value between -1 and 1.

- 1 → Total Positive Linear Correlation
- 0 → No Linear Correlation
- 1 → Total Negative Linear Correlation

Pearson Correlation Coefficient is the covariance of the two variables divided by the product of their standard deviations.

Pearson Correlation Coefficient when applied to a population is commonly represented by ' ρ ' and may be referred to as the Population Correlation Coefficient (or) Population Pearson Correlation Coefficient.

Given a pair of random variables (X,Y), the formula for ' ρ ' is

$$\rho_{x,y} = (1/(\sigma_x * \sigma_y)) * \text{Covariance}(X, Y)$$

σ_x → standard deviation of 'X'

σ_y → standard deviation of 'Y'

Pearson Correlation Coefficient when applied to a sample is commonly represented by ' r ' and may be referred to as the Sample Correlation Coefficient (or) Sample Pearson Correlation Coefficient.

Given a sample of points for random variables (X,Y), the formula for ' r_{xy} ' is given as

$$r_{xy} = (\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})) / (\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} * \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2})$$

n → number of points in the sample

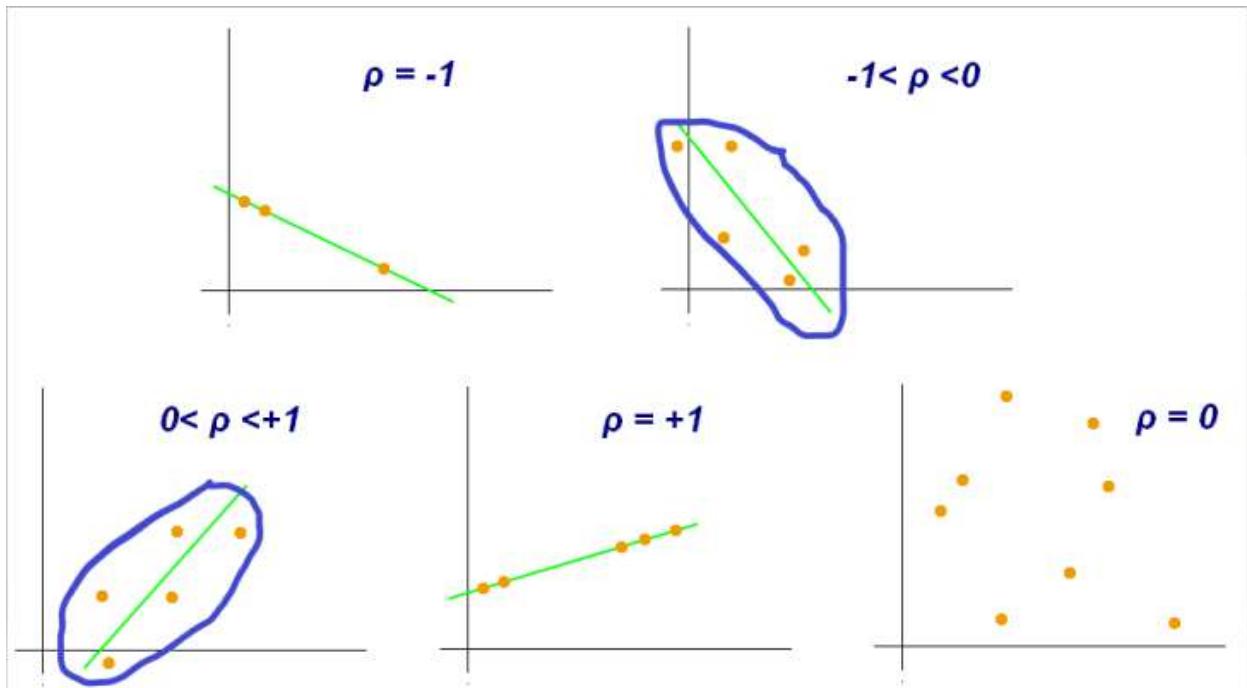
\bar{x} → sample mean of 'X'

\bar{y} → sample mean of 'Y'

The problem with the Covariance is that it doesn't take the standard deviations of the random variables into consideration, whereas the Pearson Correlation Coefficient takes the standard deviations into consideration.

Different cases that show how the value of the Pearson Correlation Coefficient Changes

Below are the cases that show how the PCC value changes with different correlations among the data. It has been discussed starting from the timestamp 2:00

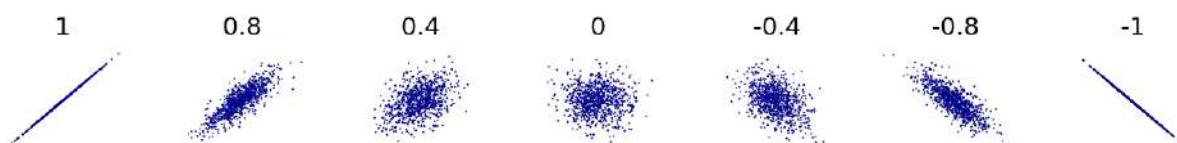


If the points are more spread on both sides of a straight line, but still in an oval shape, the ' ρ ' value lies in between (-1,0) or (0,1). If all the points lie on a straight line, then only we can expect the ' ρ ' value to be either +1 or -1.

There is also a limitation with PCC. Since both Covariance and PCC are biased towards the linear relationships, we could get a better score of PCC, when the points lie on a straight line. If the points are little dispersed, we get a low score of PCC.

We also can see different scenarios of correlation, that was discussed starting from the timestamp 5:15

a) Type 1 (on the basis of the deviation from the straight line)



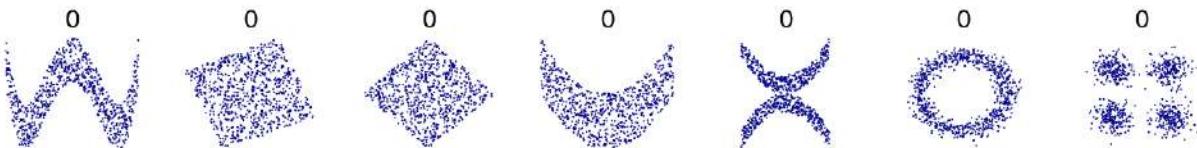
The maximum variance is seen when $\rho=0$ and the least variance is seen when $\rho=1$ and $\rho=-1$

b) Type 2 (on the basis of the slope)



The value of ' ρ ' will be either 1 or -1, if there exists a linear relationship between 'X' and 'Y'. It doesn't bother about the angle the lines make with the ground. It only bothers about whether a linear relationship exists between the variables or not.

c) Type 3 (on the basis of linearity)



In all the above nonlinear relationships, $\rho=0$ because PCC cannot find the correlation if both the given variables are non linear. In order to estimate non linear distributions, we have to build a full fledged model.

Note:

Covariance and PCC do not work for nonlinear relationships. When it comes to linear relationships, Covariance doesn't take the variability into consideration, due to which we see huge differences in the magnitude of covariance when the distance of the points from the mean varies. Whereas PCC takes the standard deviation into consideration.

We cannot make any inferences on the basis of the covariance score, whereas we can make inferences on the basis of the PCC score.

PCC = 1 indicates a perfect positive linear relationship between the features.

PCC = -1 indicates a perfect negative linear relationship between the features.

In the above two cases, all the points fit exactly on a straight line. But in real time, it is always not possible for all the points to fit on a straight line.

PCC cannot handle monotonic functions such as non decreasing or non increasing functions because it can handle only linear relationships whereas Spearman's Rank Correlation Coefficient (SRCC) can handle monotonic functions. When the data doesn't follow normal distribution, then SRCC may be more appropriate. Since PCC works only if the relationship is linear, it doesn't give an appropriate value when the two given random variables 'X' and 'Y' are not linearly related. If 'X' and 'Y' are not linearly related, then **Covariance(X,Y) = PCC(X,Y) = 0**.

20.17 Spearman Rank Correlation Coefficient (SRCC)

Pearson Correlation Coefficient is useful when the relationship between two random variables is linear. When we have non linear relationships (like monotonically increasing/ decreasing/ non increasing / non decreasing functions), the best measure for correlation is Spearman Rank Correlation Coefficient.

Let us look at the below example that was discussed starting from the timestamp 1:00..

X	Y	Rank of values of 'X' (r_x)	Rank of values of 'Y' (r_y)
160	52	4	3
150	66	2	4
170	68	5	5
140	46	1	1
158	51	3	2

The value with the least number will have less rank. Instead of computing the PCC on 'X' and 'Y', SRCC says to compute the PCC between the ranks of 'X' and 'Y'.

$$\text{SRCC}(r) = \rho_{r_x, r_y}$$

In case of SRCC

- If 'Y' increases with an increase in 'X', irrespective of whether the relationship between 'X' and 'Y' is linear or not, the value of SRCC = 1.
- If 'Y' decreases with a decrease in 'X', irrespective of whether the relationship between 'X' and 'Y' is linear or not, the value of SRCC = 1.
- If 'Y' increases with a decrease in 'X' (or) if 'Y' decreases with an increase in 'X', then irrespective of whether the relationship between 'X' and 'Y' is linear or not, the value of SRCC = -1.

In case of PCC

- If 'Y' increases with an increase in 'X' (or) if 'Y' decreases with a decrease in 'X', then if the relationship between 'X' and 'Y' is linear, then PCC = 1.
- If 'Y' increases with a decrease in 'X' (or) if 'Y' decreases with an increase in 'X', then if the relationship between 'X' and 'Y' is linear, then PCC = -1.
- If the relationship between 'X' and 'Y' is non linear, then PCC is not applicable.
- If $\rho_{x,y} = 0$, it doesn't mean that there is no relationship between 'X' and 'Y'. It only means that 'X' and 'Y' are not linearly related to each other.
- Similarly if $r = \rho_{r_x, r_y} = 0$, it doesn't mean that there is no relationship between 'X' and 'Y'. It only means that 'X' and 'Y' are not monotonically related to each other.
- While assigning the ranks to the data of random variables while estimating SRCC, if two or more numbers are same, then some implementations use the

same rank and some other implementations use fractional ranks.

For example, if the rank '5' is repeated two times for two numbers then we skip the rank '6' and give the next entity, a rank of '7'. For these two values, we distribute the ranks between '5' and '6', which is 5.5, to both of them. This is called Fractional Ranking and it is mostly used.

- SRCC is the PCC of ranks. If the ranks have a covariance of 0, then the SRCC also will be 0. So when the PCC of ranks is 0, then the SRCC also will become 0.
- The assumptions of SRCC are that data must be at least ordinal and the scores on one variable must be monotonically related to the other variable. In an ordinal scale, the levels of variables are ordered such that one level can be considered higher/lower than another.
- SRCC works only if two variables are monotonically increasing or decreasing, but not both at a time. In case, if we have both monotonically increasing and monotonically decreasing natures together, then we should split the function into intervals and then apply SRCC on each nature separately.

Disadvantages of SRCC

SRCC cannot completely overcome the problem of non-linear relationships, as it uses only the information of the order through ranks.

For example, if we take $Y=\sin(X)$, if we plot 'X' (vs) 'Y', visually we'll be able to see a correlation between 'X' and 'Y'. But SRCC fails to understand the correlation between the values as it is concerned only with the ordering using the ranks and monotonicity.

We are losing some information by taking the ranks into consideration, but SRCC is still used in real world scenarios where we do not care about exact numeric values, but only about the order.

If $\text{SRCC}(X,Y) = 0$, then it only means 'X' and 'Y' are monotonically independent. They also could be related in non-monotonic ways like $Y = \sin(X)$ (or) $Y = \cos(X)$, etc.

20.18 Correlation vs Causation

Correlation doesn't imply causation. We sometimes see as 'X' increases, 'Y' also increases and as 'X' decreases, 'Y' also decreases. Here due to the presence of correlation between 'X' and 'Y', we cannot say that 'X' causes 'Y'.

Even though the correlation between 'X' and 'Y' is very high, we cannot say 'X' causes 'Y'. In order to confirm whether 'X' causes 'Y', we need to have the domain expertise. If 'X' and 'Y' are correlated, then sometimes 'X' may cause 'Y' and sometimes it may not.

Correlation doesn't imply Causation all the time. It implies only sometimes.

20.19 How to use Correlations?

Example 1: Is salary correlated with the square feet of home?

Here high salaried people look for homes with higher square feet. It is not always mandatory for high salaried employees to purchase only homes with higher square feet, but mostly the high salaried people prefer higher square feet houses. Also if the people are highly salaried, then if they prefer higher square feet houses, it makes the business easier and gain huge profits. Also if we mention our salary range, then the real estate company can estimate what kind of house we are interested, depending on the budget and shows us only those houses.

Example 2: Is the number of years of education correlated with the income?

For the education or labour department, if they know that the number of years of education fetches more income, then the department makes sure every individual gets as many number of years of education in order to generate more income.

Example 3: Is the time spent on an e-commerce web page in the last 24 hours correlated with money spent on the website?

From the data we have, for multiple users, if we can draw the conclusions that the total money spent on the website by each customer increases, as the time spent by him/her browsing the web page increases, then the e-commerce company can design the website in a much more attractive way such that the customers spend more time browsing it and spend more money.

Also one more correlation can be made between the number of unique customers (vs) the total sales. If the number of unique customers increases, the total sales also increase. So the business focuses on increasing the number of customers, so that it could increase the sales.

Note

If the correlation coefficient is positive, then the two variables are directly proportional. If the correlation coefficient is negative, then the two variables are inversely proportional.

First of all, we have to check if correlation exists between the variables. Once the presence of correlation is confirmed, then we can perform tests for the presence of causality.

Normalizing the covariance to the range [-1,1] gives the correlation.

Causation indicates a relationship between two events where one event is affected by the other. In statistics, when the value of one event or variable increases or decreases, as a result of other events, then there is said to be causation.

20.20 Confidence Interval (CI): An Introduction

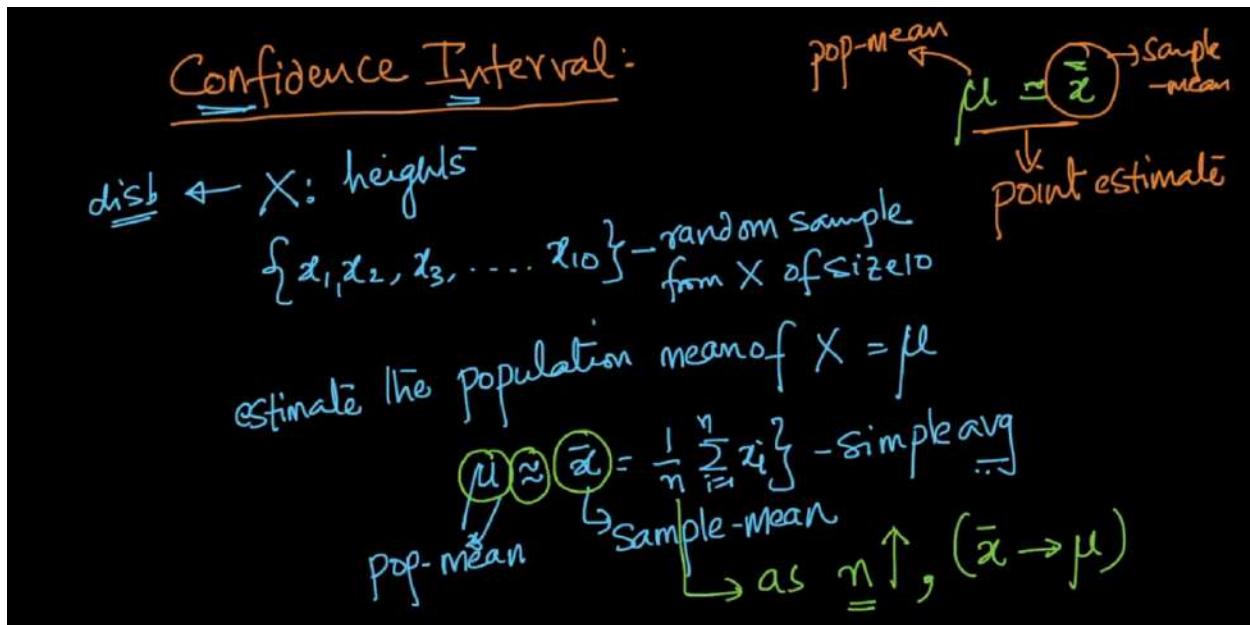
Let us assume we have a random variable 'X' that represents the heights of the people. Let us assume we have a sample of 10 points.

$$X = \{x_1, x_2, x_3, \dots, x_{10}\} \text{ (random sample of size 10)}$$

Our job is to estimate the population mean ' μ ' of 'X'.

$$\mu = \bar{x} = (1/n) * \sum_{i=1}^n x_i \quad (\bar{x} \rightarrow \text{sample mean}, 'n' \rightarrow \text{sample size})$$

As the value of 'n' increases, the sample mean becomes more and more nearer to the population mean. If the estimate of the population mean(μ) is exactly equal to the sample mean(\bar{x}), then it is called **point estimate**.



Point Estimate of Population Mean(μ) = Sample Mean(\bar{x}) = $(1/n) * \sum_{i=1}^n x_i$ ('n' \rightarrow sample size)

In case, if we can make a statement that the population mean(μ) lies in an interval $[a,b]$ with 'x%' probability, then it means we are saying with x% confidence that the population mean lies in the interval $[a,b]$.

$M \in [a,b]$. Here 'x%' is the percentage of confidence.

$[a,b]$ is the interval estimate of the population mean and the technical name given to this interval is called **Confidence Interval**.

In the above example, we can say that in 'x%' of the cases, the population mean(μ) lies in the interval $[a,b]$ and in the remaining '(100-x)%' of the cases, the population mean(μ) lies outside the interval $[a,b]$.

Making an interval estimate is much more richer in terms of population estimate when compared to a point estimate.

Need for Confidence Interval over the point estimates

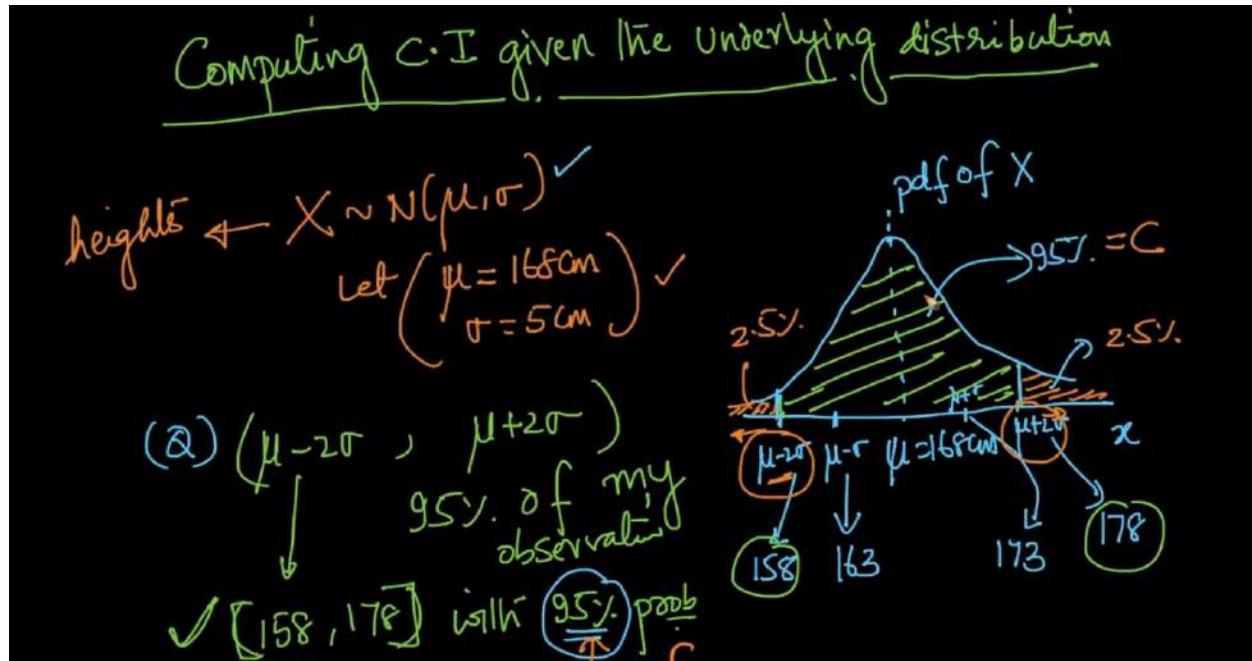
It is always not possible to consider the whole data and find the mean and standard deviation values in real time, so we work with samples many times.

The reason why we prefer interval estimates over the point estimates is that, every time if we keep changing the points in the samples, then the point estimate (ie., sample mean) keeps changing every time. So it is better to give an interval estimate than a point estimate and hence we choose confidence intervals.

Interval estimates are preferred by the statisticians because the interval estimates are accompanied by a statement concerning the degree of confidence that the interval contains the population parameter being estimated.

It is desirable for a point estimate to be consistent so the larger the sample size, the more accurate the estimate would be (or) we can say the population mean is nearly equal to the sample mean, but in general the sample mean varies from one sample to another.

20.21 Computing the Confidence Interval given the underlying distribution



Let us assume we have a random variable 'X' and it follows normal distribution.
 $X \sim N(\mu, \sigma)$

From the 68-95-99.7 rule, we know that 95% of the values lie in the interval $[\mu - 2\sigma, \mu + 2\sigma]$. In this case, we shall denote 95% with 'C'.

So $C = 95\%$. 'C' denotes confidence.

In case, if we have to find the confidence interval of 90% of the values, then $C = 90\%$. In order to compute the confidence intervals for the values $C = 90\%, 80\%, 70\%$, etc, we have to check the normal distribution table.

In case, if the given distribution is not a gaussian distribution and we are asked to find out the confidence interval, then for the confidence interval estimation of non gaussian distribution, we have to go for **Bootstrapping**.

If the given distribution is gaussian, then irrespective of the number of points in the distribution, the confidence intervals are the same. That is

- I. 68% confidence interval $\rightarrow [\mu - \sigma, \mu + \sigma]$
- II. 95% confidence interval $\rightarrow [\mu - 2\sigma, \mu + 2\sigma]$
- III. 99.7% confidence interval $\rightarrow [\mu - 3\sigma, \mu + 3\sigma]$

In case of the gaussian distribution, if the data points increase, then there are chances for the confidence interval limits to change. If the sample size is increased, the confidence interval range reduces. Similarly, if the sample size is decreased, the confidence interval range increases.

According to the Central Limit Theorem, the population mean(μ) lies in the interval $[\bar{x} - (2\sigma/\sqrt{n}), \bar{x} + (2\sigma/\sqrt{n})]$ with 95% confidence. (where \bar{x} → sample mean, n → sample size).

What is the use of Confidence Interval?

The purpose of confidence intervals is to give us a range of values for our estimated population parameter rather than a single value or a point estimate. The estimated confidence interval gives us a range of values within which we believe with certain probability that the true population value falls. This probability is confidence level.

For instance, if repeated samples were taken and 95% confidence interval for the mean was computed for each sample, then 95% of the intervals would contain the population mean. We expect that 5% of the interval would not contain the true value.

Computing the Confidence Interval using CDF

For example, if we want to compute the range of values within which 95% of the values lie, then we have to first compute the 2.5 percentile (say 'a') and 97.5 percentile (say 'b') values from the CDF. So we can conclude that the random variable has a 95% confidence interval as [a,b] which is computed using the CDF.

20.22 Confidence Interval for mean of a random variable

Let us assume we have a distribution 'X' (not sure about the type of distribution) with a population mean of ' μ ' and a standard deviation of ' σ '. Let us assume we have picked a sample of size 10 (ie., $x_1, x_2, x_3, \dots, x_{10}$).

Let us assume we have to find out the 95% confidence interval estimate for the population mean.

Case 1: (If we already knew the population standard deviation ' σ ')

Discussed starting from the timestamp 2:30

The handwritten notes show the following steps:

- Case 1:** $\sigma = 5 \text{ cm}$ {we know pop-std-dev}
- CLT:** $\bar{x} = \text{Sample Mean} = \frac{1}{10} \sum_{i=1}^{10} x_i$ ($n = 10$)
- $\bar{x} \sim N(\mu, \frac{\sigma}{\sqrt{n}})$ (Gaussian)
- \bar{x} is the Sample Mean, μ is the pop-mean, and $\frac{\sigma}{\sqrt{n}}$ is the pop-std-dev.
- $\bar{x} = 168.5 \text{ cm}$, $\sqrt{n} = \sqrt{10}$
- $\mu \in [\bar{x} - 2\frac{\sigma}{\sqrt{n}}, \bar{x} + 2\frac{\sigma}{\sqrt{n}}]$ with 95% confidence

From Central Limit Theorem, we have learnt that if we have any distribution (whether it is gaussian or non gaussian) with a finite and valid values of the population mean and standard deviation, then

the random variable $\bar{x} \sim N(\mu, \sigma/\sqrt{n})$ where 'n' → sample size, \bar{x} → mean of all sample means.

The sample mean follows a gaussian distribution with the population mean(μ) as its mean and a standard deviation of ' σ/\sqrt{n} '. So now, we can say that the population mean(μ) lies in $\mu \in [\bar{x} - 2\sigma/\sqrt{n}, \bar{x} + 2\sigma/\sqrt{n}]$.

Case 2: (If the population standard deviation ' σ ' is not known)

Discussed starting from the timestamp 7:45.

Case 2: if we don't know σ (pop-std-dev)

Sample of n

t-distr

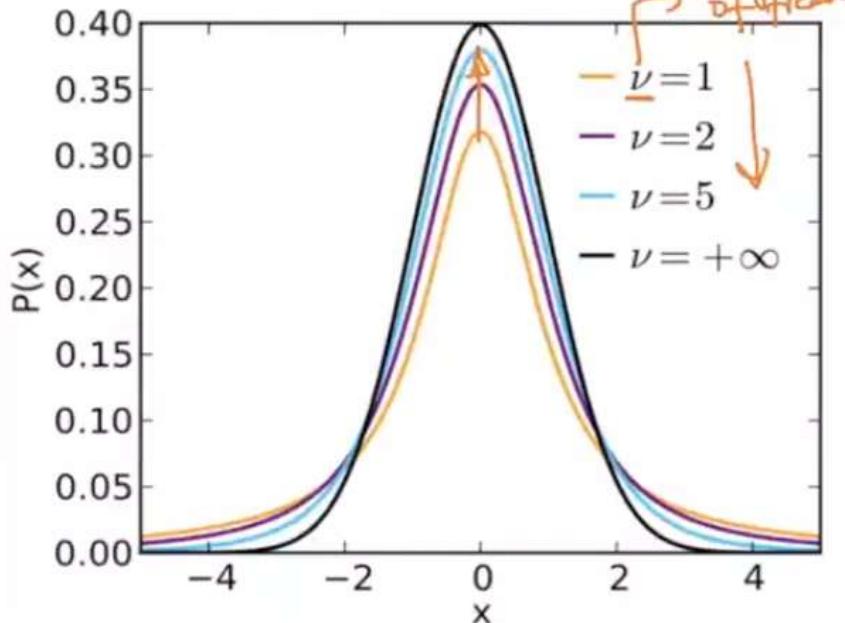
Student's t-distr

$$\bar{x} \sim t(n-1)$$

↑
degrees of freedom
sample mean t-distr

Student's t

Probability density function



Cumulative distribution function

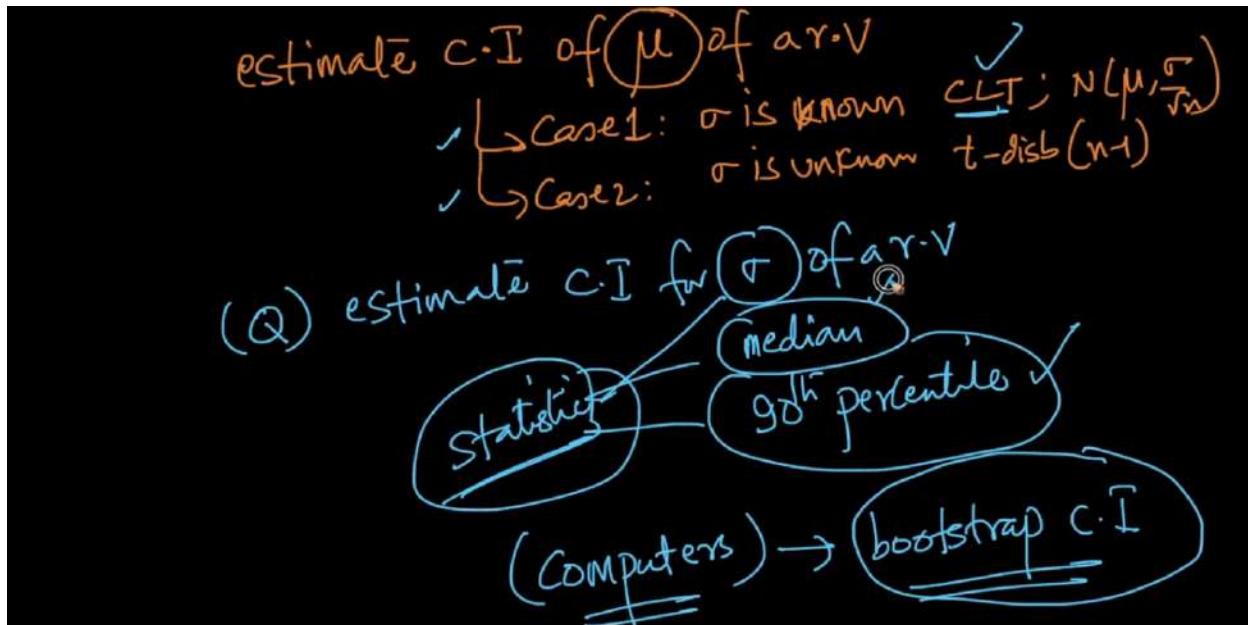
If we do not know the population standard deviation(σ), then we could not apply the Central Limit theorem, as the value of ' σ ' is unknown. So we need to go for the t-distribution for estimating the confidence interval of the population mean. This t-distribution is also known as **Student's t-distribution**.

Theoretically the student's t-distribution states that the sample mean follows t-distribution with $(n-1)$ degrees of freedom.

$$\bar{x} \sim t(n-1)$$

Where 't' denotes the t-distribution and $(n-1)$ denotes the degrees of freedom.

The PDF of student's t-distribution looks similar to the gaussian curve, but not exactly a gaussian curve. The student's t-distribution is exclusively developed for computing the confidence intervals. The major application of student's t-distribution is computing the confidence intervals, when the population standard deviation is unknown.



So in order to estimate the confidence intervals of the population mean,

- If the population standard deviation is known, we have to apply CLT.
- If the population standard deviation is unknown, we have to apply the student's t-distribution.

But when it comes to the estimation of other population statistics (like population standard deviation, median, 90th percentile, etc), the above 2 techniques do not work. So we have to go for a technique called **Bootstrapping**.

22.23 Confidence Interval using Bootstrapping

Bootstrapping can be used to compute the confidence intervals not only for the mean, but also for the other statistics like median, variance, standard deviation, 90th percentile, etc.

Let us assume we are given a random variable 'X' whose distribution is unknown. Let us pick a sample of size 'n' and we have to compute the confidence interval of the median of the population of 'X'.

Let 'S' be the sample and its size be 'n'.

$$S = \{x_1, x_2, x_3, \dots, x_n\}$$

Let us now pick a sample ' S_1 ' of size 'm' from 'S', such that $m \leq n$.

$$S_1 = \{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_m^{(1)}\} \quad (m \leq n)$$

Here ' S ' → Random sample of size 'n' generated from the population.

' S_1 ' → Random sample of size 'm' generated from the sample 'S'.

The points in ' S_1 ' are selected randomly one after the other from 'S' while sampling. Here we may also get repetition of the points in ' S_1 ' while sampling from 'S'. This is called Sampling with Repetition.

While sampling the points from 'S' into ' S_1 ', the chances of selecting a point is equal for all the points. Hence we generate a uniform random variable $U(1,n)$ and we run a loop for 'm' times, as we have to pick 'm' points into the sample ' S_1 '. While running this loop for 'm' times and picking 'm' points randomly, we may get some repeated points. Here 'U' is a discrete uniform random variable.

So now from the sample 'S', we generate 'K' different samples, each of size 'm' using a random distributed discrete random variable using sampling with replacement.

$$S_1 = \{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_m^{(1)}\}$$

$$S_2 = \{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \dots, x_m^{(2)}\}$$

$$S_3 = \{x_1^{(3)}, x_2^{(3)}, x_3^{(3)}, \dots, x_m^{(3)}\}$$

.

.

.

$$S_k = \{x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \dots, x_m^{(k)}\}$$

These samples are created artificially from the sample 'S' and hence they are called **Bootstrap Samples**. As our task is not to compute the confidence intervals for the population median, we have to compute the medians of each of these samples.

$$S_1 = \{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_m^{(1)}\} \rightarrow 'm_1' \text{ is the median}$$

$$S_2 = \{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \dots, x_m^{(2)}\} \rightarrow 'm_2' \text{ is the median}$$

$$S_3 = \{x_1^{(3)}, x_2^{(3)}, x_3^{(3)}, \dots, x_m^{(3)}\} \rightarrow 'm_3' \text{ is the median}$$

.

.

$$S_k = \{x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \dots, x_m^{(k)}\} \rightarrow 'm_k' \text{ is the median}$$

These are the medians computed for the bootstrap samples. We now have to sort all these medians and let those sorted medians be

$$m_1' \leq m_2' \leq m_3' \leq \dots \leq m_k'$$

From these sorted medians, we have to obtain the confidence intervals.

For example if $k=1000$, then it means we have 1000 bootstrap samples and 1000 medians associated with them. So then the sorted means would be m_1' , m_2' , m_3' , ..., m_{1000}' .

In case, if we want to compute the 95% confidence interval, as the remaining is 5%, we shall divide it into two halves and it would be 2.5% each. (It is because the CLT is applicable for not only mean, but also for other statistics like median, 90th percentile, etc. That is if we pick multiple samples and then compute the median for each of these samples, then all these sample medians follow normal distribution. So 68-95-99.7 rule applies for these sample medians.)

Among the sorted medians, pick the lower bound such that 2.5% of the medians are to its left side and pick the upper bound such that 2.5% of the medians are to its right side. The interval with these lower and upper bounds will be the 95% confidence interval.

(Here out of 1000 medians, 2.5% is 25. So the lower and the upper bounds should be chosen in such a way that 25 medians are present on the left side and 25 on the right side. So the lower and upper bounds are m_{25}' and m_{975}' respectively.).

If we want to compute the confidence intervals for other population statistics such as variance, standard deviation, 90th percentile, etc, then instead of computing the medians, we have to compute the respective statistics for each of the bootstrap samples.

Bootstrapping is a non parametric approach which doesn't make any assumptions about the distribution of the data used. If the bootstrap sample sizes are small, then the confidence intervals are wider. If the bootstrap samples are larger, then the confidence intervals are narrower.

How is Bootstrapping better over the other techniques?

The reason for choosing bootstrapping is that it is a very generic technique that is extremely powerful and makes no assumptions about the underlying distribution. With more and more data as we have nowadays and availability of computational resources, bootstrapping is a super powerful technique which is very popular in data science as compared to t-distribution.

Below is the code snippet that was discussed at the timestamp 13:24 in the video.

```
import numpy
from pandas import read_csv
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
from matplotlib import pyplot

# Load dataset
x = numpy.array([180,162,158,172,168,150,171,183,165,176])

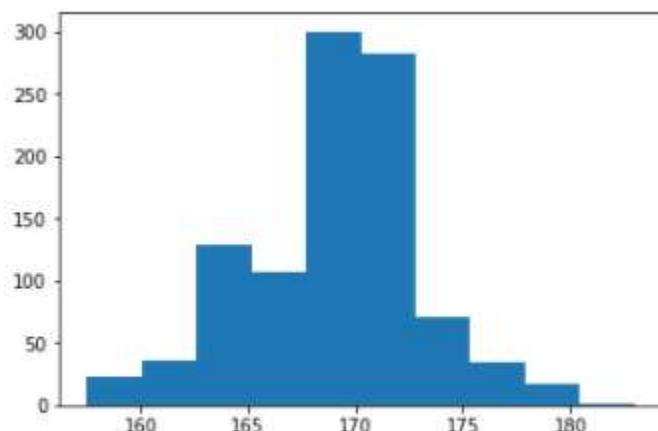
# configure bootstrap
n_iterations = 1000
n_size = int(len(x))

# run bootstrap
medians = list()
for i in range(n_iterations):
    # prepare train and test sets
    s = resample(x, n_samples=n_size);
    m = numpy.median(s);
    #print(m)
    medians.append(m)

# plot scores
pyplot.hist(medians)
pyplot.show()

# confidence intervals
alpha = 0.95
p = ((1.0-alpha)/2.0) * 100
lower = numpy.percentile(medians, p)

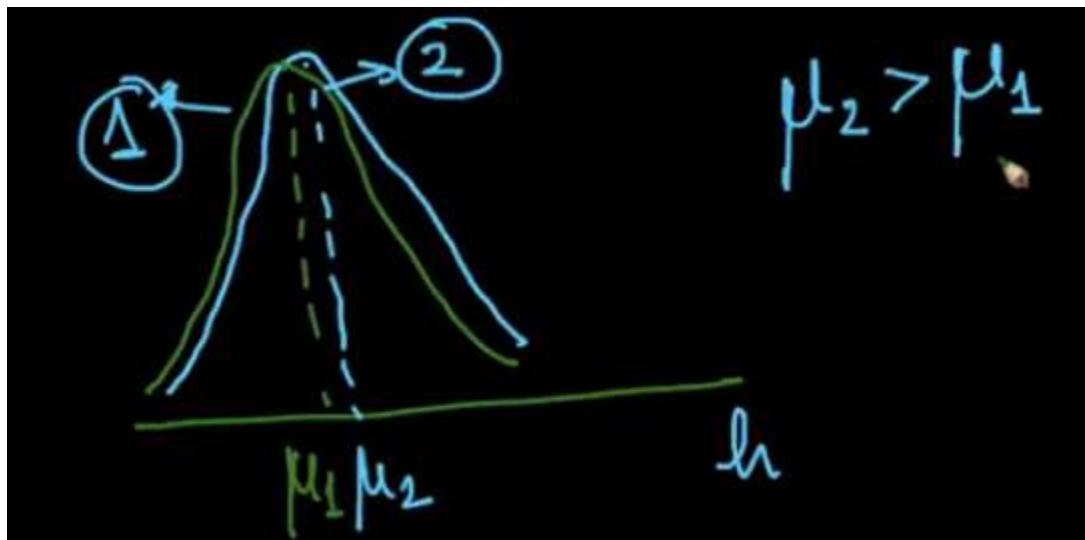
p = (alpha+((1.0-alpha)/2.0)) * 100
upper = numpy.percentile(medians, p)
print('%.1f confidence interval %.1f and %.1f' % (alpha*100, lower, upper))
```



95.0 confidence interval 161.5 and 176.0

22.24 Hypothesis Testing Methodology - Null Hypothesis, p-value

Let us assume we have two classes and we have the heights of the students in each class. Let us assume the distributions of the heights of both these classes are as shown below at the timestamp 2:43.



We can say that $\mu_2 > \mu_1$ from the plot, but how confidently can we say that height difference exists between class 1 and class 2 (Because if the scale of 'h' is large, then both the means look as if they are overlapping). We can test whether height difference exists or not using Hypothesis Testing.

Process for solving this problem

Step 1: Choosing a test statistic

Here we have to come up with a number that can say there exists a difference in the heights of two classes. This number is the test statistic and in this context, we choose the difference between the mean of the heights of classes 1 and 2, as the test statistic. (ie., $\mu_2 - \mu_1$).

$\mu_2 \rightarrow$ Mean of the heights of the sample from class '2'.

$\mu_1 \rightarrow$ Mean of the heights of the sample from class '1'.

If the value of $\mu_2 - \mu_1 = 0$, then we can say that there exists no difference in the heights of the students in the classes.

Note: If the populations of class 1 and class 2 are different, then we should pick samples of the same size from both the populations.

Step 2: Null Hypothesis (H_0)

This whole method follows proof by contradiction. Null Hypothesis is an assumption we are making for solving a problem and working on the problem towards it.

For this given problem, the null hypothesis is

H_0 : There is no difference between ' μ_1 ' and ' μ_2 '.

There is another concept called Alternative Hypothesis which is an inverse of Null Hypothesis. The Alternative Hypothesis is denoted as ' H_1 '.

For this given problem, the alternative hypothesis is

H_1 : There exists a difference between ' μ_1 ' and ' μ_2 '.

As this is a proof by contradiction, if we believe that null hypothesis is True, we have to prove that alternative hypothesis is False and should accept Null Hypothesis. (or) In case, if we are unable to prove that the null hypothesis is true, we have to accept the alternative hypothesis.

Step 3: Computing 'p' value (Computation part which is done using Resampling and Permutation test)

'p' value is the probability of observing a value for $(\mu_2 - \mu_1)$ if our null hypothesis is true. Let us assume the null hypothesis (H_0) is true.

- 1) Let us assume a value of 10cm for $\mu_2 - \mu_1$ (ie., $\mu_2 - \mu_1 = 10$)

If p-value = 0.9, it means the probability of $(\mu_2 - \mu_1 = 10)$ is 0.9, if our null hypothesis is true. It means, if there is not much difference in the heights of the means, we still see a difference of 10cm with 90% probability.

Here if the sample size is small, the difference between the mean heights may be large, because the samples might be biased sometimes. But if the samples are large in size, then the samples become more appropriate representatives of the population and both the sample means will be very much near to each other, than the sample means of smaller samples. This could reduce the differences in the sample means, when the sample sizes are large.

If the sample sizes are small, then we see huge variations in the sample means and whereas if the sample sizes are large, then we do not see much variation among the sample means (as both of them will be nearer to each other). So the difference between the samples is low.

So if the samples are larger the difference between the sample means will become almost nearer (or) equal to zero and in such cases, if p=0.9, then we can easily confirm that the null hypothesis is true. We then accept the null hypothesis.

- 2) Let us assume p-value = 0.05 which means there is 5% chance that $(\mu_2 - \mu_1 = 10)$, if the null hypothesis is true. So here if 'p' value is very low, then we say that the chances for the null hypothesis to be true are very low. In such cases, we have to reject the null hypothesis, and accept the alternative hypothesis.

22.25 Hypothesis Testing Intuition with coin toss example

Given a coin, we have to determine if the coin is biased towards the head or not. It means we have to determine if $P(H) > 0.5$

If a coin is not biased, then $P(H) = 0.5$. Let us perform an experiment of tossing a coin (say 5 times). Let the count of the number of heads be 'X'. (This is the test statistic)

So let us assume the occurrences of tossing a coin for 5 times be $\{H, H, H, H, H\}$.

So now $X = \text{Number of Heads} = 5$ (This is an observation from the experiment)

We shall now work on computing $P(X=5 | \text{coin is not biased towards the head})$ (ie., $P(X=5 | H_0)$).

Where $H_0 \rightarrow \text{Null Hypothesis}$ which says the given coin is not biased towards the head.

As it is given that the coin is not biased towards the head, $P(H) = \frac{1}{2}$

So $P(X=5 | H_0) = P(H) * P(H) * P(H) * P(H) * P(H) = (\frac{1}{2})^5 = 0.03 = 3\%$

It means there are 3% chances of getting 5 heads, if the coin is not biased towards the head.

So here p-value = $P(\text{observation by experiment} | \text{assumption})$

Here $P(X=5 | H_0) = 0.03$. It means the probability of the observation to be true, when the given assumption is true, is 0.03 (ie. 3%)

(In general, 5% is the rule of thumb. If the percentage of p-value is $< 5\%$, then we reject the null hypothesis). Here in this case, the null hypothesis is "Coin is not biased towards the head". So we reject it. The alternative hypothesis(H_1) in this case is, "The coin is biased towards the head" and we accept it.

For example, if we reduce the number of tosses to 3, then

$P(X=3 | H_0) = (\frac{1}{2})^3 = 12.5\%$

Here as $12.5\% > 5\%$, we accept the null hypothesis. Here changing the number of tosses has changed the result completely.

Note: If the 'p' value is less than 0.05 (say), then we can't reject the observation as it has already been observed. In such cases, we can only reject the null hypothesis.

While performing Hypothesis Testing, we should carefully

- 1) Choose the sample sizes (in this experiment, the sample size is the number of coin tosses).
- 2) Define the null hypothesis, in such a way that $P(\text{observation} | \text{assumption})$ is easy and feasible to compute.
- 3) Choose the test statistic.

22.26 Resampling and Permutation Test

Let us consider the example of the heights of students in classes 1 and 2(which was the first example discussed in Hypothesis Testing). Let us assume, we have picked samples of size 50(say) from each of these classes and have computed the means for both the samples, and let them be denoted as ' μ_1 ' and ' μ_2 ' respectively. Initially let the difference between these two sample means be denoted by ' Δ '.

$$\Delta = \mu_2 - \mu_1$$

Now let us combine all the elements present in both these samples into a single set, and then pick two random samples(these samples should be of the same size 50 throughout). Let us compute the means for the new randomly picked samples and compute the difference between them. Let this difference be denoted as ' δ_1 '.

We shall repeat this process for a certain number of times(say 10000) and obtain all the values of the sample mean differences. Let them be denoted as ' δ_1 ', ' δ_2 ', ' δ_3 ',.....' δ_{10k} '.

We should now sort all these sample mean differences and let these sample means in the sorted order be denoted as ' δ'_1 ', ' δ'_2 ', ' δ'_3 ',.....' δ'_{10k} '. Now from the increasing order of these sample means, we have to check what percentage of the values are greater than ' Δ '. (ie., $\Delta = \mu_2 - \mu_1$)

Let's say 5% of the values are greater than ' Δ ', then

$$P\text{-value} = 5/100 = 0.05$$

If $x\%$ of the values are greater than ' Δ ', then

$$P\text{-value} = x/100$$

In the olden days, the statisticians assumed that the sample mean differences follow gaussian distribution.

$$\delta_i \sim N(0,1)$$

So if $\Delta = 2$, then $P(\delta_i \geq 2) = 0.025$ (It is because 2.5% of the points lie to the right side of 2σ on the 'X' axis for a normal distribution). So p-value = 0.025.

Note: The notes for the video lecture 22.27 is ignored as the same examples were discussed in this course.

22.28 K-S Test for Similarity of Two Distributions

Let us assume we have two random variables ' X_1 ' and ' X_2 ' and we have to pick samples from them of sizes ' n_1 ' and ' n_2 ' respectively.

Sample of $X_1 = \{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_{n_1}^{(1)}\}$

Sample of $X_2 = \{x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \dots, x_{n_2}^{(2)}\}$

We have to check if the distributions of ' X_1 ' and ' X_2 ' are the same or not. For any distribution, let us say the population mean is ' μ ' and the population standard deviation is ' σ ', then if we standardize the distribution and if the standardized form is $N(0,1)$, then we can say that the given population follows gaussian distribution.

Objective: To determine if both ' X_1 ' and ' X_2 ' have the same distribution.

' X_1 ' has ' n_1 ' observations and ' X_2 ' has ' n_2 ' observations. Let us now plot the CDF of ' X_1 ' and ' X_2 '.

For this problem, the null hypothesis and the alternative hypothesis would be
H0: ' X_1 ' and ' X_2 ' have the same distribution.

H1: ' X_1 ' and ' X_2 ' don't have the same distribution.

If we pick any point on the 'X' axis, we find the differences in the CDF values. In case, if the given two random variables follow the same distribution, then the CDFs of both the plots will overlap. When the CDFs overlap completely, then at any point on the 'X' axis, the difference in the CDF is zero.

If both ' X_1 ' and ' X_2 ' follow the same distribution, then

- If both the samples are of smaller size(ie., ' n_1 ' and ' n_2 ' are small), then we see the differences in the CDFs, even if both the random variables follow the same distribution.
- If both the sample sizes are large(ie., ' n_1 ' and ' n_2 ' are large), then if both the random variables follow the same distribution, the more the CDFs would overlap.

So the test statistic in this context is

$D_{n_1, n_2} = \text{Sup}_x |F_{1, n_1}(x) - F_{2, n_2}(x)|$ which is pronounced as "Supremum over all x's of " $F_{1, n_1}(x) - F_{2, n_2}(x)$ ". Here supremum means the maximum value.

$F_{1, n_1}(x) \rightarrow$ CDF of random variable ' X_1 ' over ' n_1 ' points.

$F_{2, n_2}(x) \rightarrow$ CDF of random variable ' X_2 ' over ' n_2 ' points.

If both ' n_1 ' and ' n_2 ' are large and ' X_1 ' and ' X_2 ' come from the same distribution, then the difference between the CDFs at any point is zero, and finally the value of $D_{n_1, n_2} = 0$. (We can accept the null hypothesis).

After a thorough research, it is found that we can reject the null hypothesis at a level ' α ' if $D_{n_1, n_2} > c(\alpha) * \sqrt{(n_1 + n_2) / (n_1 * n_2)}$

Where $n_1, n_2 \rightarrow$ sample sizes of ' X_1 ' and ' X_2 ' respectively.

$\alpha \rightarrow$ 'p' value

22.29 p-value for K-S Test

So far in hypothesis testing, we have learnt to obtain the 'p' value first and then decide whether to accept or reject the null hypothesis. In Hypothesis testing, we reject the null hypothesis at a significant level 'α', if the p-value < α.

But here in KS test, we reject the null hypothesis at a significance level 'α', if

$$D_{n_1, n_2} > c(\alpha) * \sqrt{((n_1 + n_2) / (n_1 * n_2))}$$

The value of $c(\alpha) = \sqrt{-\frac{1}{2}} * \ln(\alpha/2)$

So for the time being we shall denote D_{n_1, n_2} as 'D'. So now the above condition becomes

$$D > c(\alpha) * \sqrt{((n_1 + n_2) / (n_1 * n_2))}$$

We are now substituting the value of 'c(α)' here and then it becomes

$$D > \sqrt{-\frac{1}{2}} * \ln(\alpha/2) * \sqrt{((n_1 + n_2) / (n_1 * n_2))}$$

$$D * \sqrt{((n_1 * n_2) / (n_1 + n_2))} > \sqrt{(-\frac{1}{2}) * \ln(\alpha/2)}$$

Applying square on both the sides, then it becomes

$$D^2 * ((n_1 * n_2) / (n_1 + n_2)) > (-\frac{1}{2}) * \ln(\alpha/2)$$

Multiplying with a minus on both sides.

$$-D^2 * ((n_1 * n_2) / (n_1 + n_2)) < (\frac{1}{2}) * \ln(\alpha/2)$$

Multiplying with '2' on both sides.

$$-2*D^2 * ((n_1 * n_2) / (n_1 + n_2)) < (\ln(\alpha/2))$$

In order to remove the logarithm on the right hand side, we have to apply exponential function on both sides/

$$\exp(-2*D^2 * ((n_1 * n_2) / (n_1 + n_2))) < (\alpha/2)$$

Multiplying by 2 on both sides, so that the denominator in the right hand side gets cancelled.

$$2*\exp(-2*D^2 * ((n_1 * n_2) / (n_1 + n_2))) < \alpha$$

When we compare this to the hypothesis testing (where null hypothesis gets rejected if $p < \alpha$), if we apply the same logic here, then we can say that

$$\text{P-value} = 2*\exp(-2*D^2 * ((n_1 * n_2) / (n_1 + n_2)))$$

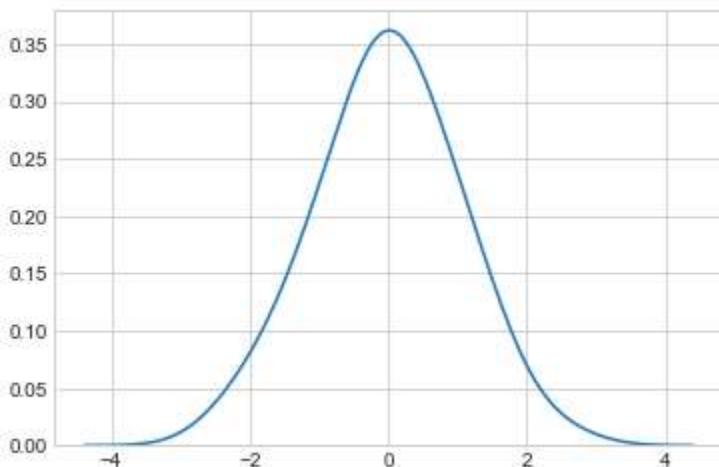
22.30 Code Snippet - KS Test

So far we have learnt the theoretical concept of KS test and now we shall look at the code part.

```
import numpy as np
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt

#generate a gaussian r.v X
x = stats.norm.rvs(size=1000);
sns.set_style('whitegrid')
sns.kdeplot(np.array(x), bw=0.5)
plt.show()
```

In the above code snippet, we are first importing the libraries and then creating a random variable ‘x’ that follows normal distribution. After that, we are plotting its PDF. The plot looks as below



Let us now perform the KS test using the below code snippet.

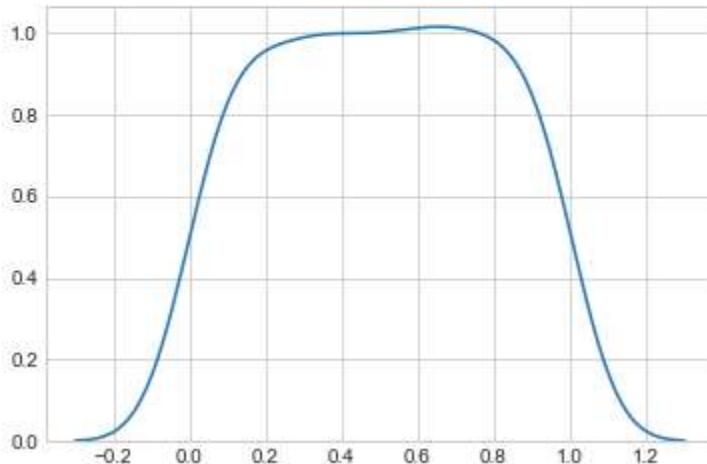
```
stats.kstest(x, 'norm')
```

```
KstestResult(statistic=0.021308397286061931, pvalue=0.75424031453335627)
```

After the execution of the above statement, in the result, we get two parameters. One is the ‘p’ value and the other is the ‘statistic’ parameter which indicates the ‘ $D_{n1,n2}$ ’ value. Here we accept the null hypothesis.

Now we shall perform the same KS test for a uniform distributed random variable.

```
# Y ~ Continuous Uniform Distribution(0,1)
y = np.random.uniform(0,1,10000);
sns.kdeplot(np.array(y), bw=0.1)
plt.show()
```



Here Using the above code snippet, we are creating a continuous uniform random variable and are plotting its PDF. Let us now perform the KS test.

```
stats.kstest(y, 'norm')

KstestResult(statistic=0.50159644397739489, pvalue=0.0)
```

Here we get the ‘p’ value and the statistic as the result. ‘P’ value of 0.0 indicates that the two distributions are not the same and we can reject the null hypothesis.

22.31 Hypothesis Testing - Another Example

Let us assume we have two cities 'C₁' and 'C₂'. Our task here is to determine if the population means of heights of people living in these two cities is same or not.

Let us now pick two random samples(one from each city) of size(say 50) and let them be denoted as 'C₁' and 'C₂'.

$$C_1 = [h_1, h_2, h_3, \dots, h_{50}]$$

$$C_2 = [h'_1, h'_2, h'_3, \dots, h'_{50}]$$

Let ' μ_1 ' and ' μ_2 ' be the sample means heights of the people living in the cities 'C₁' and 'C₂' respectively. For example, we have observed the values as $\mu_1 = 162\text{cm}$ and $\mu_2 = 167\text{cm}$. So the difference between these sample means = $\mu_2 - \mu_1 = 167 - 162 = 5\text{cm}$.

So the test statistic here would be $X = \mu_2 - \mu_1$

Null Hypothesis(H₀): There is no difference between the population means.

Now we have to compute the value of $P(X=5\text{cm}|H_0)$ which is the probability of observing a difference of 5cm in the sample means of sample size 50, if the null hypothesis is true.

Case 1:

If $P(X=5\text{cm}|H_0) = 0.2$ (say), it means there is a 20% chance of observing a difference of 5cm in the sample mean heights of 'C₁' and 'C₂' with a sample of 50, if there is no population mean difference. As this probability is true, we can confirm that our assumption is true and we accept the null hypothesis.

Case 2:

If $P(X=5\text{cm}|H_0) = 0.03$ (say), it means there is a 3% chance of observing a difference of 5cm in the sample mean heights of 'C₁' and 'C₂' with a sample of 50, if there is no population mean difference. This implies our assumption is wrong and we reject the null hypothesis and accept the alternative hypothesis.

22.32 How to use Hypothesis Testing?

Application 1

KS Test is one of the best applications where hypothesis testing is used to check if two random variables follow the same distribution or not.

Application 2 (Designing medicine/drugs)

Let us assume we have two drudges 'D₁' and 'D₂' for curing a disease. 'D₁' is manufactured by company 'C₁' and is in use whereas 'D₂' is recently manufactured by a company 'C₂'.

Let's say 'D₁' cures the disease in 4 hours and the company 'C₂' claims that 'D₂' can cure the same disease faster than 'D₁'.

Claim by 'C₂': 'D₂' cures the disease faster than 'D₁'.

Task/Experiment: To determine if the claim is true or not.

In order to conduct this experiment, let's pick a sample of 100 patients and divide them into 2 groups. They are set 'S₁' and set 'S₂' with a size of 50 patients each.

Let the patients in 'S₁' be given the drug 'D₁' and the patients in 'S₂' be given the drug 'D₂'. Let the values in the sets be filled with the duration taken by the patients to recover.

'μ₁' → mean time of 'S₁' (say 4 hours)

'μ₂' → mean time of 'S₂' (say 2 hours)

Here we got these mean values for samples of smaller sizes. If the sample sizes are large, then these values will change. So instead of going with these sample means, we can go for hypothesis testing.

Null Hypothesis (H₀): 'D₁' and 'D₂' are not different. (ie., they both take the same time)

Test statistic(X) = (μ₂-μ₁)

μ₂-μ₁ = 2 hours

if P(X>=2|H₀) = 0.01 (say it is small), then we reject the null hypothesis and accept alternative hypothesis.

Significance Level (α):

We have to be really sure when we reject the null hypothesis. If the case is really critical like in healthcare, we have to choose α = 1%.

If it is something like an e-commerce domain problem, then we have to choose α = 3%. But typically 5% of 'α' is used in most of the cases.

Note

One best example of hypothesis testing in e-commerce is usage of credit cards for purchasing.

Sample 1 (S_1): Customers who have a Visa Credit Card.

Sample 2 (S_2): Customers who have a Mastercard Credit Card.

Let ' μ_1 ' be the mean of the sample ' S_1 ' and ' μ_2 ' be the mean of the sample ' S_2 '.

Visa claims that its customers make more purchases than Mastercard.

Null Hypothesis (H_0): $(\mu_2 - \mu_1)$ is large.

Test Statistic: $(\mu_2 - \mu_1)$

22.33 Proportional Sampling

It is one of the important topics in machine learning. Let us assume we are given an array of values. Let the array be $d = [2.0, 6.0, 1.2, 5.8, 20.0]$ and let these numerical values be denoted as d_1, d_2, d_3, d_4 and d_5 respectively.

The task of proportional sampling is to pick an element among the given ‘n’ elements such that the probability of picking an element is proportional to the d_i ’s.

If we randomly pick a value, then all these values are equally probable. When we look at the values in the array, the value of ‘ d_5 ’ is the highest and the value of ‘ d_3 ’ is the least. So in case of proportional sampling, the probability of picking ‘ d_5 ’ should be the highest and the probability of picking ‘ d_3 ’ should be the least.

Procedure for Proportional Sampling

Step 1

- Compute the sum of all the values of the array.

$$S = \sum_{i=1}^n d_i \quad (\text{For the given values in the array, } S = 35)$$

- Normalize all the values by dividing them by the sum.

$$d_i' = d_i/S$$

$$d_1' = d_1/S, d_2' = d_2/S, d_3' = d_3/S, d_4' = d_4/S, d_5' = d_5/S$$

So after normalization, the values are

$$d_1' = 0.0571, d_2' = 0.171428, d_3' = 0.034, d_4' = 0.165, d_5' = 0.571$$

All the above normalized values lie in $[0,1]$ and also all these values sum to 1.

(ie., $\sum_{i=1}^n d_i' = 1$)

- Compute Cumulative Normalized Sum

$$d_1_tilde = d_1' = 0.0571$$

$$d_2_tilde = d_1_tilde + d_2' = 0.228528$$

$$d_3_tilde = d_2_tilde + d_3' = 0.262828$$

$$d_4_tilde = d_3_tilde + d_4' = 0.428528$$

$$d_5_tilde = d_4_tilde + d_5' = 1.00$$

Step 2

We have to sample one value from a uniform random variable $U(0,1)$. Let this value be denoted as ‘r’.

`r = numpy.random.uniform(0.0, 1.0, 1)`

Step 3

Proportional Sampling happens now.

If $r \leq d_1_tilde$, then **return 1**

If $r \leq d_2_tilde$, then **return 2**

If $r \leq d_3_tilde$, then **return 3**

If $r \leq d_4_{\text{tilda}}$, then return 4

If $r \leq d_5_{\text{tilda}}$, then return 5

Now

$d_1_{\text{tilda}} < d_2^{\dagger} < d_2_{\text{tilda}}$ (because $d_2_{\text{tilda}} = d_1_{\text{tilda}} + d_2^{\dagger}$)

$d_2_{\text{tilda}} < d_3^{\dagger} < d_3_{\text{tilda}}$ (because $d_3_{\text{tilda}} = d_2_{\text{tilda}} + d_3^{\dagger}$)

$d_3_{\text{tilda}} < d_4^{\dagger} < d_4_{\text{tilda}}$ (because $d_4_{\text{tilda}} = d_3_{\text{tilda}} + d_4^{\dagger}$)

$d_4_{\text{tilda}} < d_5^{\dagger} < d_5_{\text{tilda}}$ (because $d_5_{\text{tilda}} = d_4_{\text{tilda}} + d_5^{\dagger}$)

The probability of picking '4' = probability of 'r' lying between ' d_3_{tilda} ' and ' d_4_{tilda} ' = d_4^{\dagger}

But $d_4^{\dagger} \propto d_4$ (because $d_4^{\dagger} = d_4/5$). So

The probability of picking '4' in $U(0,1) = d_4$

The probability of picking '3' in $U(0,1) = d_3$

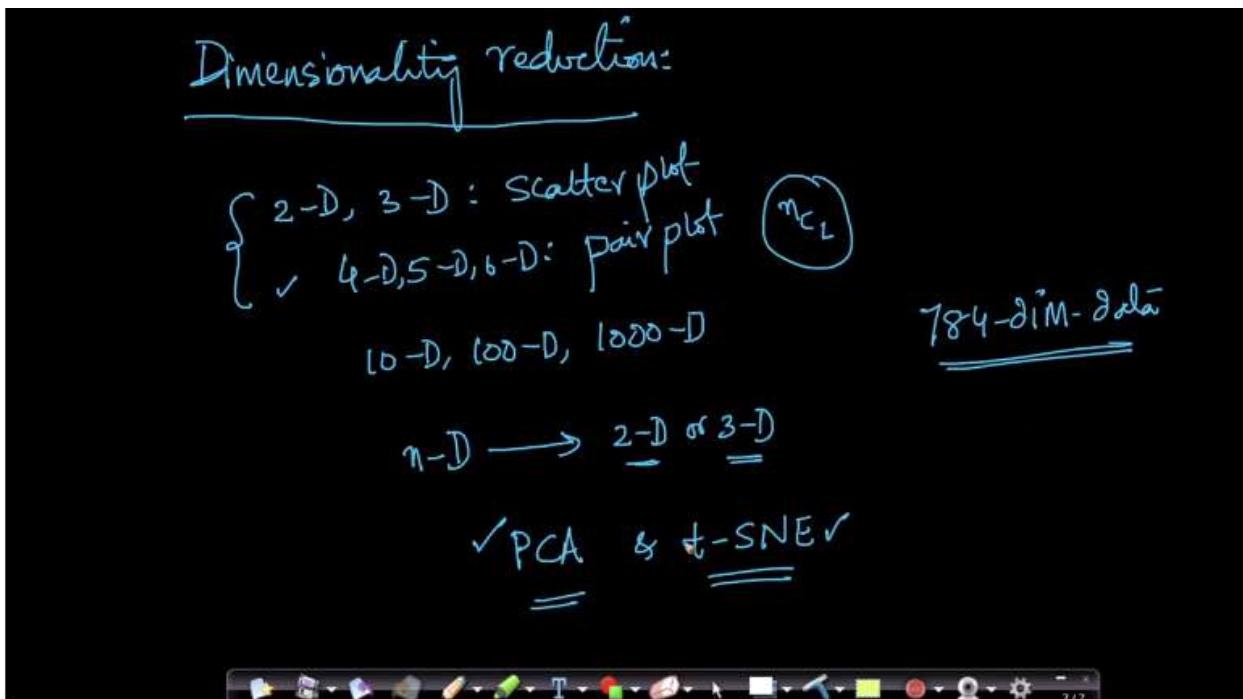
The probability of picking '2' in $U(0,1) = d_2$

The probability of picking '1' in $U(0,1) = d_1$

The probability of picking '5' in $U(0,1) = d_0$

Higher weightage is given to the larger values in proportion sampling.

25.1 What is Dimensionality Reduction?



Timestamp: 2:11

We have seen that 2-D and 3-D data can be visualized using scatter plots, but as the number of dimensions increases we need $nC2$ number of pair plots, which can be difficult to visualize. Hence we use dimensionality reduction techniques like PCA and t-SNE using which we can reduce the dimensionality of the data from n-D to 2-D and 3-D, which we can visualize easily.

25.2 Row vector and Column vector?

Row-vector & column-vector

flower: $\underbrace{[SL, PL, SW, PW]}_{\text{real-values}}$

\mathbb{R} : real Space

ith point: $x_i \in \mathbb{R}^d \rightarrow d\text{-dim. columnvector}$

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ \vdots \\ x_{id} \end{bmatrix}_{d \times 1} : \text{column-vector}$$

$$f_1 = \begin{bmatrix} 2.1 \\ 3.2 \\ 1.6 \\ 4.2 \end{bmatrix}$$

Column-vector

Timestamp: 4:06

An ith datapoint x_i is often represented either as a column vector or a row vector. If not mentioned explicitly then it is assumed to be a column vector. A column vector looks like as mentioned in the above figure. A d dimensional vector that takes real values is often mentioned as x_i belonging to \mathbb{R}^d where \mathbb{R} stands for real-values.

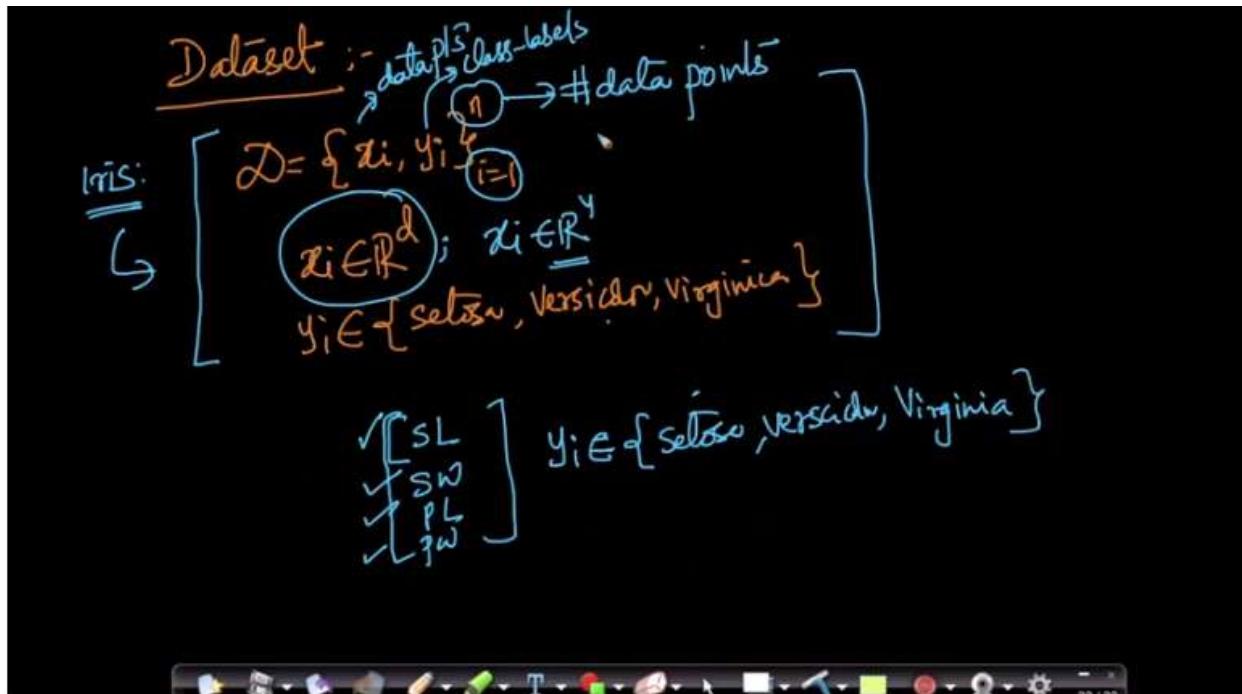
A row vector representation of x_i looks as below.

$$\vec{x}_i = \begin{bmatrix} \downarrow & \downarrow & \downarrow & \downarrow \\ 2 & 1 & 3 & 2 & 4 & 6 & 1 & 2 \end{bmatrix}_{1 \times 8} : \text{row-vector}$$

Q

Timestamp: 3:30

25.3 How to represent a data set?



Timestamp: 3:09

There are many ways to represent a dataset but one such way to represent a dataset for the example of iris dataset is as above. A dataset is represented as $D = \{x_i, y_i\}$ from 1 to n, where x_i represents the datapoint and y_i represent the corresponding label.

x_i is represented as belonging to \mathbb{R}^4 as it contains 4 real valued features and y_i can take any value from $\{\text{setosa}, \text{versicolor}, \text{virginica}\}$ the three classes or labels in the iris dataset.

25.4 How to represent a data set as a Matrix?

Dataset as a data-matrix:

$$X = \begin{bmatrix} f_1 & f_2 & f_3 & \dots & f_j & \dots & f_d \end{bmatrix}_{n \times d}$$

x_i^T

each datapoint: row
each column: feature

$\mathcal{D} = \{x_i, y_i\}_{i=1}^n$

$x_i \in \mathbb{R}^d$
 $y_i \in \{s, v_i, v_e\}$

→ d-features
→ column-vector

$\begin{cases} x_i \text{ is a col-vector} \\ x_i^T \text{ is a row-vector} \end{cases}$

Timestamp: 2:09

There are two ways to represent our data matrix X , the first way is to represent it in such a way that each row represents a datapoint and each column represents a feature. Notice that since x_i is a column vector, in order for us to represent it as a row vector, we are using x_i^T .

$$X = \begin{bmatrix} f_1 & f_2 & f_3 & \dots & f_i & \dots & f_n \\ x_1 & x_2 & x_3 & \dots & x_i & \dots & x_n \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ f_d & f_d & f_d & \dots & f_d & \dots & f_d \end{bmatrix}_{d \times n}$$

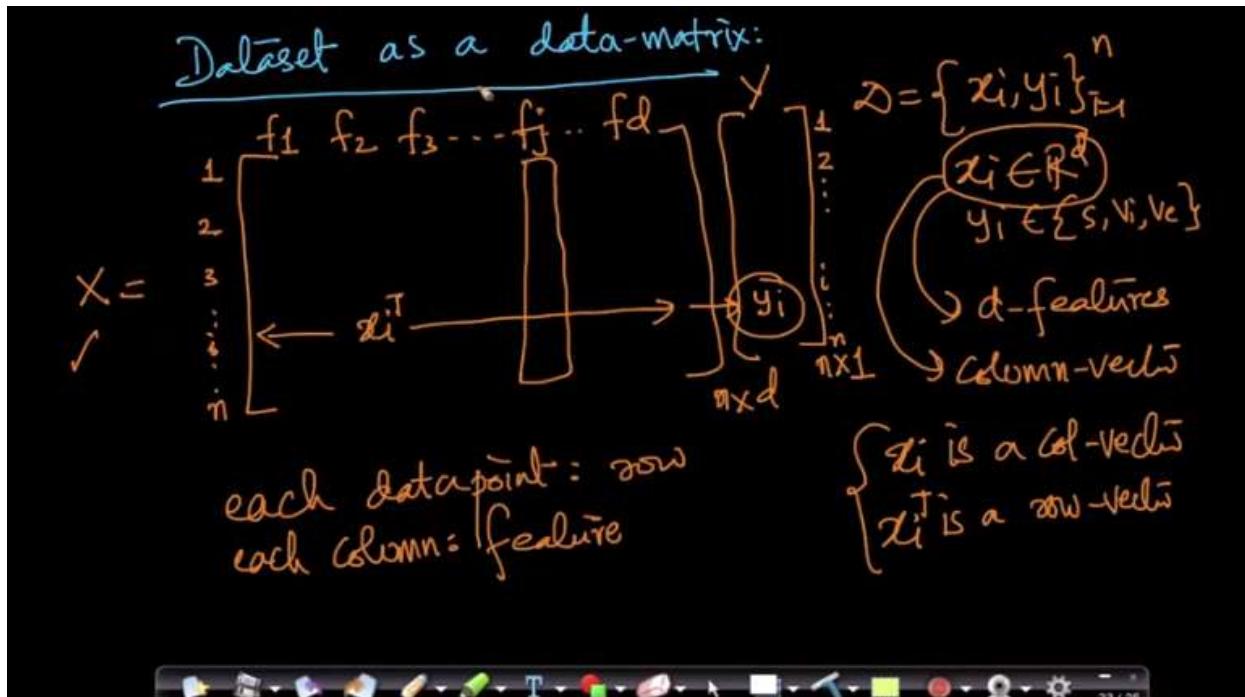
column: data-point
row: feature variable

$$\begin{aligned} f_1 &= PL \\ f_2 &= PW \\ f_3 &= SL \\ f_4 &= SW \end{aligned}$$

Timestamp: 3:56

The other way is to just represent it as above, where each column represents a data point and each row represents a feature. This data matrix X is just a transpose of the data matrix X in the first approach.

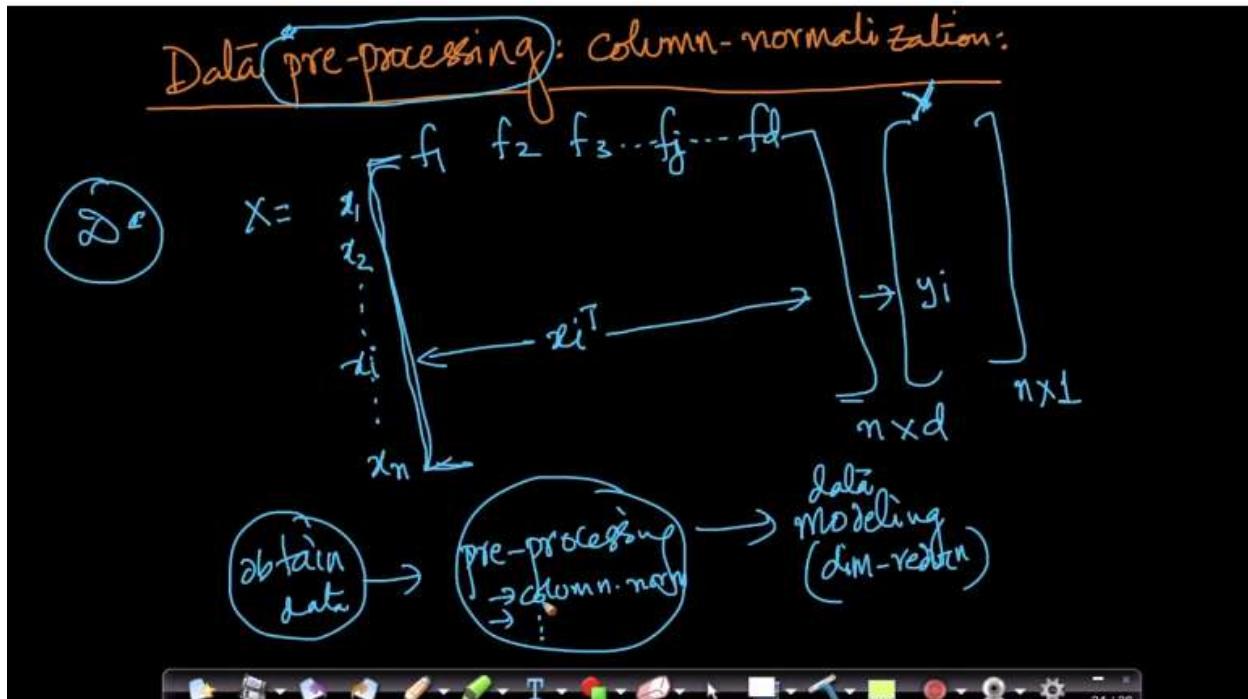
The most common approach is the first one, where each row represents a datapoint and each column represents a feature. The entire dataset now looks like below using the first approach.



Timestamp: 6:35

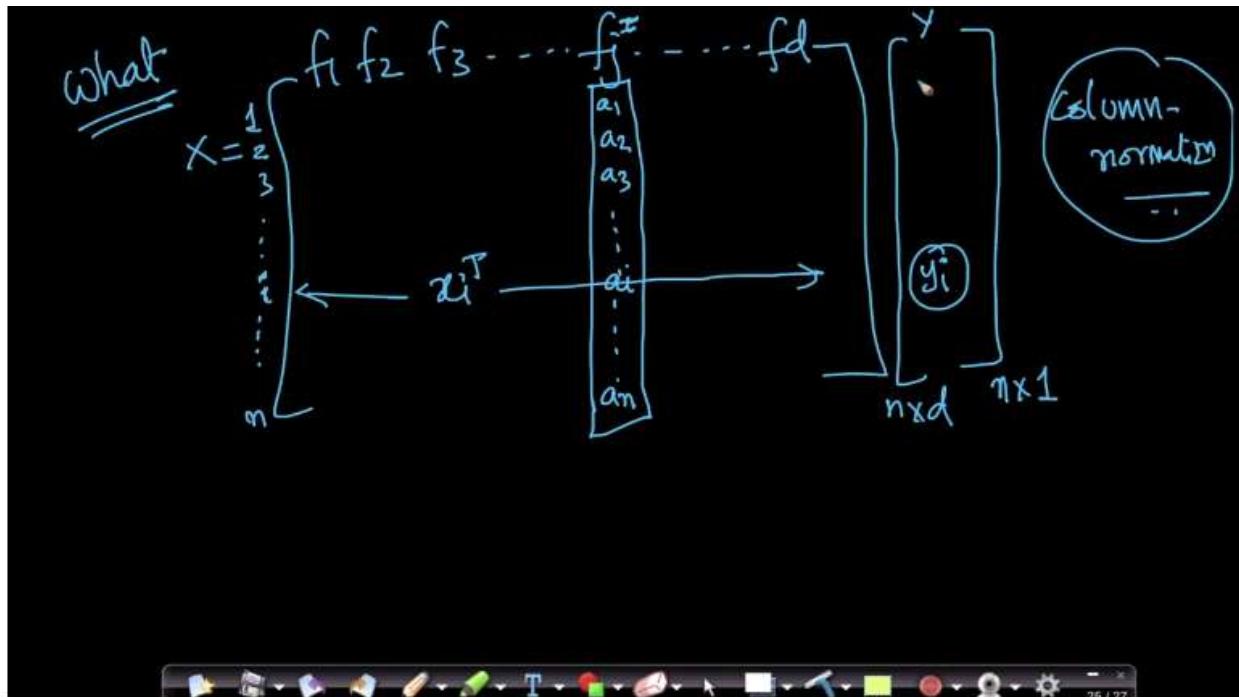
Notice that Y is represented as a column matrix where each row y_i represents the label for the corresponding datapoint x_i in X .

25.5 Data Preprocessing: Feature Normalization



Timestamp: 3:10

After obtaining the data, we need to preprocessing of the data such that our dimensionality reduction algorithms can work better. One such preprocessing is column normalization.



Timestamp: 5:26

For column normalization ,we pick each feature f_j , lets say it contains numbers a_1, a_2, \dots, a_n as above then we do column normalization as follows.

column: $a_1, a_2, \dots, a_i, \dots, a_n$ → n-values of f_j

$\max(a_i) = a_{\max}$

$\min(a_i) = a_{\min}$

$a'_1, a'_2, a'_3, a'_4, \dots, a'_i, \dots, a'_n$

$a'_i = \frac{(a_i - a_{\min})}{(a_{\max} - a_{\min})}$

$\frac{1}{a_{\max} - a_{\min}}$

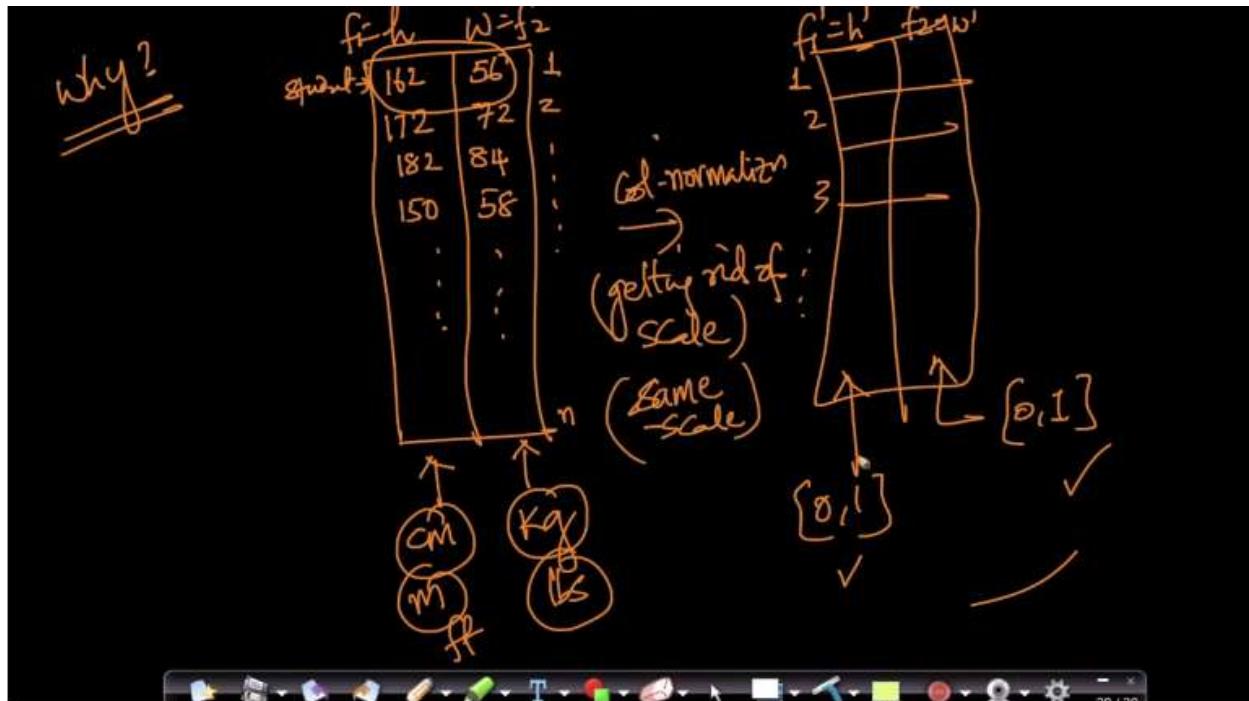
$\frac{(a_{\min} - a_{\min})}{a_{\max} - a_{\min}} = 0$

$a'_{\max} = \frac{a_{\max} - a_{\min}}{a_{\max} - a_{\min}} = 1$

Timestamp: 9:30

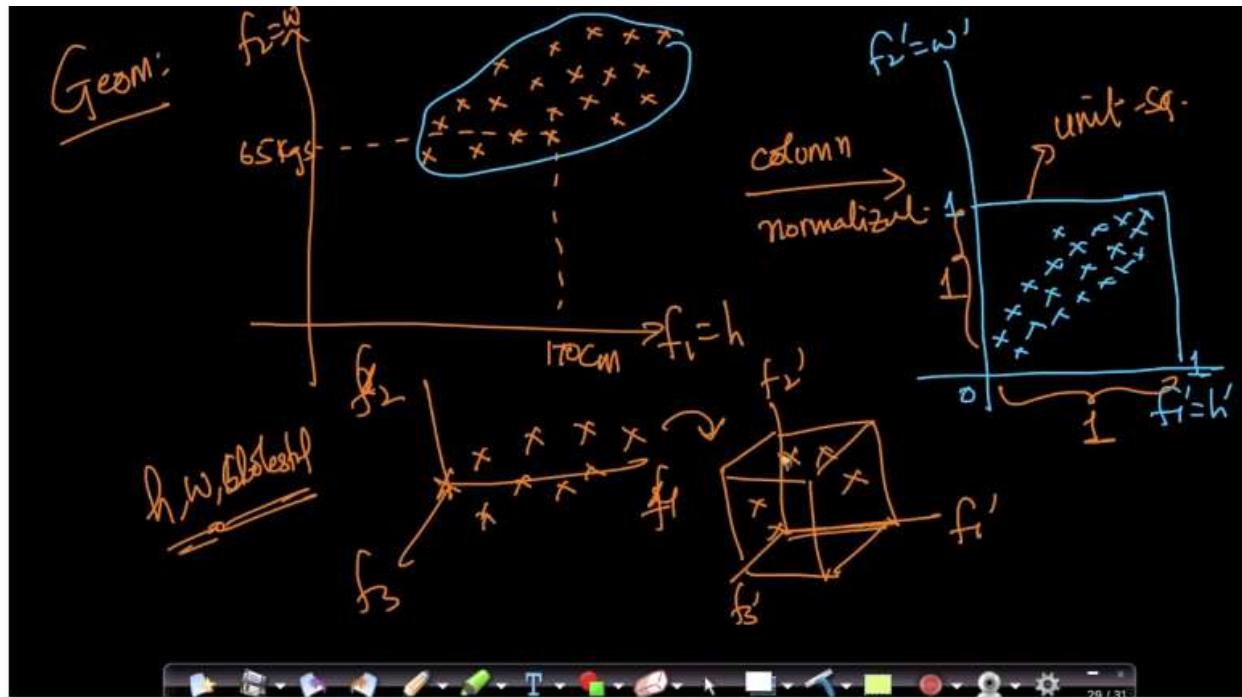
As shown in the above figure, with column normalization what we are doing is, from a_1, a_2, \dots, a_n we are getting a'_1, a'_2, \dots, a'_n as below, such that each a'_i now belongs to $[0,1]$

$$a'_i = (a_i - a_{\min}) / (a_{\max} - a_{\min})$$



Timestamp: 14:32

Since features like height, weight etc can be measured in different scales such as cm,m,etc, we are doing column normalization to get rid of the problems that come up with the scale(which will be discussed later) and we are converting all the features into the same scale using column normalization.



Timestamp: 17:45

Geometrically, through column normalization, we are squishing data points that are spread over the entire space into a unit-square when it is 2-D data or into unit-cube when it is 3-D data and n-D unit hyper cube when the data is of n-D data.

25.6 Mean of a Data Matrix

$x_1 = \begin{bmatrix} f_1 \\ 2 \cdot 2 \\ 4 \cdot 2 \end{bmatrix} \in \mathbb{R}^3$
 $x_2 = \begin{bmatrix} 1 \cdot 2 \\ 3 \cdot 2 \end{bmatrix} \in \mathbb{R}^2$

$(x_1 + x_2) = \begin{bmatrix} 3 \cdot 4 \\ 7 \cdot 4 \end{bmatrix}$

$\bar{x} \in \mathbb{R}^d$
 $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$

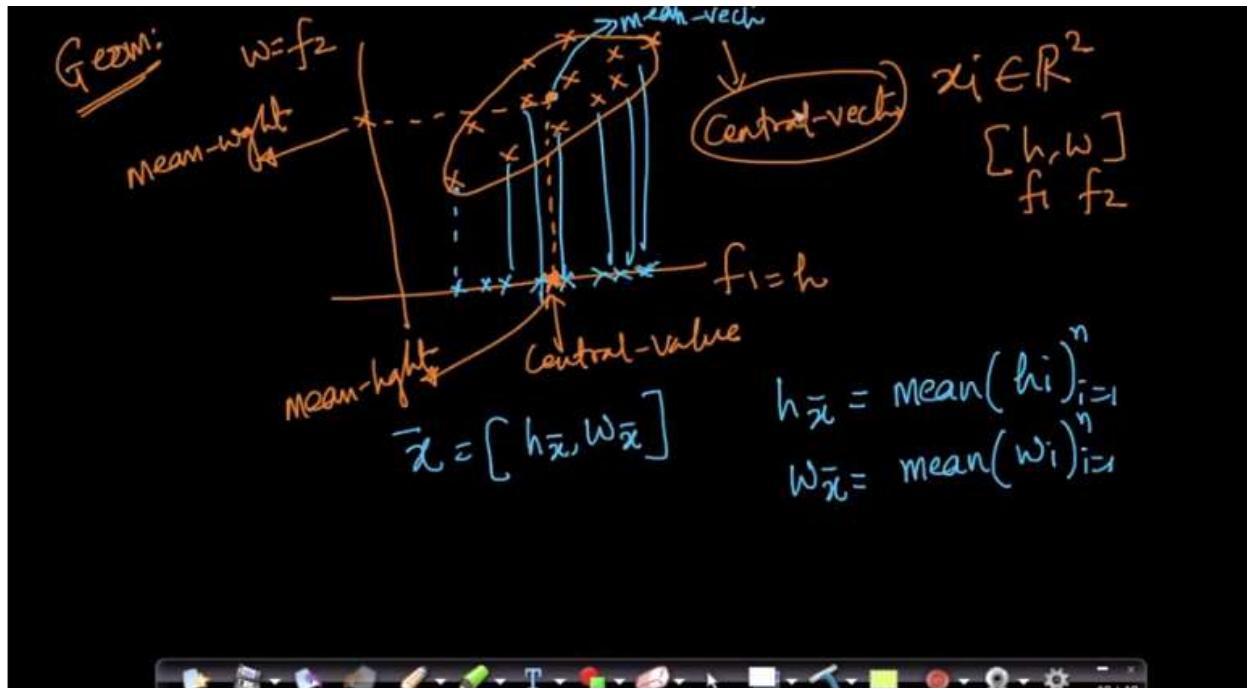
$x_i \in \mathbb{R}^d$
 $= \frac{1}{n} (x_1 + x_2 + \dots + x_n)$

Mean vector

Timestamp: 2:31

As shown in the above picture, the sum of two vectors x_1 and x_2 is just a vector where each element is the sum of corresponding elements in the vectors x_1 and x_2 .

The mean vector of n data points, is the sum of n datapoints divided by n as shown above.



Timestamp: 5:47

Geometrically, mean vector is like a central value of the data.

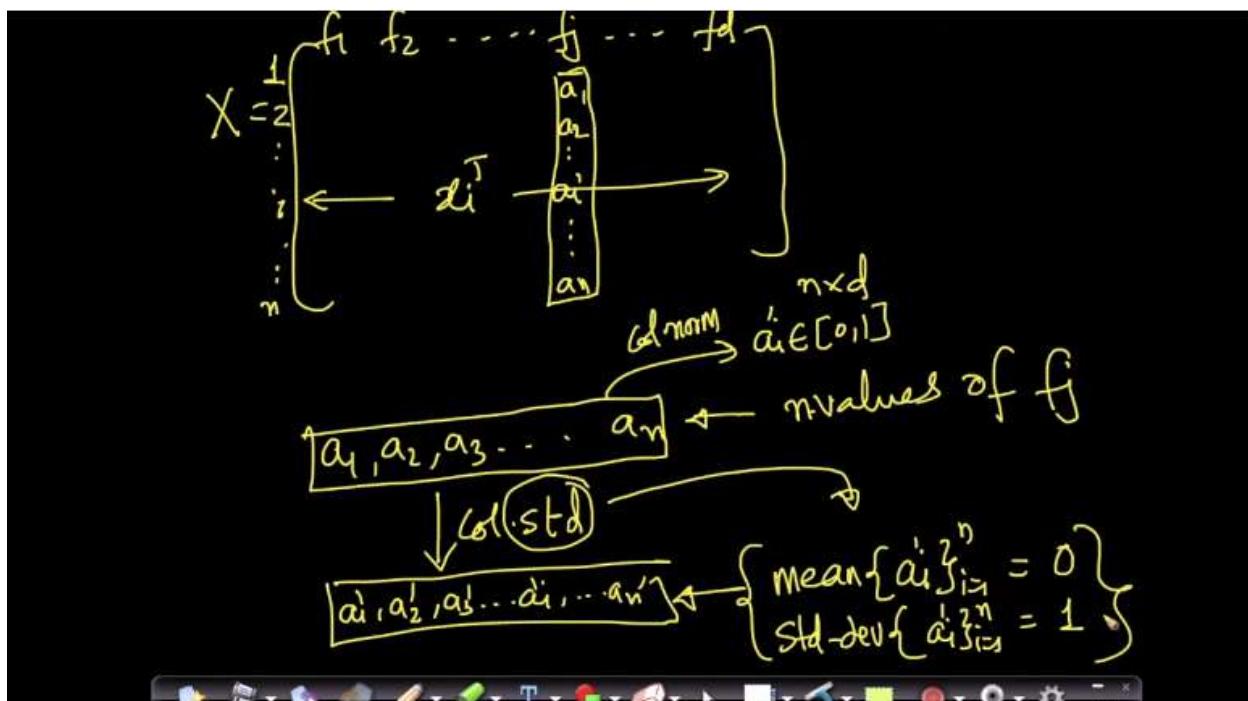
25.7 Data Preprocessing: Column Standardization

Data-preprocessing: Column-standardization

✓ Column-normalization :- $[0, 1]$ \leftarrow get rid of scales of each feature
 \hookrightarrow unit hyper-cube
Col. Std :- more often used in practice

Timestamp: 1:11

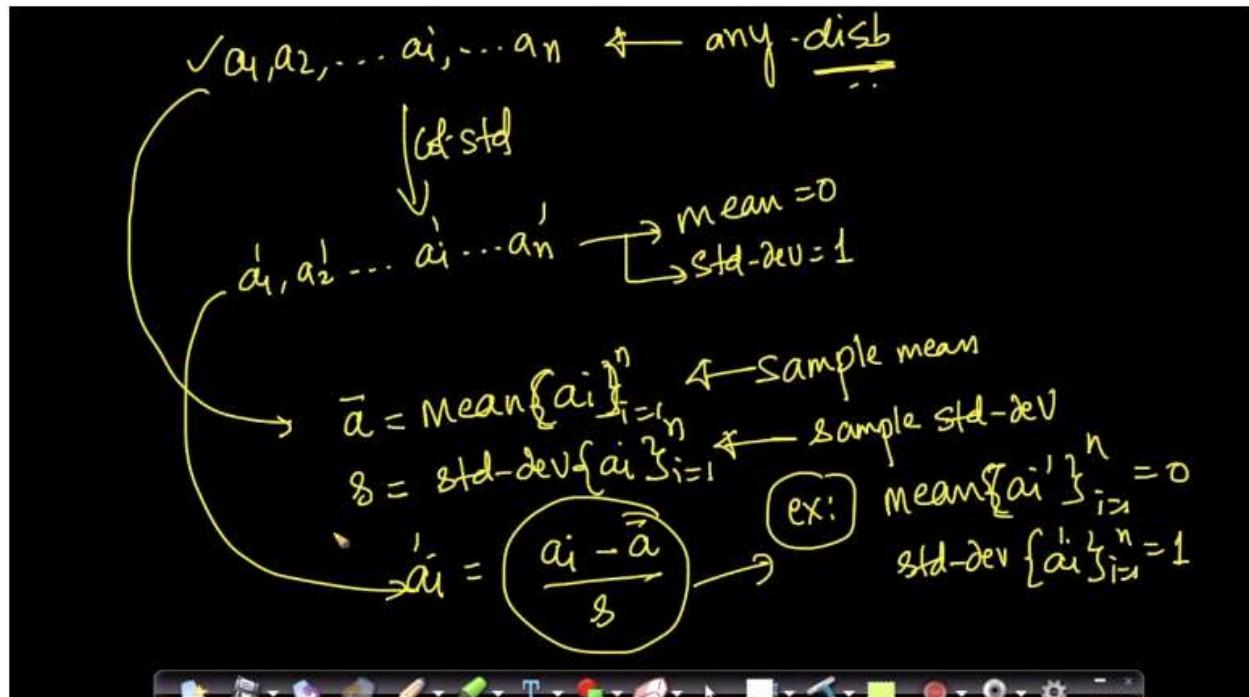
Column standardization is a similar preprocessing technique similar to column normalization and it is used more often in practice.



Timestamp: 4:26

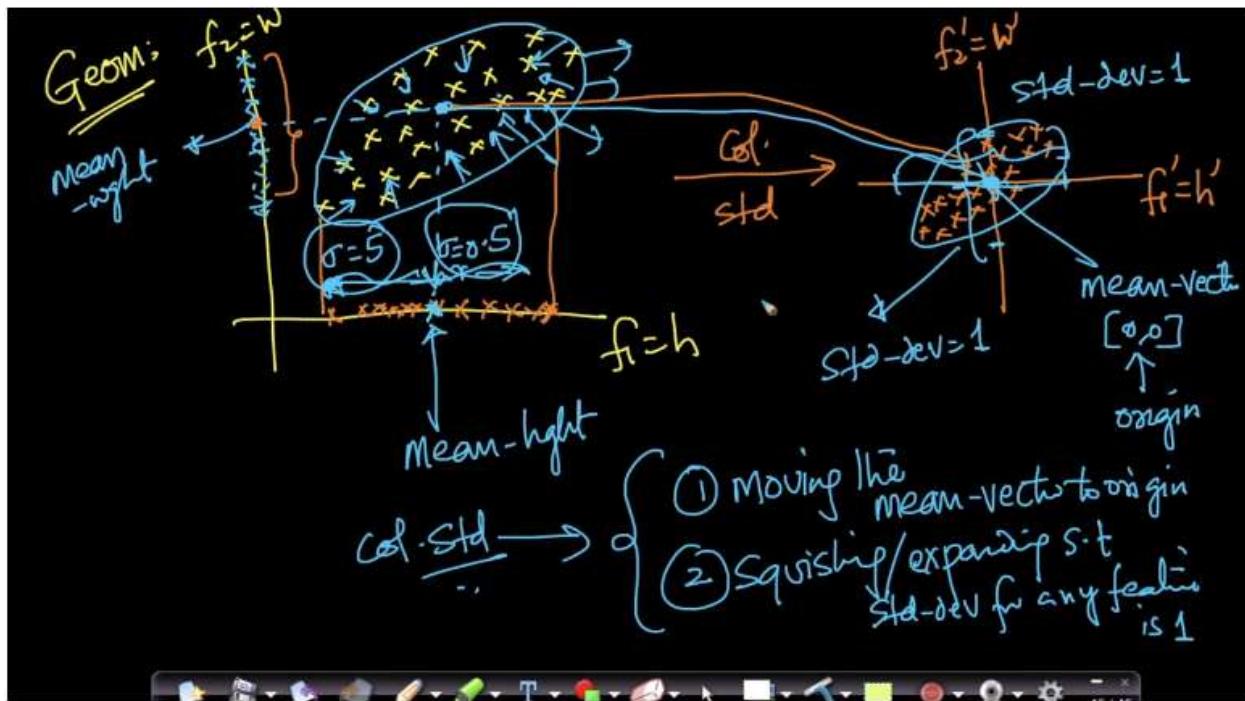
Given a data matrix X, for each feature containing n values a_1, a_2, \dots, a_n , we convert them into a'_1, a'_2, \dots, a'_n such that the mean of a_i 's is 0 and std of a_i 's is 1.

We do it, by getting a'_i from a_i as below



Timestamp: 6:57

We subtract the mean of a_i and divide by the standard deviation of a_i s from each a_i to get a'_i .



Timestamp: 14:01

Geometrically, with column standardization we are moving the mean vector of our data to origin and squishing or expanding our data such that the standard deviation for any feature is 1.

Col. Standardizatⁿ :- mean - centering → origin
+ scaling → $Std - dev = 1$
 for all features

Timestamp: 14:50

Basically, column standardization is considered as mean centering (such that mean vector is origin) plus scaling (such that standard deviation is 1 for each feature).

25.8 Co-variance of a Data Matrix

The diagram illustrates the definition of the covariance matrix S from a data matrix X .
 - **Def:** $S = \begin{bmatrix} & j \\ i & \end{bmatrix}$ is the covariance matrix of X , where i and j are indices ranging from 1 to d (number of features).
 - **Elements:** S_{ij} is the element in the i th row and j th column of S .
 - **Data Matrix:** $X = \begin{bmatrix} x_1 & x_2 & \dots & x_d \end{bmatrix}$ is an $n \times d$ matrix where x_i is the i th feature vector.
 - **Elements:** x_{ij} is the j th element of the i th feature vector x_i .
 - **Annotations:**
 - f_j = col-vector j th feature
 - (S_{ij}) = i th row & j th col.
 - x_{ij} = j th feature for i th data point.

Timestamp: 4:00

We define the covariance matrix of our data S as a square matrix of $d \times d$ shape, where d is the number of features of our data. Each element S_{ij} which is the element of i th row and j th column is defined as the covariance of i th feature and j th feature as below.

$$S_{ij} = \text{cov}(f_i, f_j) = \text{cov}(f_j, f_i) = S_{ji}$$

$\boxed{\text{cov}(x, y) = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_x)(y_i - \mu_y)}$

$$\text{cov}(f_i, f_j) = \cancel{\text{Var}}(\text{Var}(f_i))$$

$$\left\{ \begin{array}{l} \checkmark \text{cov}(x, x) = \text{Var}(x) \rightarrow 1 \\ \checkmark \text{cov}(f_i, f_i) = \text{cov}(f_i, f_i) \rightarrow 2 \end{array} \right.$$

Timestamp: 9:27

Notice that due to how covariance is defined, $\text{cov}(f_i, f_j) = \text{cov}(f_j, f_i)$ hence $s_{ij} = s_{ji}$.

$S =$

Sym. Matrix

$\begin{matrix} S_{11} & S_{12} & \dots & S_{1d} \\ S_{21} & S_{22} & \dots & S_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ S_{d1} & S_{d2} & \dots & S_{dd} \end{matrix}$

$\xrightarrow{\text{Symmetric matrix}}$

$\xrightarrow{\text{Symmetric matrix}}$

$\xrightarrow{\text{Sq. Matrix (d: rows, d: cols)}}$

$\xrightarrow{\text{Var of feature}}$

$\xrightarrow{\text{Sq. Symm matrix}}$

$\xrightarrow{\text{Symmetric matrix}}$

$\xrightarrow{\text{A} \in \mathbb{R}^{d \times d}, \quad A_{32} = S = A_{2,3}}$

$A_{21} = A_{12}$

$\boxed{A_{ij} = A_{ji} \quad \forall i, j}$

Timestamp: 10:15

As shown above, the covariance matrix S is a square symmetric matrix, as it satisfies all the rules of a square matrix where the number of rows must be equal to the number of columns and that of a symmetric matrix where $S_{ij}=S_{ji}$.

$$X = \frac{1}{n} \begin{bmatrix} f_1 & f_2 & \dots & f_d \\ x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix}$$

Let \textcircled{X} col-standardized \Rightarrow $\text{mean}\{f_i\} = 0$
 $\text{std-dev}\{f_i\} = 1$

$\text{cov}(f_1, f_2) = \frac{1}{n} \sum_{i=1}^n (x_{i1} - \text{mean}(f_1)) \cdot (x_{i2} - \text{mean}(f_2))$

Timestamp: 14:47

If our data matrix X is column standardized then the mean of each feature is 0 and the standard deviation of each feature is 1, hence we are left with $\text{cov}(f_1, f_2)$ as below

$$\text{Cov}(f_1, f_2) = \frac{1}{n} \sum_{i=1}^n x_{i1} * x_{i2}$$

$$\text{Cov}(f_1, f_2) = (f_1^T f_2) * \frac{1}{n}$$

Timestamp: 17:19

So $\text{Cov}(f_1, f_2)$ would just be the summation as above. $\text{Cov}(f_1, f_2)$ can also be represented as $(f_1^T f_2)/n$.

$$S_{d \times d} = \frac{1}{n} \underbrace{(X^T)(X)}_{\substack{\text{data-matrix} \\ d \times n}}_{n \times d} = (d \times d) \checkmark$$

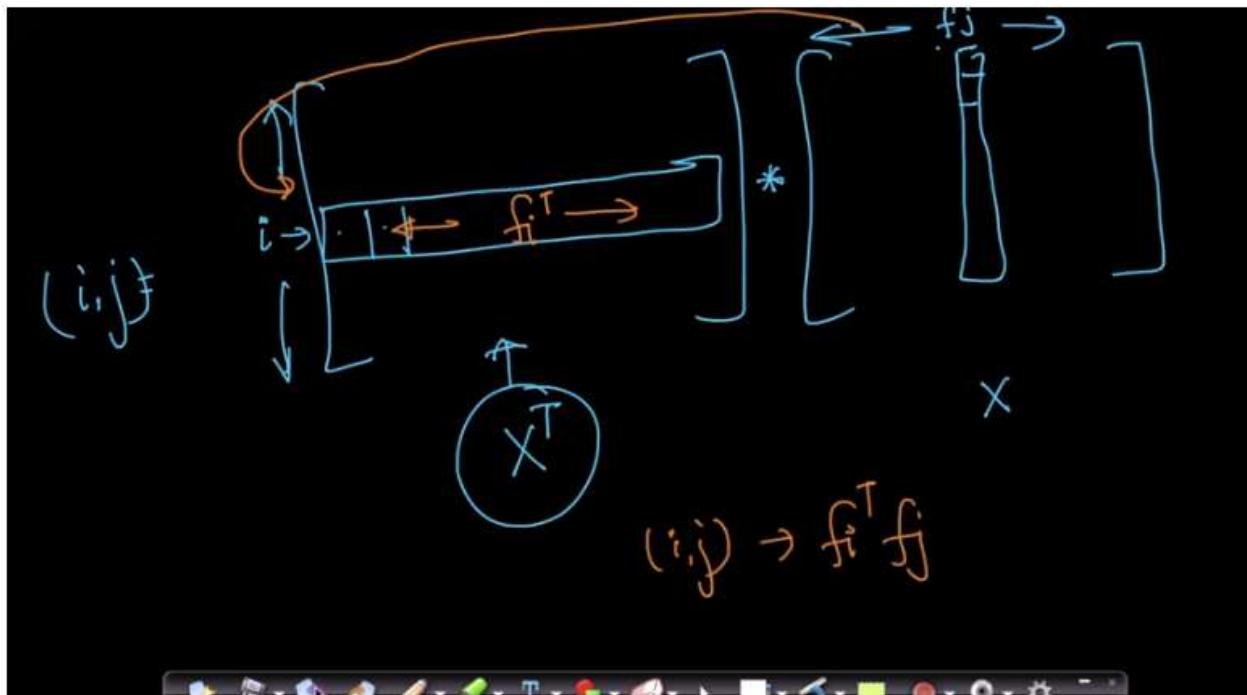
(* assuming X has been col. std.)

LHS $S_{ij} = \text{Cov}(f_i, f_j) = \frac{f_i^T f_j}{n}$

Timestamp: 19:40

Since each element S_{ij} of covariance matrix S is the covariance of feature f_i and f_j , and the $\text{cov}(f_i, f_j) = (f_i^T f_j)/n$, therefore $S_{ij} = (f_i^T f_j)/n$.

We can prove that $S = (X^T X)/n$, by using the below figure.



Timestamp: 22:24

From the above figure we can see the (i,j) the element of $X^T X$ is $f_i^T f_j$ hence the (i,j) the element of $(X^T X)/n$ will be $(f_i^T f_j)/n$ which is how we defined S_{ij} .

Hence $S = (X^T X)/n$ if X has been column standardized.

25.9 MNIST dataset

Timestamp: 3:43

MNIST dataset is a computer vision dataset, where we are given 28x28 pixel images containing hand written digits and their corresponding labels.

MNIST dataset

is :- 4 dim. dataset

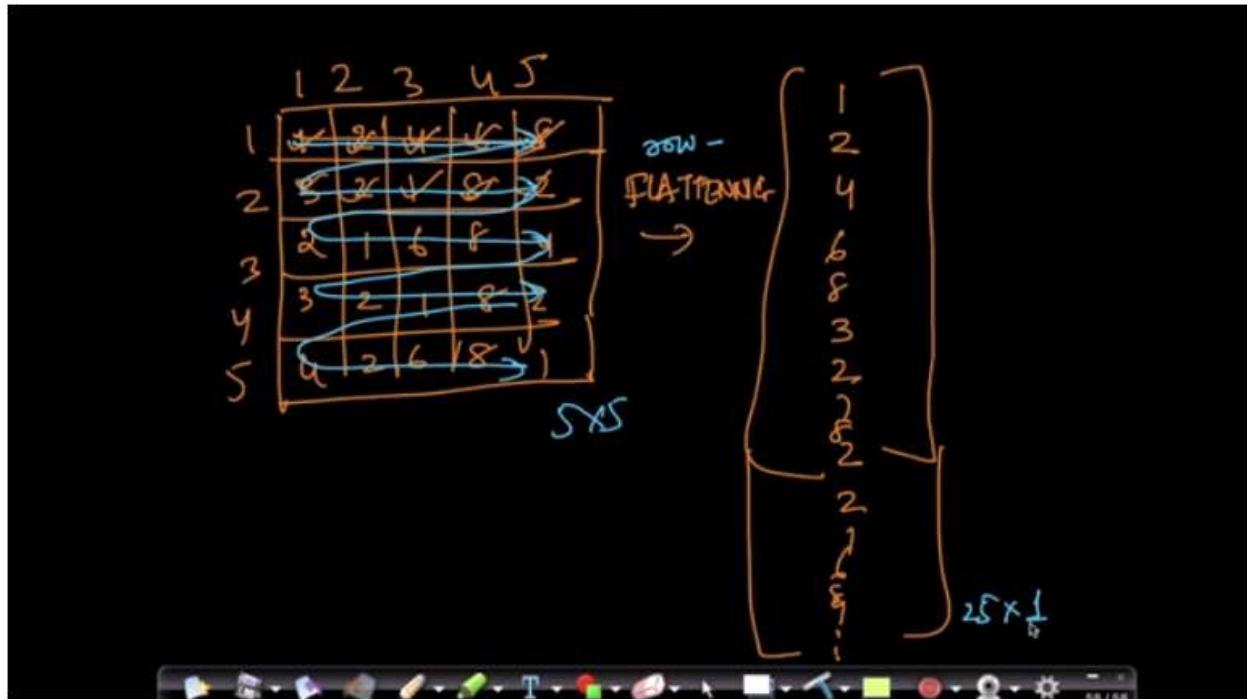
$$\mathcal{D} = \{x_i, y_i\}_{i=1}^{60k}$$

Obj: classify the written char into one of the 10 numeric char.

$x_i : \boxed{\text{[} \text{]}}_{28}^{28} \quad y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Timestamp: 5:30

The dataset can be represented as $D = \{x_i, y_i\}$ where i is from 1 to n , where each x_i is an image containing 784 pixels which are represented by real values (1 indicating black and 0 indicating white and values between 0 and 1 representing shades of grey) and y_i is the corresponding numeric value of that image between 0 and 9.



Timestamp: 10:24

Above is an example how we can do row-flattening to convert a matrix of shape 5×5 to 25×1 by first adding all the elements of the first row, followed by all elements of second row and so on.

$$x_i = \begin{array}{c} \text{image} \\ \xrightarrow{\text{NOT data matrix}} \\ \text{matrix representation of image} \end{array} \xrightarrow{\text{row flattening}} \begin{array}{c} \text{numerical / real} \\ \text{matrix} \end{array} \xrightarrow{\text{row flattening}} \begin{array}{c} x_i \in \mathbb{R}^d \\ \xrightarrow{\text{row flattening}} \\ \begin{bmatrix} 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \\ 28 \times 28 \\ 784 \times 1 \end{array}$$

Timestamp: 10:24

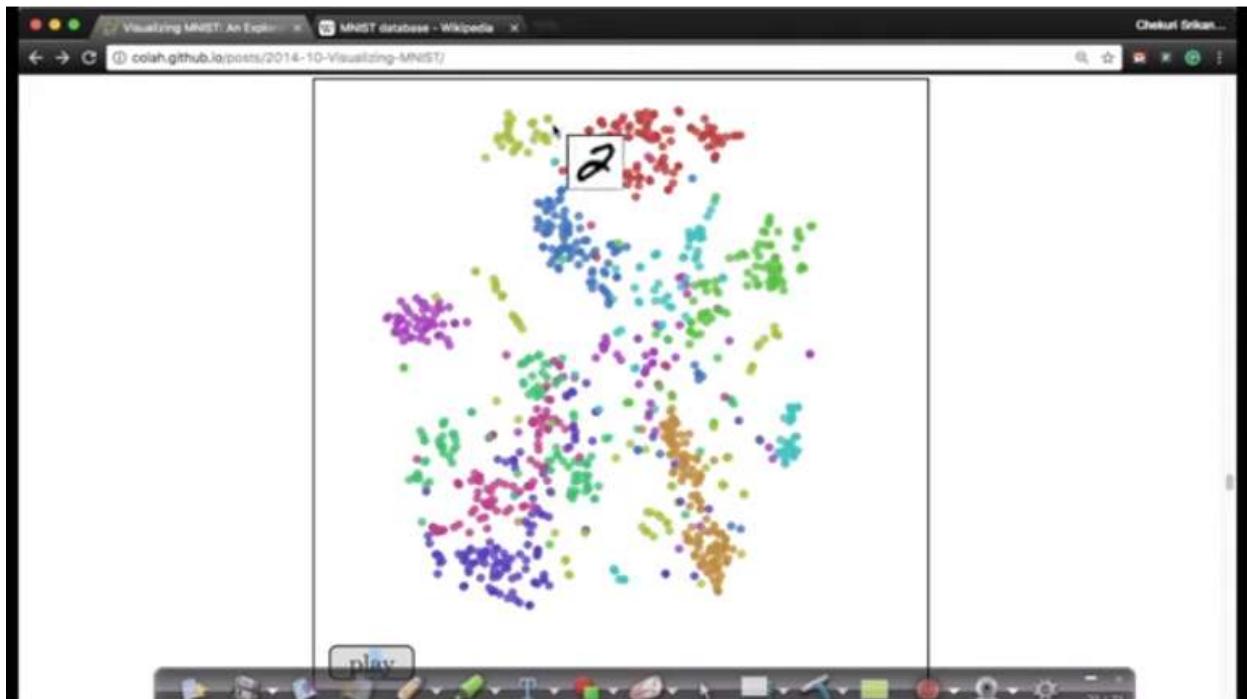
So for each data point x_i , if we do row-flattening we get a column vector of 784×1 dimension.

$$\begin{array}{c} X = \begin{array}{c} f_1 & f_2 & f_3 & \dots \\ \downarrow & & & \\ 1 & & & \\ 2 & & & \\ 3 & & & \\ \vdots & & & \\ \vdots & & & \\ 60k & & & \end{array} \\ \text{MNIST} \end{array} \xrightarrow{\text{row flattening}} \begin{array}{c} x_i^T \in \mathbb{R}^{784} \\ 784 \text{ dim} \end{array} \xrightarrow{\text{row flattening}} \begin{array}{c} f_{784} \\ \left[\begin{array}{c} y_1 \\ \vdots \\ y_n \end{array} \right] \\ n \times 1 \\ \{0, 1, 2, \dots, 9\} \end{array}$$

$n = 60k$
 $d = 784$

Timestamp: 13:06

So our dataset will be represented as above, where X is the data matrix of dimensions $60K \times 784$ since there are $60K$ training data points and 784 features for each data point and y_i contains the label for the corresponding datapoint x_i and belongs to $[0,9]$.



Timestamp: 18:51

Since we have 784 dimensional data which is difficult to visualize, we use dimensionality reduction technique called t-SNE and above are the results (we will see how to apply t-SNE in later chapters). Notice that each point represents each datapoint in the reduced dimension and the color represents the class-label of the datapoint.

25.10 Code to load MNIST dataset

The screenshot shows a Jupyter Notebook interface with three tabs at the top: "mnist_loadData_pca_tsne.ipynb", "Digit Recognizer | Kaggle", and "Visualizing MNIST: An Exploratory Data Analysis". The notebook is running on "localhost:8888/notebooks/Google%20Drive/OnlineVideos/6/Code/mnist_loadData_pca_tsne.ipynb". The code in the cell is as follows:

```
In [36]: # MNIST dataset downloaded from Kaggle :  
#https://www.kaggle.com/c/digit-recognizer/data  
  
# Functions to read and show images.  
  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
  
d0 = pd.read_csv('./mnist_train.csv')  
print(d0.head(5)) # print first five rows of d0.  
  
# save the labels into a variable l.  
l = d0['label']  
  
# Drop the label feature and store the pixel data in d.  
d = d0.drop("label",axis=1)
```

Timestep: 1:42

Downloaded the data from the link mentioned, renamed the train csv as mnist_train.csv, imported the necessary libraries and storing the label data and pixel data into l and d respectively.

The screenshot shows a Jupyter Notebook interface with three tabs at the top: "mnist_loadData_pca_tsne.ipynb", "Digit Recognizer | Kaggle", and "Visualizing MNIST: An Exploratory Data Analysis". The notebook is running on "localhost:8888/notebooks/Google%20Drive/OnlineVideos/6/Code/mnist_loadData_pca_tsne.ipynb". The code in the cell is as follows:

```
d = d0.drop("label",axis=1)
```

A red curly brace highlights the "label" column, and a red arrow points to the "label" column header. A red horizontal line highlights the "pixel0" through "pixel778" columns. The data table below shows the first few rows of the DataFrame:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	pixel16	pixel17	pixel18	pixel19	pixel20	pixel21	pixel22	pixel23	pixel24	pixel25	pixel26	pixel27	pixel28	pixel29	pixel30	pixel31	pixel32	pixel33	pixel34	pixel35	pixel36	pixel37	pixel38	pixel39	pixel40	pixel41	pixel42	pixel43	pixel44	pixel45	pixel46	pixel47	pixel48	pixel49	pixel50	pixel51	pixel52	pixel53	pixel54	pixel55	pixel56	pixel57	pixel58	pixel59	pixel60	pixel61	pixel62	pixel63	pixel64	pixel65	pixel66	pixel67	pixel68	pixel69	pixel70	pixel71	pixel72	pixel73	pixel74	pixel75	pixel76	pixel77	pixel78	pixel79	pixel80	pixel81	pixel82	pixel83	pixel84	pixel85	pixel86	pixel87	pixel88	pixel89	pixel90	pixel91	pixel92	pixel93	pixel94	pixel95	pixel96	pixel97	pixel98	pixel99	pixel100	pixel101	pixel102	pixel103	pixel104	pixel105	pixel106	pixel107	pixel108	pixel109	pixel110	pixel111	pixel112	pixel113	pixel114	pixel115	pixel116	pixel117	pixel118	pixel119	pixel120	pixel121	pixel122	pixel123	pixel124	pixel125	pixel126	pixel127	pixel128	pixel129	pixel130	pixel131	pixel132	pixel133	pixel134	pixel135	pixel136	pixel137	pixel138	pixel139	pixel140	pixel141	pixel142	pixel143	pixel144	pixel145	pixel146	pixel147	pixel148	pixel149	pixel150	pixel151	pixel152	pixel153	pixel154	pixel155	pixel156	pixel157	pixel158	pixel159	pixel160	pixel161	pixel162	pixel163	pixel164	pixel165	pixel166	pixel167	pixel168	pixel169	pixel170	pixel171	pixel172	pixel173	pixel174	pixel175	pixel176	pixel177	pixel178	pixel179	pixel180	pixel181	pixel182	pixel183	pixel184	pixel185	pixel186	pixel187	pixel188	pixel189	pixel190	pixel191	pixel192	pixel193	pixel194	pixel195	pixel196	pixel197	pixel198	pixel199	pixel200	pixel201	pixel202	pixel203	pixel204	pixel205	pixel206	pixel207	pixel208	pixel209	pixel210	pixel211	pixel212	pixel213	pixel214	pixel215	pixel216	pixel217	pixel218	pixel219	pixel220	pixel221	pixel222	pixel223	pixel224	pixel225	pixel226	pixel227	pixel228	pixel229	pixel230	pixel231	pixel232	pixel233	pixel234	pixel235	pixel236	pixel237	pixel238	pixel239	pixel240	pixel241	pixel242	pixel243	pixel244	pixel245	pixel246	pixel247	pixel248	pixel249	pixel250	pixel251	pixel252	pixel253	pixel254	pixel255	pixel256	pixel257	pixel258	pixel259	pixel260	pixel261	pixel262	pixel263	pixel264	pixel265	pixel266	pixel267	pixel268	pixel269	pixel270	pixel271	pixel272	pixel273	pixel274	pixel275	pixel276	pixel277	pixel278	pixel279	pixel280	pixel281	pixel282	pixel283	pixel284	pixel285	pixel286	pixel287	pixel288	pixel289	pixel290	pixel291	pixel292	pixel293	pixel294	pixel295	pixel296	pixel297	pixel298	pixel299	pixel300	pixel301	pixel302	pixel303	pixel304	pixel305	pixel306	pixel307	pixel308	pixel309	pixel310	pixel311	pixel312	pixel313	pixel314	pixel315	pixel316	pixel317	pixel318	pixel319	pixel320	pixel321	pixel322	pixel323	pixel324	pixel325	pixel326	pixel327	pixel328	pixel329	pixel330	pixel331	pixel332	pixel333	pixel334	pixel335	pixel336	pixel337	pixel338	pixel339	pixel340	pixel341	pixel342	pixel343	pixel344	pixel345	pixel346	pixel347	pixel348	pixel349	pixel350	pixel351	pixel352	pixel353	pixel354	pixel355	pixel356	pixel357	pixel358	pixel359	pixel360	pixel361	pixel362	pixel363	pixel364	pixel365	pixel366	pixel367	pixel368	pixel369	pixel370	pixel371	pixel372	pixel373	pixel374	pixel375	pixel376	pixel377	pixel378	pixel379	pixel380	pixel381	pixel382	pixel383	pixel384	pixel385	pixel386	pixel387	pixel388	pixel389	pixel390	pixel391	pixel392	pixel393	pixel394	pixel395	pixel396	pixel397	pixel398	pixel399	pixel400	pixel401	pixel402	pixel403	pixel404	pixel405	pixel406	pixel407	pixel408	pixel409	pixel410	pixel411	pixel412	pixel413	pixel414	pixel415	pixel416	pixel417	pixel418	pixel419	pixel420	pixel421	pixel422	pixel423	pixel424	pixel425	pixel426	pixel427	pixel428	pixel429	pixel430	pixel431	pixel432	pixel433	pixel434	pixel435	pixel436	pixel437	pixel438	pixel439	pixel440	pixel441	pixel442	pixel443	pixel444	pixel445	pixel446	pixel447	pixel448	pixel449	pixel450	pixel451	pixel452	pixel453	pixel454	pixel455	pixel456	pixel457	pixel458	pixel459	pixel460	pixel461	pixel462	pixel463	pixel464	pixel465	pixel466	pixel467	pixel468	pixel469	pixel470	pixel471	pixel472	pixel473	pixel474	pixel475	pixel476	pixel477	pixel478	pixel479	pixel480	pixel481	pixel482	pixel483	pixel484	pixel485	pixel486	pixel487	pixel488	pixel489	pixel490	pixel491	pixel492	pixel493	pixel494	pixel495	pixel496	pixel497	pixel498	pixel499	pixel500	pixel501	pixel502	pixel503	pixel504	pixel505	pixel506	pixel507	pixel508	pixel509	pixel510	pixel511	pixel512	pixel513	pixel514	pixel515	pixel516	pixel517	pixel518	pixel519	pixel520	pixel521	pixel522	pixel523	pixel524	pixel525	pixel526	pixel527	pixel528	pixel529	pixel530	pixel531	pixel532	pixel533	pixel534	pixel535	pixel536	pixel537	pixel538	pixel539	pixel540	pixel541	pixel542	pixel543	pixel544	pixel545	pixel546	pixel547	pixel548	pixel549	pixel550	pixel551	pixel552	pixel553	pixel554	pixel555	pixel556	pixel557	pixel558	pixel559	pixel560	pixel561	pixel562	pixel563	pixel564	pixel565	pixel566	pixel567	pixel568	pixel569	pixel570	pixel571	pixel572	pixel573	pixel574	pixel575	pixel576	pixel577	pixel578	pixel579	pixel580	pixel581	pixel582	pixel583	pixel584	pixel585	pixel586	pixel587	pixel588	pixel589	pixel590	pixel591	pixel592	pixel593	pixel594	pixel595	pixel596	pixel597	pixel598	pixel599	pixel600	pixel601	pixel602	pixel603	pixel604	pixel605	pixel606	pixel607	pixel608	pixel609	pixel610	pixel611	pixel612	pixel613	pixel614	pixel615	pixel616	pixel617	pixel618	pixel619	pixel620	pixel621	pixel622	pixel623	pixel624	pixel625	pixel626	pixel627	pixel628	pixel629	pixel630	pixel631	pixel632	pixel633	pixel634	pixel635	pixel636	pixel637	pixel638	pixel639	pixel640	pixel641	pixel642	pixel643	pixel644	pixel645	pixel646	pixel647	pixel648	pixel649	pixel650	pixel651	pixel652	pixel653	pixel654	pixel655	pixel656	pixel657	pixel658	pixel659	pixel660	pixel661	pixel662	pixel663	pixel664	pixel665	pixel666	pixel667	pixel668	pixel669	pixel670	pixel671	pixel672	pixel673	pixel674	pixel675	pixel676	pixel677	pixel678	pixel679	pixel680	pixel681	pixel682	pixel683	pixel684	pixel685	pixel686	pixel687	pixel688	pixel689	pixel690	pixel691	pixel692	pixel693	pixel694	pixel695	pixel696	pixel697	pixel698	pixel699	pixel700	pixel701	pixel702	pixel703	pixel704	pixel705	pixel706	pixel707	pixel708	pixel709	pixel710	pixel711	pixel712	pixel713	pixel714	pixel715	pixel716	pixel717	pixel718	pixel719	pixel720	pixel721	pixel722	pixel723	pixel724	pixel725	pixel726	pixel727	pixel728	pixel729	pixel730	pixel731	pixel732	pixel733	pixel734	pixel735	pixel736	pixel737	pixel738	pixel739	pixel740	pixel741	pixel742	pixel743	pixel744	pixel745	pixel746	pixel747	pixel748	pixel749	pixel750	pixel751	pixel752	pixel753	pixel754	pixel755	pixel756	pixel757	pixel758	pixel759	pixel760	pixel761	pixel762	pixel763	pixel764	pixel765	pixel766	pixel767	pixel768	pixel769	pixel770	pixel771	pixel772	pixel773	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784	pixel785	pixel786	pixel787	pixel788	pixel789	pixel790	pixel791	pixel792	pixel793	pixel794	pixel795	pixel796	pixel797	pixel798	pixel799	pixel800	pixel801	pixel802	pixel803	pixel804	pixel805	pixel806	pixel807	pixel808	pixel809	pixel810	pixel811	pixel812	pixel813	pixel814	pixel815	pixel816	pixel817	pixel818	pixel819	pixel820	pixel821	pixel822	pixel823	pixel824	pixel825	pixel826	pixel827	pixel828	pixel829	pixel830	pixel831	pixel832	pixel833	pixel834	pixel835	pixel836	pixel837	pixel838	pixel839	pixel840	pixel841	pixel842	pixel843	pixel844	pixel845	pixel846	pixel847	pixel848	pixel849	pixel850	pixel851	pixel852	pixel853	pixel854	pixel855	pixel856	pixel857	pixel858	pixel859	pixel860	pixel861	pixel862	pixel863	pixel864	pixel865	pixel866	pixel867	pixel868	pixel869	pixel870	pixel871	pixel872	pixel873	pixel874	pixel875	pixel876	pixel877	pixel878	pixel879	pixel880	pixel881	pixel882	pixel883	pixel884	pixel885	pixel886	pixel887	pixel888	pixel889	pixel890	pixel891	pixel892	pixel893	pixel894	pixel895	pixel896	pixel897	pixel898	pixel899	pixel900	pixel901	pixel902	pixel903	pixel904	pixel905	pixel906	pixel907	pixel908	pixel909	pixel910	pixel911	pixel912	pixel913	pixel914	pixel915	pixel916	pixel917	pixel918	pixel919	pixel920	pixel921	pixel922	pixel923	pixel924	pixel925	pixel926	pixel927	pixel928	pixel929	pixel930	pixel931	pixel932	pixel933	pixel934	pixel935	pixel936	pixel937	pixel938	pixel939	pixel940	pixel941	pixel942	pixel943	pixel944	pixel945	pixel946	pixel947	pixel948	pixel949	pixel950	pixel951	pixel952	pixel953	pixel954	pixel955	pixel956	pixel957	pixel958	pixel959	pixel960	pixel961	pixel962	pixel963	pixel964	pixel965	pixel966	pixel967	pixel968	pixel969	pixel970	pixel971	pixel972	pixel973	pixel974	pixel975	pixel976	pixel977	pixel978	pixel979	pixel980	pixel981	pixel982	pixel983	pixel984	pixel985	pixel986	pixel987	pixel988	pixel989	pixel990	pixel991	pixel992	pixel993	pixel994	pixel995	pixel996	pixel997	pixel998	pixel999	pixel1000
--	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	-----------

Timestamp: 5:06

```
minist_loadData_pca.ipynb    Digit Recognizer | Kaggle    Visualizing MNIST: An Exploratory Data Analysis
localhost:8888/notebooks/GoogleDrive/OneDrive/OnlineVideos/6/Code/minist_loadData_pca_time.ipynb
File Edit View Insert Cell Kernel Help Trusted Python 3
[ 3   4   0   0   0   0   0   0   0   0   0
  4   0   0   0   0   0   0   0   0   0   0
  pixel8 ... pixel774 pixel775 pixel776 pixel777 pixel778 \
  0   0   ...   0   0   0   0   0   0
  1   0   ...   0   0   0   0   0   0
  2   0   ...   0   0   0   0   0   0
  3   0   ...   0   0   0   0   0   0
  4   0   ...   0   0   0   0   0   0
  pixel779 pixel780 pixel781 pixel782 pixel783
  0   0   0   0   0
  1   0   0   0   0
  2   0   0   0   0
  3   0   0   0   0
  4   0   0   0   0
[5 rows x 785 columns]
```

Timestamp: 5:30

As discussed in our data, we have the label column and pixel data from pixel 0 to pixel 783.

Code to load MNIST Data set: Dimensionality reduction Lecture 1...

```
File Edit View Insert Cell Kernel Help Trusted Python 3
[ 42000, 104]
(42000,)

In [31]: # display or plot a number.
plt.figure(figsize=(7,7))
idx = 150

grid_data = d.iloc[idx].as_matrix().reshape(28,28) # reshape from 1d to 2d
plt.imshow(grid_data, interpolation = "none", cmap = "gray")
plt.show()

print(l[idx])
```

0
4
5

MORE VIDEOS

9:53 / 11:33 CC YouTube

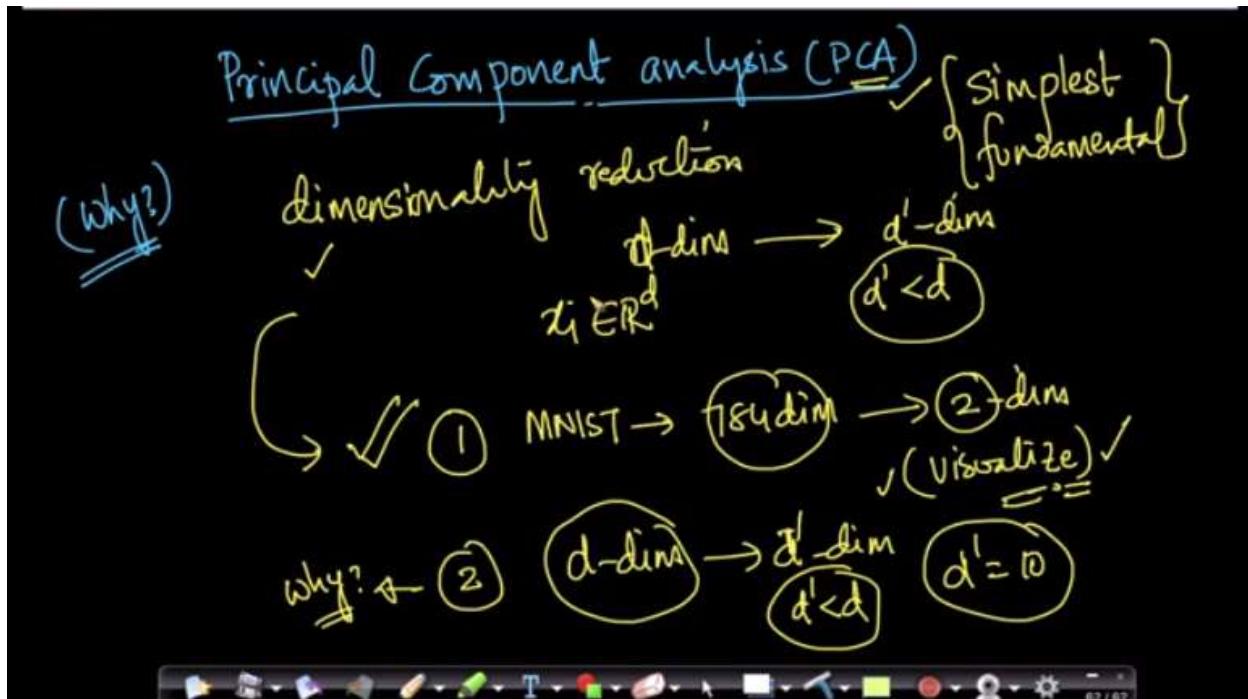
Timestamp: 9:51

Using the above code, we plot images for us to visualize. The plots look like below.



Timestamp: 9:55

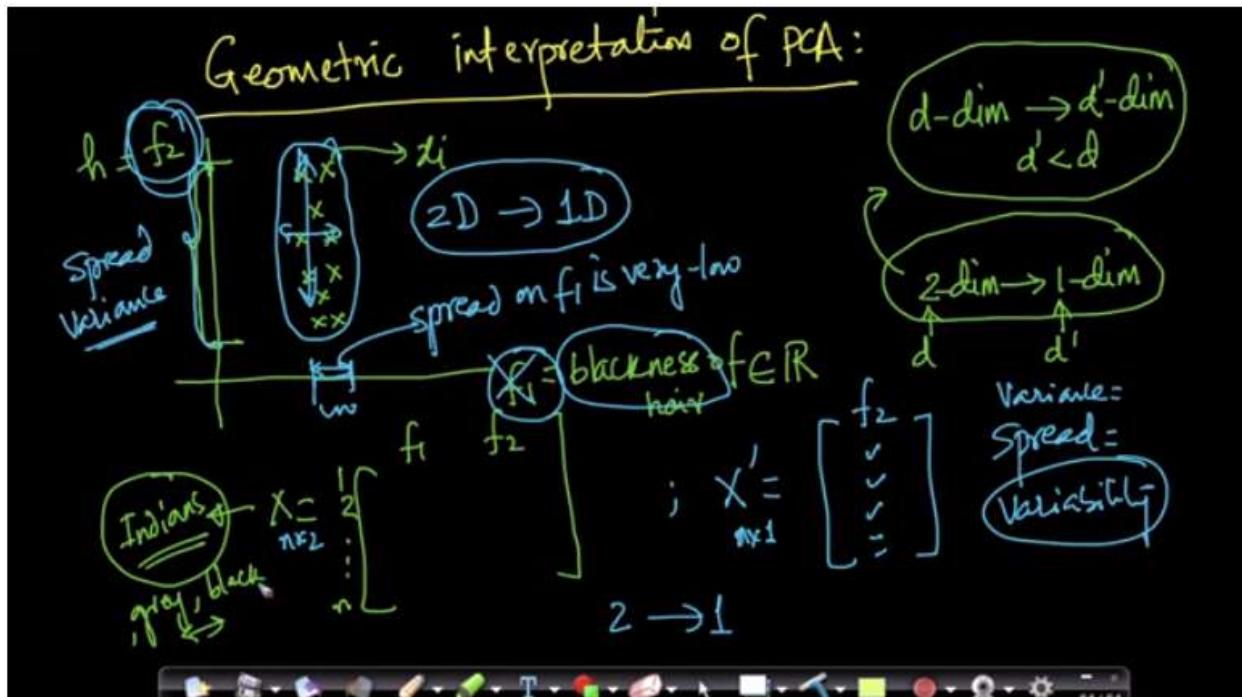
26.1 Why learn PCA?



Timestamp: 3:25

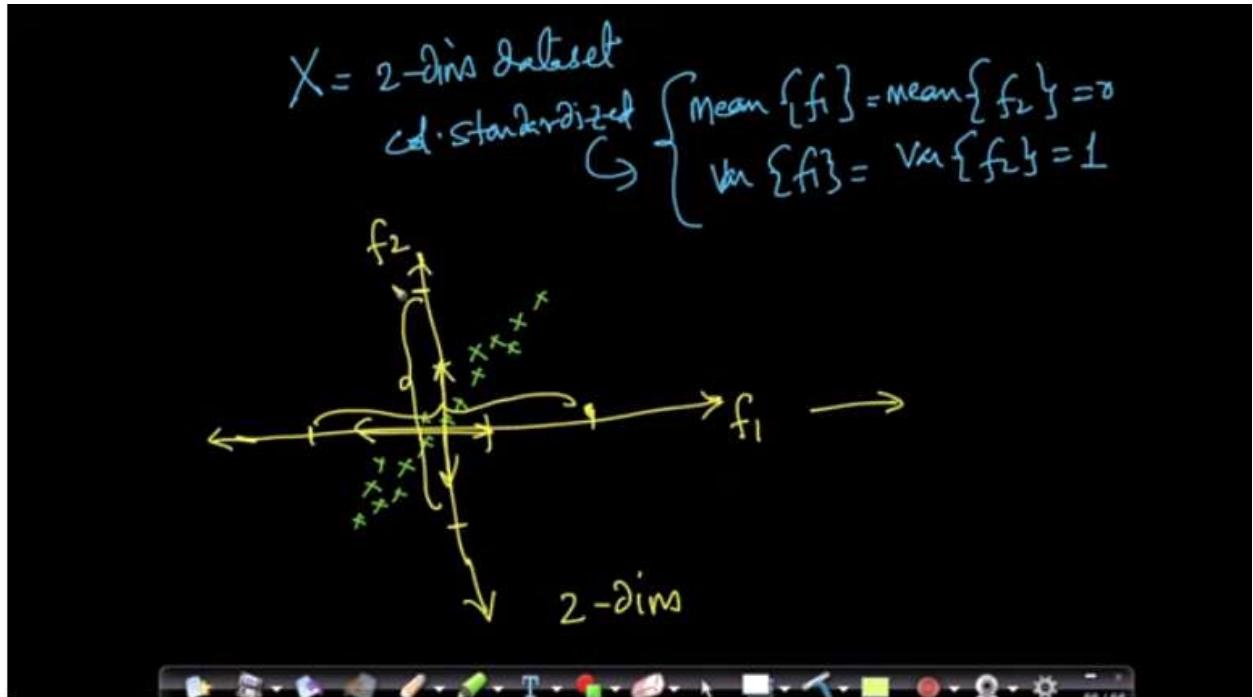
PCA is used for dimensionality reduction to convert data from d dimensions to d' dimensions where $d' < d$. It has applications to both visualize the data by converting it into 2-D or 3-D. Apart from visualization reducing the dimension of data using PCA has other applications which will be discussed in further chapters.

26.2 Geometric intuition of PCA



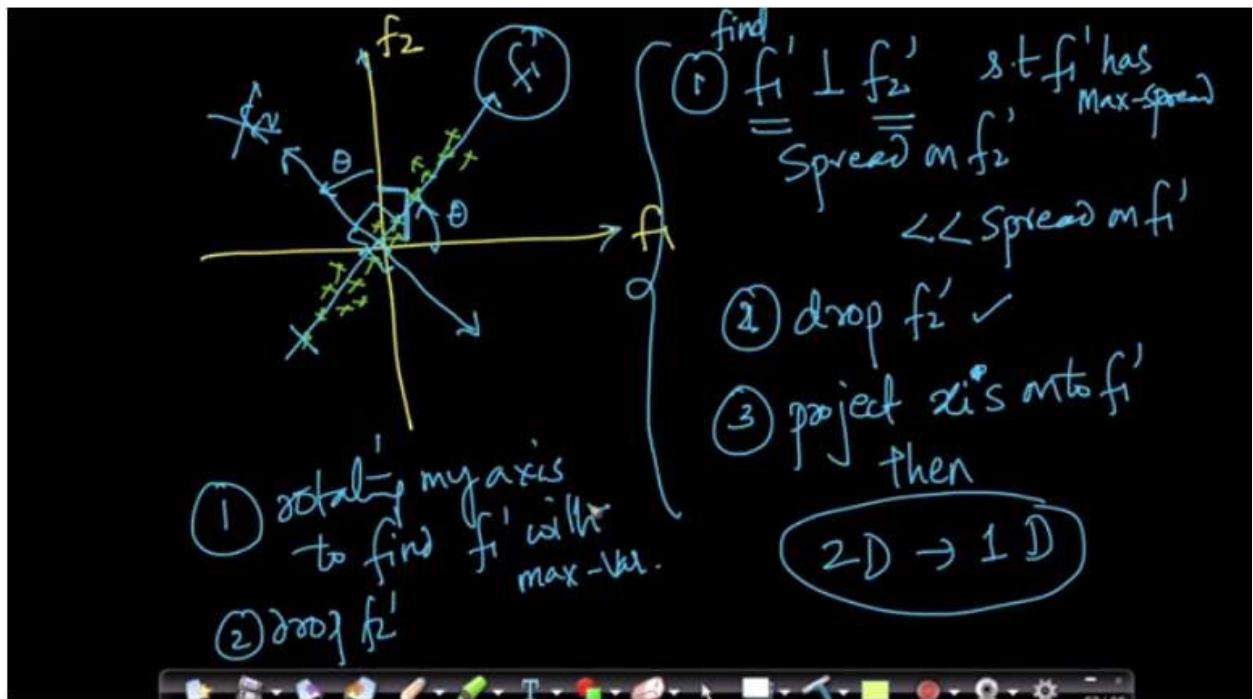
Timestamp: 5:24

As mentioned in the above figure, let's say that the feature f_2 has more spread than feature f_1 , if we have to reduce the dimensions from 2-D to 1-D then we can pick up the feature f_2 where the spread or variability of the data is much more. More the spread, the more we can visualize our data better. Hence we reduce our data which contains feature f_1 and f_2 to a data that contains only one feature f_2 since it has more spread and preserve the direction in which the data has more spread.



Timestamp: 9:16

Suppose we have a 2-D dimensional dataset which is column standardized and let us consider the data is spread more not along the direction f_1 or f_2 but rather as shown in the image. Then we cannot simply just drop f_1 or f_2 because the spread is not in either of f_1 or f_2 direction. For this we have to do something smart as below.



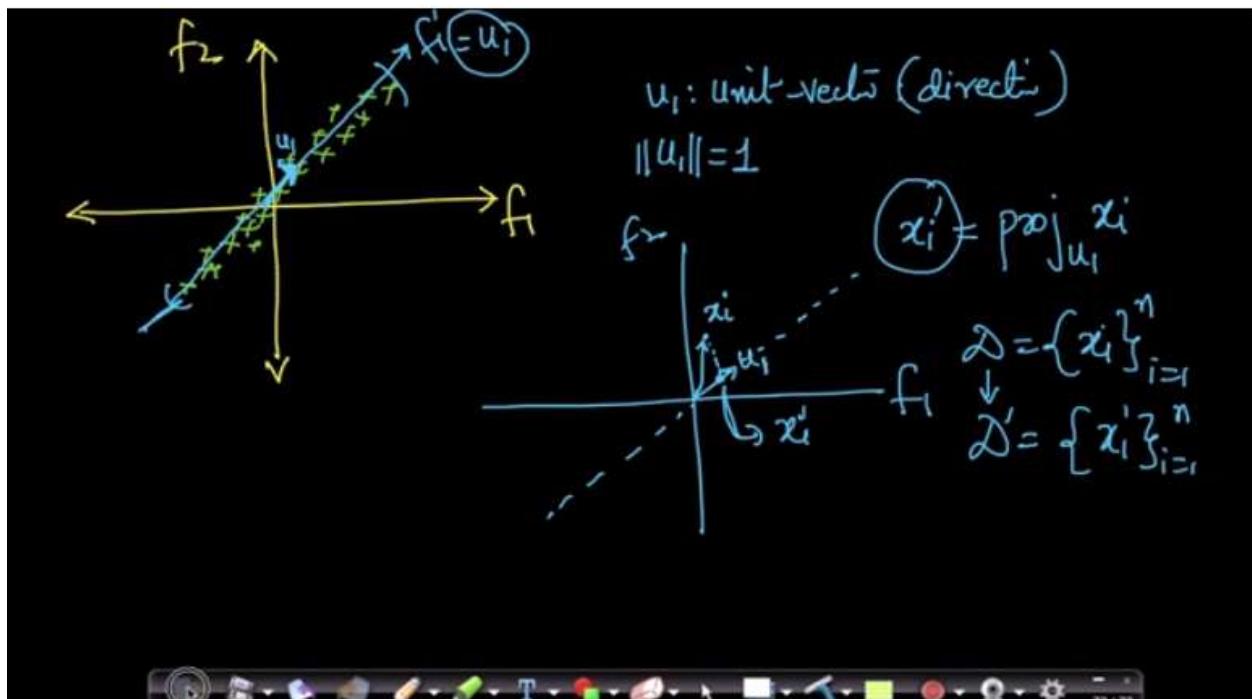
Timestamp: 13:43

As shown in the above figure, we will find f_1' and f_2' such that f_1' is perpendicular to f_2' such that the spread of f_1' is much greater than the spread of f_2' .

We will then drop f_2' and project our data to f_1' , thereby reducing data from 2-D to 1-D.

We can think of this as rotating the axis to find f_1' that has the max-variance and dropping f_2' .

26.3 Mathematical objective function of PCA



Timestamp: 2:34

As discussed before we want to find a direction where the spread of our data is more. Let us say u_1 is a unit vector in that direction. Then to get the new dataset we will project each of our points on u_1 to get the new dataset D' as shown in the image.

$$x'_i = \text{proj}_{u_1} x_i = \frac{u_1 \cdot x_i}{\|u_1\|^2=1} = u_1^T x_i$$

$$\bar{x}' = u_1^T \bar{x}$$

$$\text{mean}\{x'_i\}_{i=1}^n$$

Timestamp: 4:19

Since vector u_1 is a unit vector, the projection of point x_i on u_1 x'_i will be just $u_1^T x_i$, notice that mean of x'_i which is \bar{x}' , will be $u_1^T \bar{x}$. Meaning the mean of projections will be u_1^T multiplied by the mean of x_i s.

④ find u_1 s.t $\text{Var}\{\text{proj}_{u_1} x_i\}_{i=1}^n$ is maximal.

$$\text{Var}\{u_1^T x_i\}_{i=1}^n = \frac{1}{n} \sum_{i=1}^n (u_1^T x_i - \underbrace{u_1^T \bar{x}}_{\text{mean}\{x_i\}_{i=1}^n})^2$$

~~scale = $(u_1^T)^T x_i$~~

$\bar{x} : \text{Col. Standardized}$

$\checkmark \bar{x} = [0, 0, 0, \dots, 0]$

Timestamp: 7:28

Our problem is to find u_1 such that the variance of projections along u_1 is maximal. So writing the variance of our projections, since we have column standardized the data, the mean of the original data points will be a zero vector. Hence our variance is reduced as below.

The image shows a handwritten derivation of the PCA optimization problem. It starts with the formula for the variance of the projections:

$$\text{Var} \left\{ \tilde{x}_i \right\}_{i=1}^n = \frac{1}{n} \sum_{i=1}^n (u_1^T \tilde{x}_i)^2$$

An annotation on the left says "objective of an optimizn problem". A bracket groups the term $\frac{1}{n} \sum_{i=1}^n (u_1^T \tilde{x}_i)^2$ with the label "Data-matrix". To the right, an arrow points to a circle labeled "optimizn problem".

Below this, the optimization problem is written as:

$$\begin{aligned} \max_{u_1} & \quad \frac{1}{n} \sum_{i=1}^n (u_1^T \tilde{x}_i)^2 \\ \text{s.t. } & u_1^T u_1 = 1 = \|u_1\|^2 \end{aligned}$$

A bracket groups the constraint $u_1^T u_1 = 1 = \|u_1\|^2$ with the annotation "Constraint" and "u₁ is a unit vecr". Below the constraint, it says $u_1 = [\infty, \infty]$.

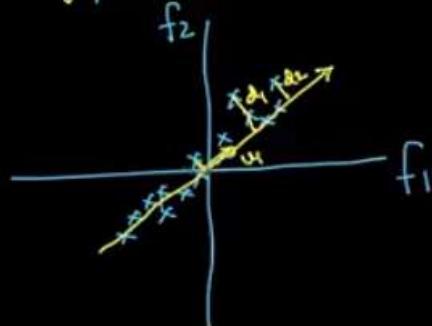
Timestamp: 11:23

So the variance of the projections is reduced to the sum mentioned in the above figure. So now we have our optimization problem for PCA is to find u_1 that maximizes the sum highlighted such that $u_1^T u_1 = 1$. This constraint of making u_1 a unit vector is important since otherwise any vector can satisfy the above equation by making its magnitude infinity. We will see how to solve these optimization problems in later chapters.

26.4 Alternative formulation of PCA: Distance minimization

Alternative formulation of PCA: dist. minimization

→ find u_1 which maximizes projected variance

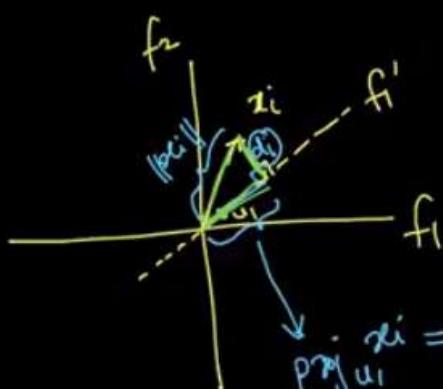


$x_i \rightarrow (\hat{d}_i)$: dist from x_i to u_1

$$\min_{u_1} \sum_{i=1}^n \hat{d}_i^2$$

Timestamp: 2:47

Distance minimization is an alternative formulation of PCA, in which instead of choosing u_1 that maximizes the variance of our projections, we want to choose u_1 that minimizes the distance from x_i to u_1 . Both are equivalent but just different formulations.



u_1 : Unit-Vektor

$$u_1^T u_1 = 1 = \|u_1\|^2$$

$$x_i = u_1^T x_i$$

$$\begin{aligned} d_i^2 &= \|x_i\|^2 - (u_1^T x_i)^2 \\ &= x_i^T x_i - (u_1^T x_i)^2 \end{aligned}$$

Timestamp: 5:36

We find the distance d_i of our point x_i from the direction u_1 , using pythagoras theorem as above. So our final objective function would look like below.

The image contains handwritten mathematical notes on PCA optimization. At the top left, there is a diagram of a point x_i in a 2D space, with its projection onto a line defined by vector u_1 . The distance from the point to the line is labeled d_i . To the left of the diagram, a green circle contains the text "dist min PCA". Below this, a green circle contains "Min u_1 " with a summation symbol $\sum_{i=1}^n$ and terms $(x_i^T x_i)$ and $(u_1^T x_i)^2$. A constraint $u_1^T u_1 = 1$ is shown below. To the right, a blue bracket indicates that this is equal to another formulation. This second formulation is shown in a green circle: "Max u_1 " with a term $\frac{1}{n} \sum_{i=1}^n (u_1^T x_i)^2$ and a constraint $u_1^T u_1 = 1$. To the right of this, a green circle contains the text "Variance maximization PCA". A matrix $X = [x_1 \ x_2 \ \dots \ x_n]$ is also written on the right.

Timestamp: 9:42

So our objective function according to distance formulation looks like above, it can be proved that it will be equal to the variance maximization formulation of PCA. (the proof will be understood when optimization is taught later in the course).

26.5 Eigenvalues and Eigenvectors(PCA): Dimensionality reduction

Solution to our Optimization problems: λ_1, v_1

$X = \begin{bmatrix} 1 & 2 & 3 & \dots & d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n & n & n & \dots & n \end{bmatrix}_{n \times d}$

Covariance matrix of $X = S$

$$S_{d \times d} = X^T X_{d \times n \quad n \times d}$$

Sq. Symm-matrix

eigen-values $(\lambda_1, \lambda_2, \dots, \lambda_d)$
eigen-vector (v_1, v_2, \dots, v_d)

Timestamp: 3:10

Solution to our optimization problem will be to find the eigenvalues and eigenvectors of the covariance matrix S of our data matrix X . (Correction in image $S=(X^T X)/n$) [$S=(X^T X)/(n-1)$ for unbiased estimate. What is meant by biased estimate and unbiased estimate will be covered in the upcoming live session]

$S_{d \times d}$

maximal eigen-value
 $\lambda_1 > \lambda_2 > \lambda_3 > \lambda_4 \dots \rightarrow \lambda_d$

eigen-values of $(S) = \lambda_1, \lambda_2, \lambda_3, \lambda_4, \dots, \lambda_d$

eigen-vectors of $(S) = v_1, v_2, v_3, v_4, \dots, v_d$

def: If $\lambda_1 v_1 = S v_1$ \rightarrow $d \times 1$ vector

$\lambda_1 v_1 = S v_1$

λ_1 : eigen value of S
 v_1 : eigen vector of S

Timestamp: 6:33

Finding eigenvalues and eigen vectors of any matrix S is pretty straightforward and we might have learnt earlier. We have to find eigen vectors and eigen values such that they satisfy the condition highlighted.

Handwritten notes on eigenvalues and eigenvectors of a covariance matrix $S_{d \times d}$. The notes show the relationship between eigenvalues $\lambda_1 > \lambda_2 > \lambda_3 > \dots > \lambda_d$ and eigenvectors $v_1, v_2, v_3, \dots, v_d$. It is stated that $v_i^\top v_j = 0 = v_i \cdot v_j = 0$ for $i \neq j$, indicating orthogonality. A specific eigenvector $u_1 = v_1$ is highlighted as the eigenvector of $S (= X^\top X)$ corresponding to the largest eigenvalue ($= \lambda_1$). A note below states "Max-variance direction".

Timestamp: 9:10

Eigenvectors have a property that every eigen vector is perpendicular to every other eigenvector. If we can find the eigenvectors and eigenvalues of the covariance matrix S , then we can easily get the direction u_1 which maximizes the variance as it is equal to the eigenvector of S that corresponds to the largest eigenvalue.

$$X = \begin{bmatrix} & & \end{bmatrix}_{n \times d}$$

① Col. Std of X is done
 ② $S_{\text{cov}} = X^T X$
 ③ Eigen values & vectors of S
 eigen(S) $\lambda_1 > \lambda_2 > \dots > \lambda_d$
 ④ $u_1 = v_1$ (why?)
 v_1, v_2, \dots, v_d

Timestamp: 11:12

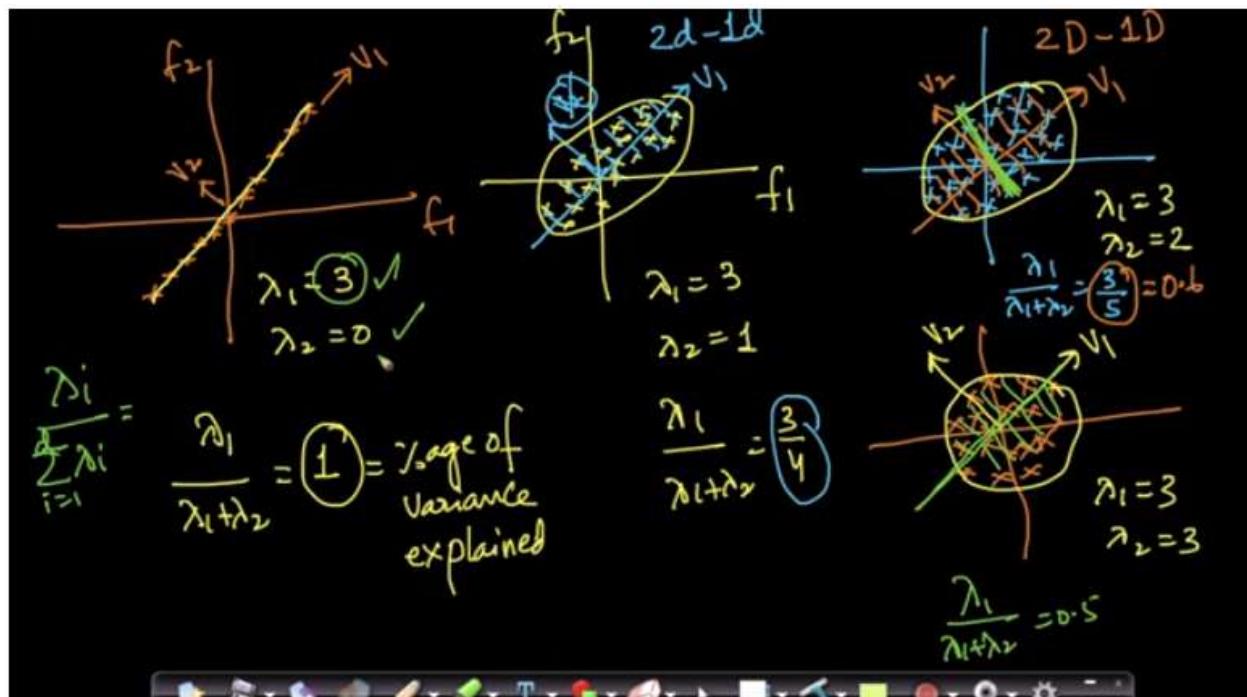
So given a matrix X , in order to find u_1 , the max-variance direction, we need to ensure that X is column standardized, post this we can calculate the covariance matrix S . Following that you can just find the eigenvalues and eigenvectors of S and u_1 will just be the eigenvector with max eigenvalue.

$$x_i \in \mathbb{R}^{10} \quad d=10$$

$\lambda_1 > \lambda_2 > \lambda_3 > \dots > \lambda_{10}$
 ↓ ↓ ↓ ↓ ↓ ↓ ↓
 v_1 v_2 v_3 ... v_{10}
 direction with Max Var 2nd direction with most variance 3rd direction with maximal variance least variance is in v_{10}

Timestamp: 15:33

So if we sort of our eigenvectors of the covariance matrix S in the decreasing order of their corresponding eigenvalues, then v_1 represents the direction with max-variance , v_2 represents the direction with second max-variance and so on.



Timestamp: 22:36

While eigenvectors v_1, v_2, \dots tell us the directions of most-variance, second most variance etc. $\lambda_1, \lambda_2, \dots$ tell us the variance explained in that direction. As shown in the above figure $\lambda_1 / (\lambda_1 + \lambda_2)$ tells us the percentage of variance explained by the direction of v_1 .

26.6 PCA for Dimensionality Reduction and Visualization

184 → 2

$$X = \begin{bmatrix} f_1 & f_2 & \dots & f_{10} \\ \vdots & \vdots & \ddots & \vdots \\ x_i^T & \longrightarrow & \end{bmatrix}_{n \times 10}$$

$\xrightarrow{\substack{\dim \text{ reduce} \\ (\text{PCA})}}$

$$X' = \begin{bmatrix} 1 & \vdots & n \\ x_i^T & \longrightarrow & \end{bmatrix}_{n \times 2}$$

(* visualize v_1 v_2)

$$S = X^T X$$

$$\text{eigen}(S) = \lambda_1 > \lambda_2 > \lambda_3 \dots > \lambda_{10}$$

$$v_1 \quad v_2 \quad \dots \quad v_{10}$$

$$x'_i = [x_i^T v_1, x_i^T v_2]$$

$$(v_1) \quad (v_2)$$


Timestamp: 5:32

Suppose we have data X of 784 dimensions, for us to visualize this data, we can take the covariance matrix S and find the eigenvalues and eigenvectors of matrix S and find the eigenvectors v_1, v_2 that have top two eigenvalues and transform this data to X' by taking each x_i and transforming into x'_i by doing $x_i^T v_1$ and $x_i^T v_2$. Since we can visualize 2-D, we can visualize X' .

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_m \end{bmatrix} \xrightarrow{\text{PCA}} X' = \begin{bmatrix} v_1 & v_2 & \dots & v_{50} \end{bmatrix}$$

$n \times n$

$x_i^T \quad \quad \quad n \times 1$

$S = X^T X$

$\lambda_1 > \lambda_2 > \dots > \lambda_{100}$

v_1, v_2, \dots, v_{100}

$x_{ij}' = x_i^T v_j$

Timestamp: 7:42

If we want to use PCA for just dimensionality reduction, to reduce our data from let's say 100 dimensions to 50 dimensions, then we can simply find the top50 eigenvectors and find x_{ij}' by just doing $x_i^T v_j$.

$$x_i \in \mathbb{R}^{100 \times 1} \quad ; \quad x_i' \in \mathbb{R}^{d'}$$

$d \xrightarrow{\text{PCA}} d'$

$100 \times 1 \quad \quad \quad d' < 100$

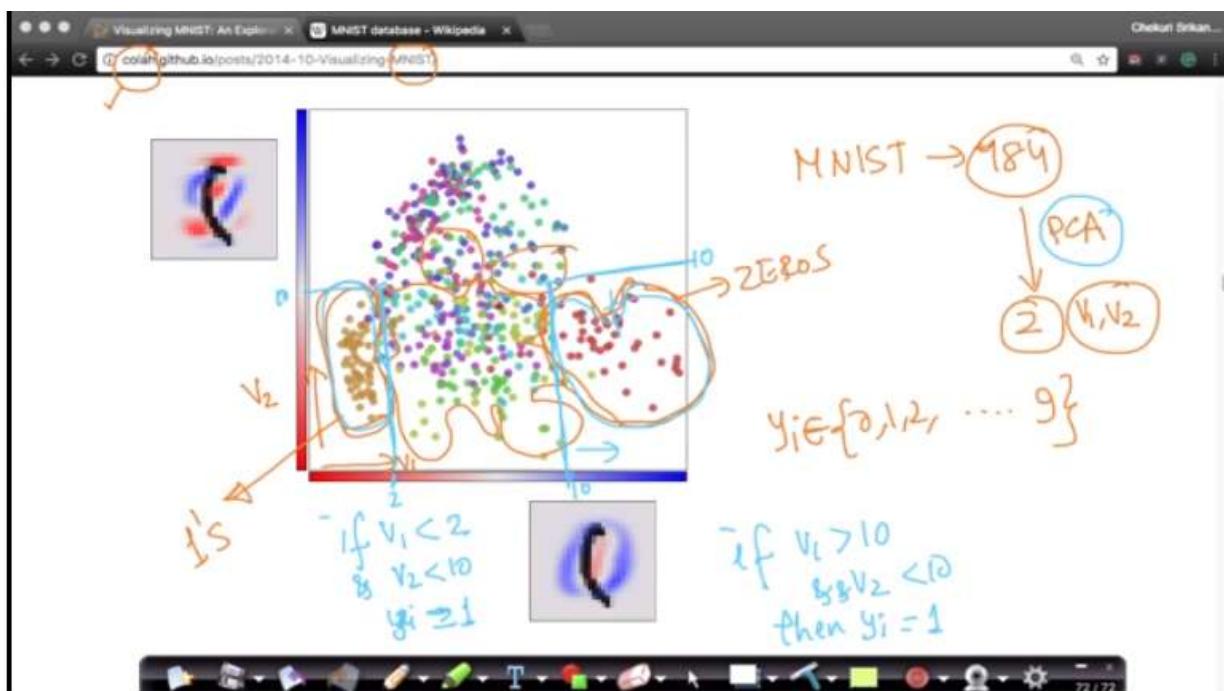
$\checkmark \text{ preserve } \frac{\text{99% of the variance}}{\sum_{i=1}^{100} \lambda_i} = 0.99$

$d' = 51$

Timestamp: 9:42

Often it is the case that we don't pre-decide the dimensions to reduce the data to, often we need to preserve 99% of variance, etc. Then we just simply find the number of eigen vectors that needs to be taken to preserve at least 99% of variance and if 51 eigenvectors account for 99% of total variance then we just project our data onto this top 51 eigenvectors to get our dimension reduced data that has 99% of its variance preserved.

26.7 Visualize MNIST data

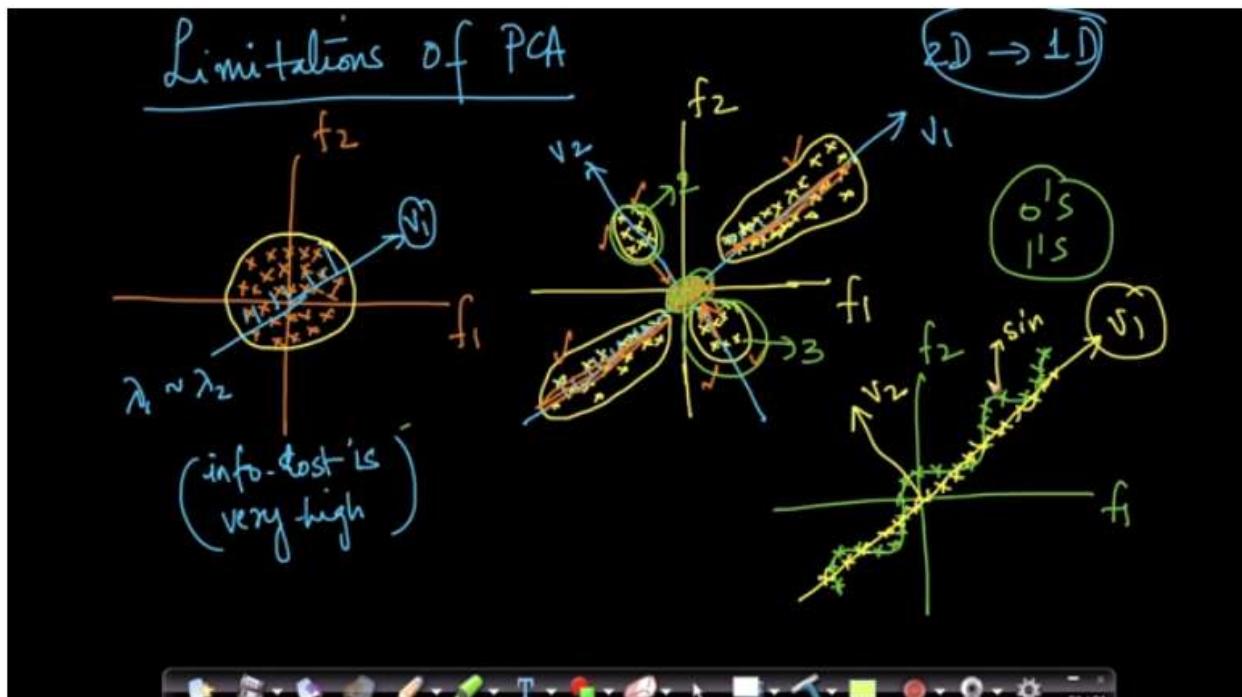


Minor Correction: @ 3:34min , it was mentioned as if $V1>10$ and $V2<10$, then $y_i = 1$. But it is actually:
if $V1>10$ and $V2<10$, then $y_i = 0$

Timestamp: 4:44

When PCA is applied on MNIST data then we get the above visualization. From the above visualization we can see that for some of the classes like zeros and ones, we can easily separate them from the rest of the classes using simple rules such as $v1 > 10$ and $v2 < 10$ then $y=0$, etc.

26.8 Limitations of PCA

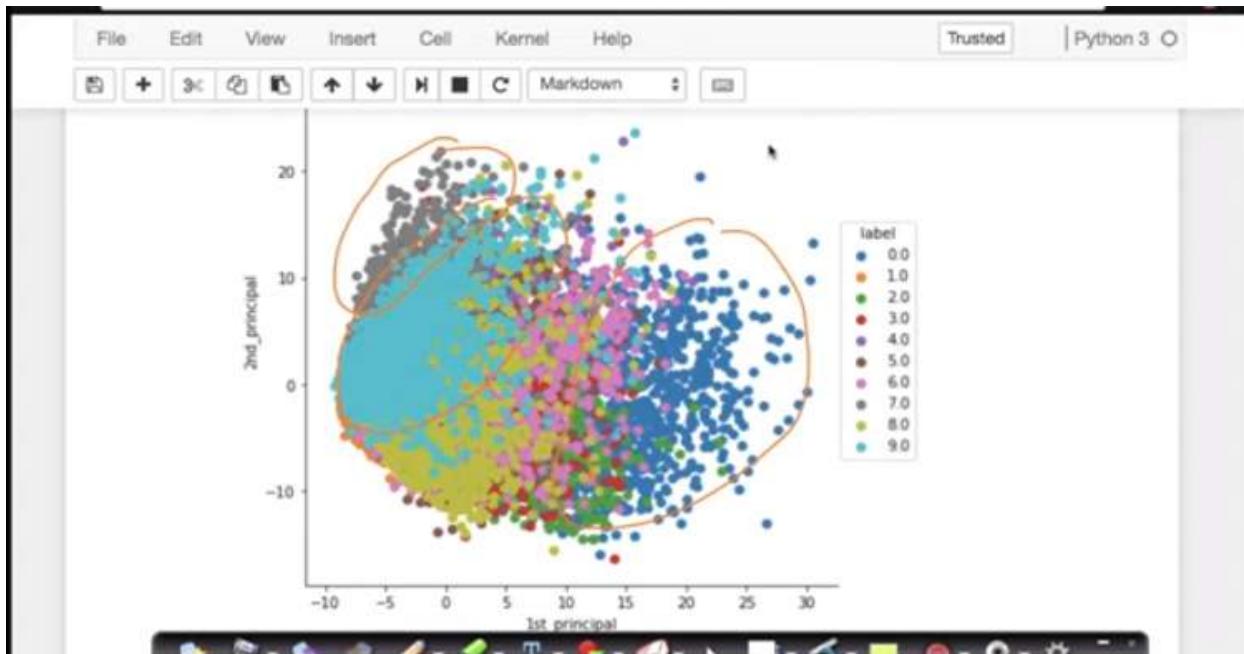


Timestamp: 4:47

There are certain limitations of PCA, as shown in the above images

- 1) Suppose if the original data is like a circle, then the variance is spread among all the directions, but if we reduce the data to lower dimensions, we lose most of the useful information.
- 2) As shown in the second image, when initially the clusters of points are well separated in the original space but when the dimensionality of the data is reduced then points which are well separated might be projected together and hence difficult to separate.
- 3) If the original data has nice shapes like sin wave, etc when dimension is reduced we lose this information.

26.9 PCA code example



Timestamp: 18:15

In this video, we have implemented the code with both our custom code and with PCA provided by scikit-learn. Please refer to the notebook link provided in the video description for the complete code and explanation via comments. From the above plot we can see that some of the classes are well separated and others are not well separated when we visualize the data generated by PCA.

A PCA code Example using visualization :Dimensionality reduction ... Watch later Share Trusted Python 3

PCA using Scikit-Learn

```
In [25]: # initializing the pca
from sklearn import decomposition
pca = decomposition.PCA()

In [26]: # configuring the parameters
the number of components = 2
pca.n_components = 2
pca_data = pca.fit_transform(sample_data)

# pca_reduced will contain the 2-d projects of simple data
print("shape of pca_reduced.shape = ", pca_data.shape)
```

shape of pca_reduced.shape = (15000, 2)

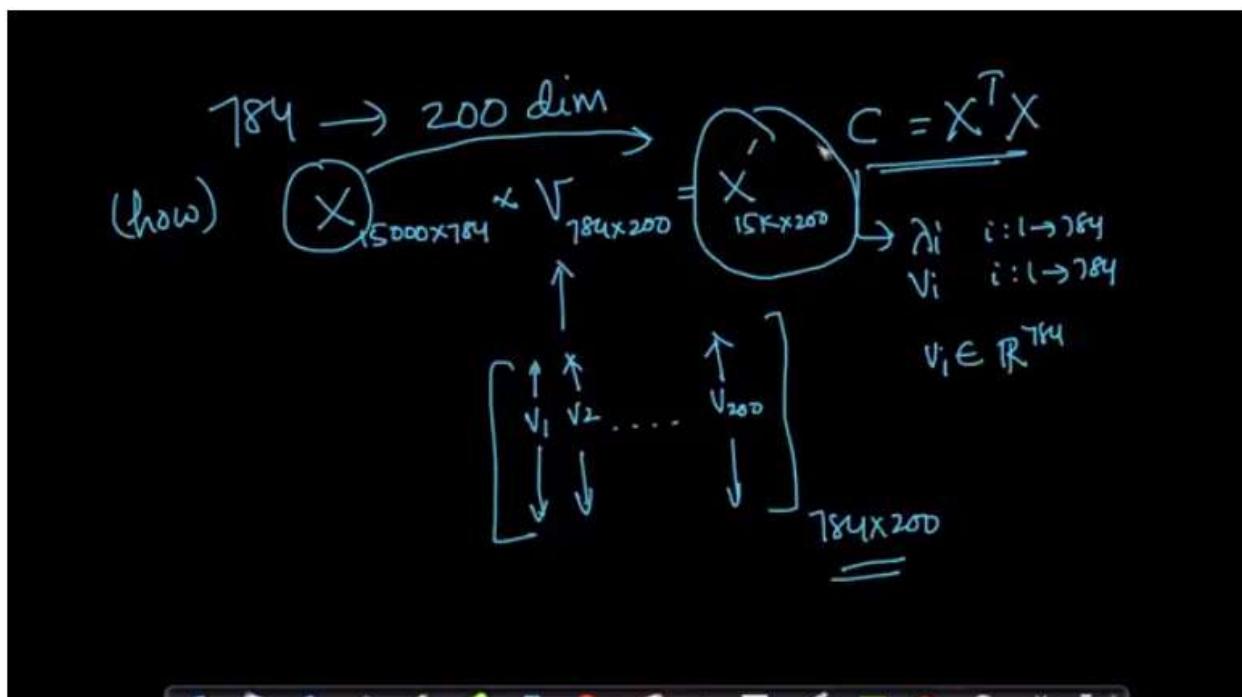
MORE VIDEOS

18:46 / 18:59 CC YouTube

Timestamp: 18:46

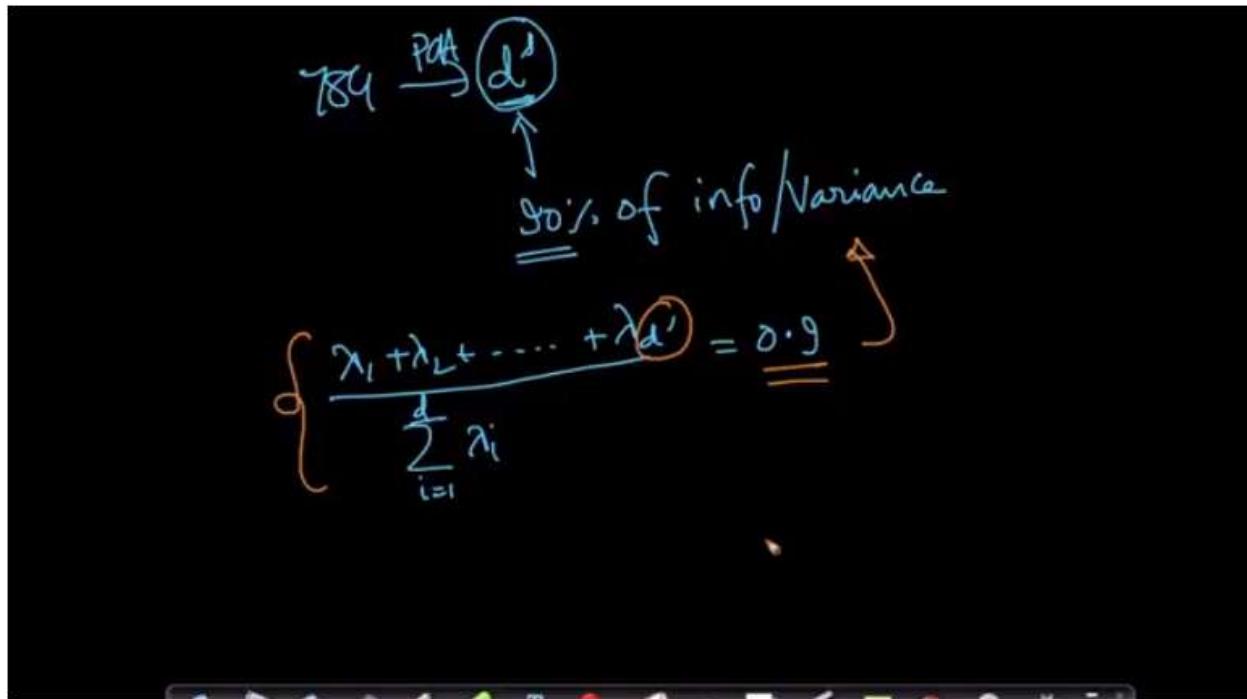
The above is the simple code of scikit learn PCA, using which we can convert the sample data of MNIST of 784 dimensions to 2 dimensions.

26.10 PCA for dimensionality reduction (not-visualization)



Timestamp: 4:21

Using PCA, to reduce the dimensions from 784 dimensions to 200 dimensions, we will just multiply the data X with the matrix of eigenvectors V to get the data of reduced dimension X' . Notice that our matrix of eigenvectors V is formed by placing the top 200 eigenvectors of the correlation matrix of X as a column vector.



Timestamp: 9:46

As discussed before, people often doesn't predecide d' but rather decide on the amount of information/variance of the data they want to preserve and find such d' that preserves 90% of variance of our data and reduce our data from d to d' dimensions.

```

In [28]: # PCA for dimensionality reduction (non-visualization)
pca.n_components = 784
pca_data = pca.fit_transform(sample_data)

percentage_var_explained = pca.explained_variance_ / np.sum(pca.explained_v
cum_var_explained = np.cumsum(percentage_var_explained)

# Plot the PCA spectrum
plt.figure(1, figsize=(6, 4))

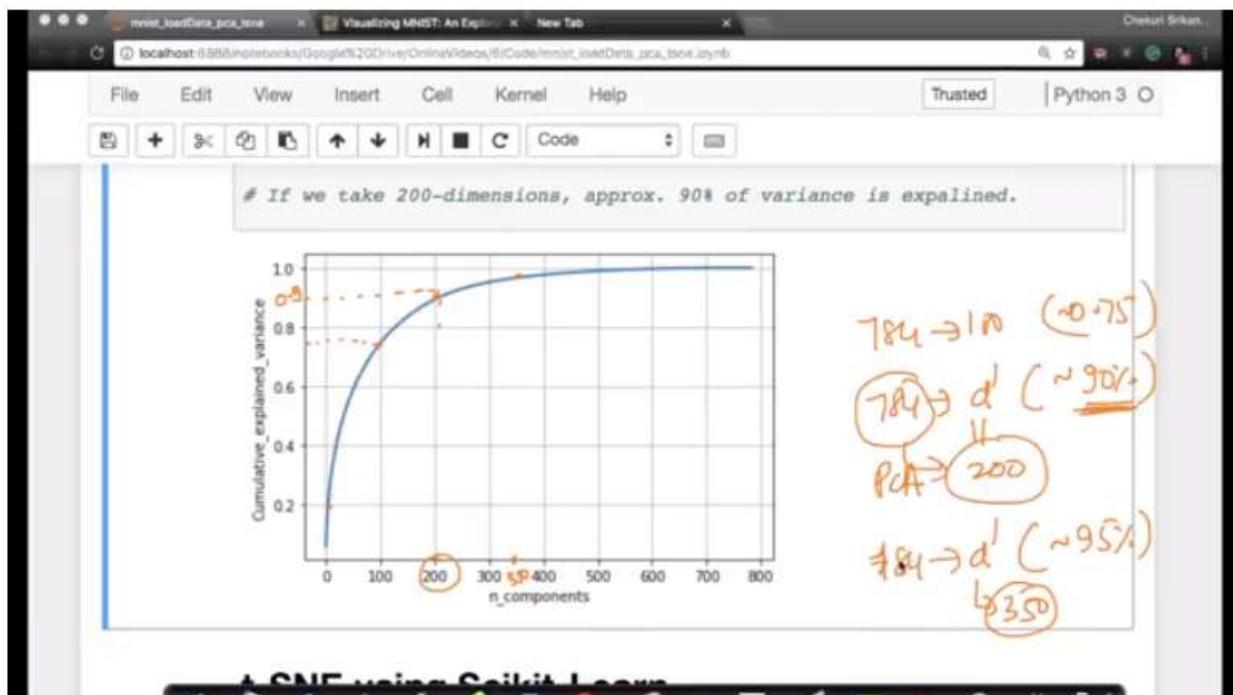
plt.clf()
plt.plot(cum_var_explained, linewidth=2)
plt.axis('tight')
plt.grid()
plt.xlabel('n_components')
plt.ylabel('Cumulative_explained_variance')
plt.show()
    
```

Annotations on the right side of the code cell:

- $\frac{\lambda_1}{\sum \lambda_i}$
- $\frac{\lambda_1}{\sum \lambda_i}, \frac{\lambda_2}{\sum \lambda_i}, \frac{\lambda_3}{\sum \lambda_i}, \dots$
- $\frac{\lambda_1}{\sum \lambda_i}, \frac{\lambda_1 + \lambda_2}{\sum \lambda_i}, \frac{\lambda_1 + \lambda_2 + \lambda_3}{\sum \lambda_i}, \dots$

Timestamp: 12:31

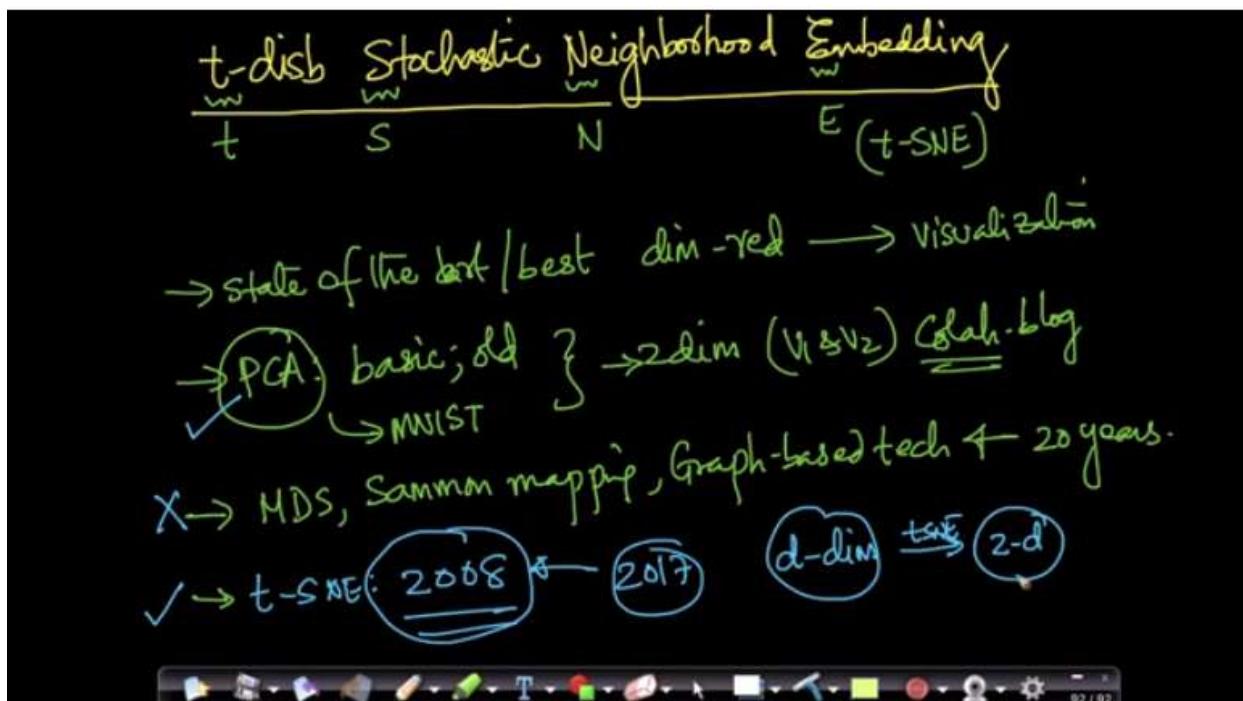
The above is the code to plot the variance explained when we take different d' , the dimensions to reduce the data into.



Timestamp: 14:36

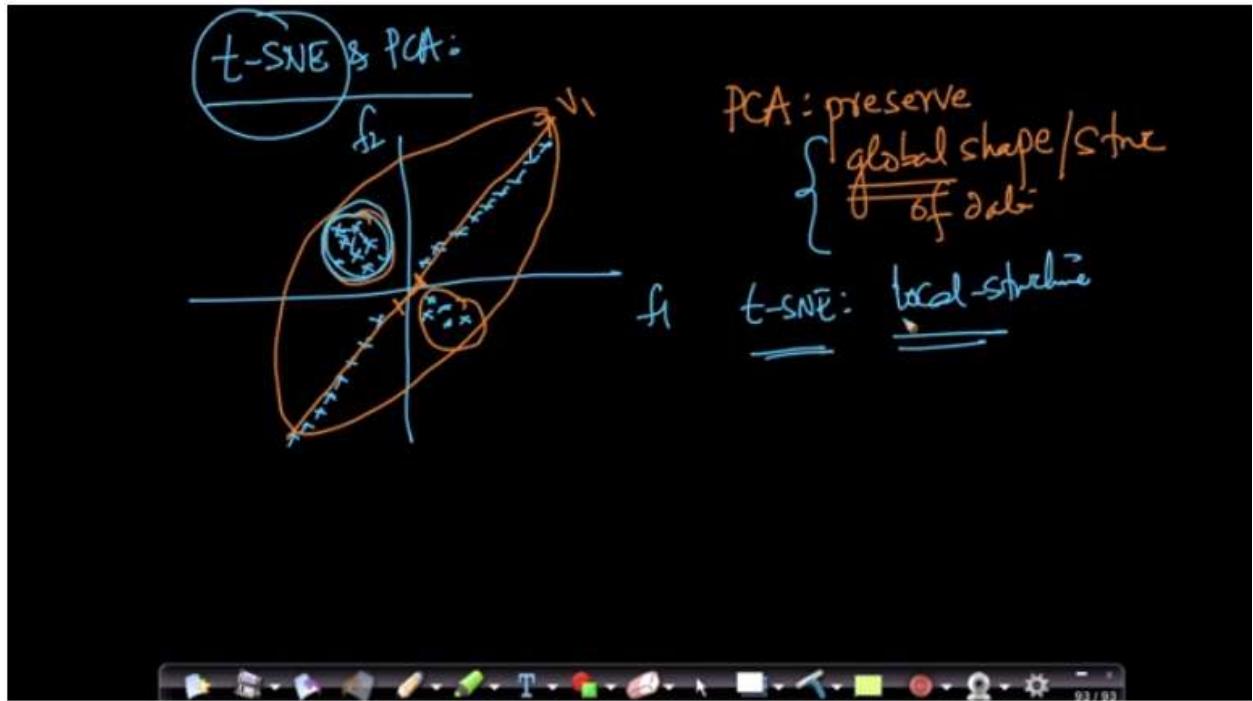
From the above plot, we can see that about 75% of variance can be explained if we reduce the data to 100 dimensions, similarly around 90% of variance can be explained when we choose d' to be 200 dimensions. Hence if we decide that it is ok to preserve 90% variance of our original data, we can then choose d' to be 200 and can reduce the data to 200 dimensions using PCA.

27.1 What is t-SNE?



Timestamp: 3:52

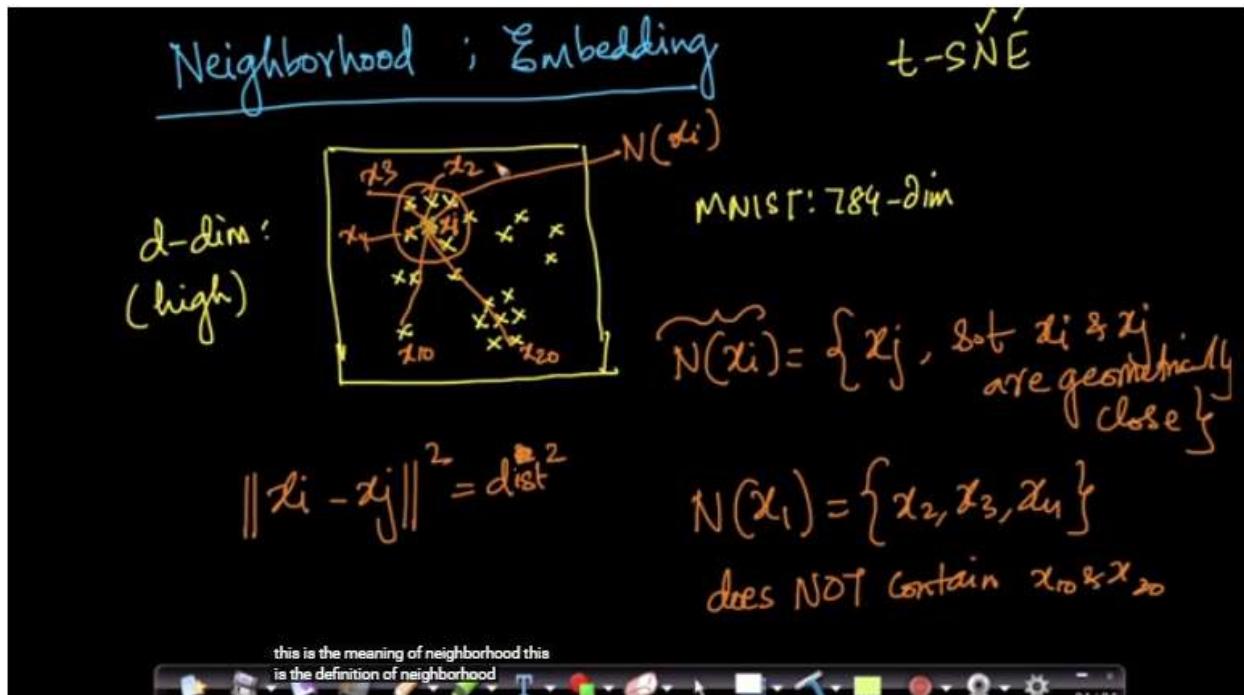
t-SNE is the state of the art technique for visualization. There exists other visualization techniques like MDS, sammon mapping, etc but t-SNE performs comparably well than most other techniques.



Timestamp: 6:21

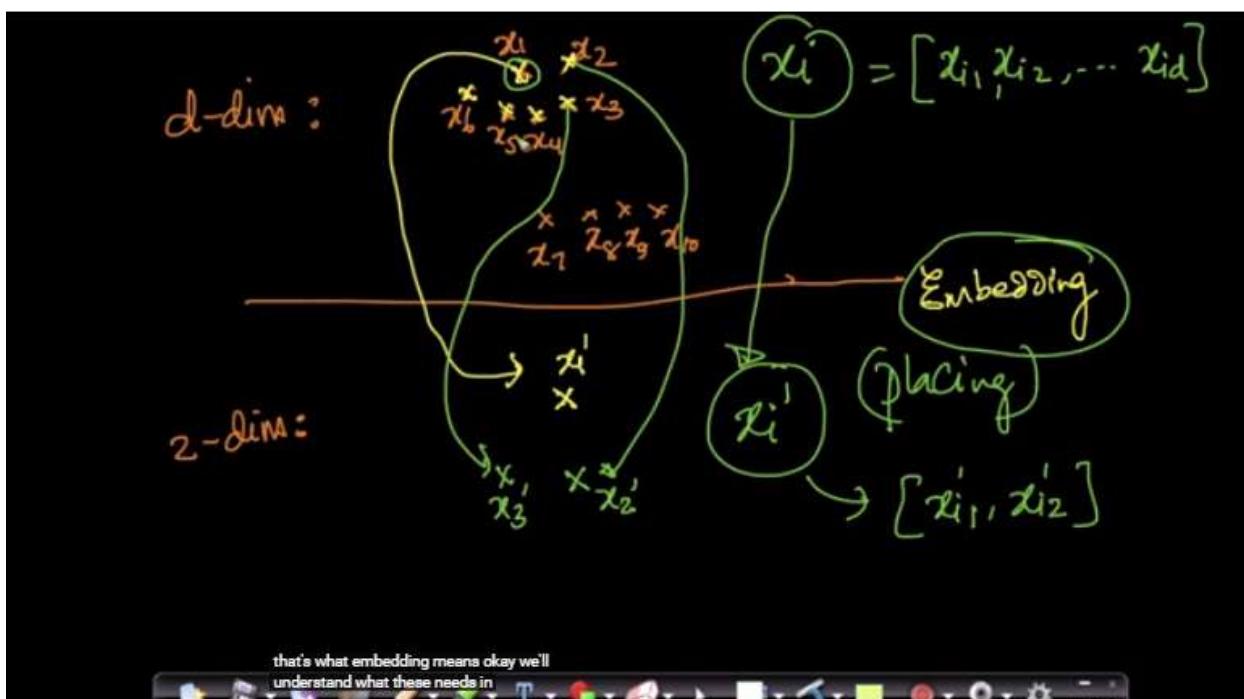
One of the key disadvantages of PCA that we discussed in “limitations of PCA” is that when they are nice clusters of points that are present in the original space when they are converted to lower dimensions, these get mixed up with other points. So PCA preserves only global structure and doesn't preserve local structure. By changing parameters in t-SNE we can make it preserve local structure as well.

27.2 Neighborhood of a point, Embedding



Timestamp: 3:40

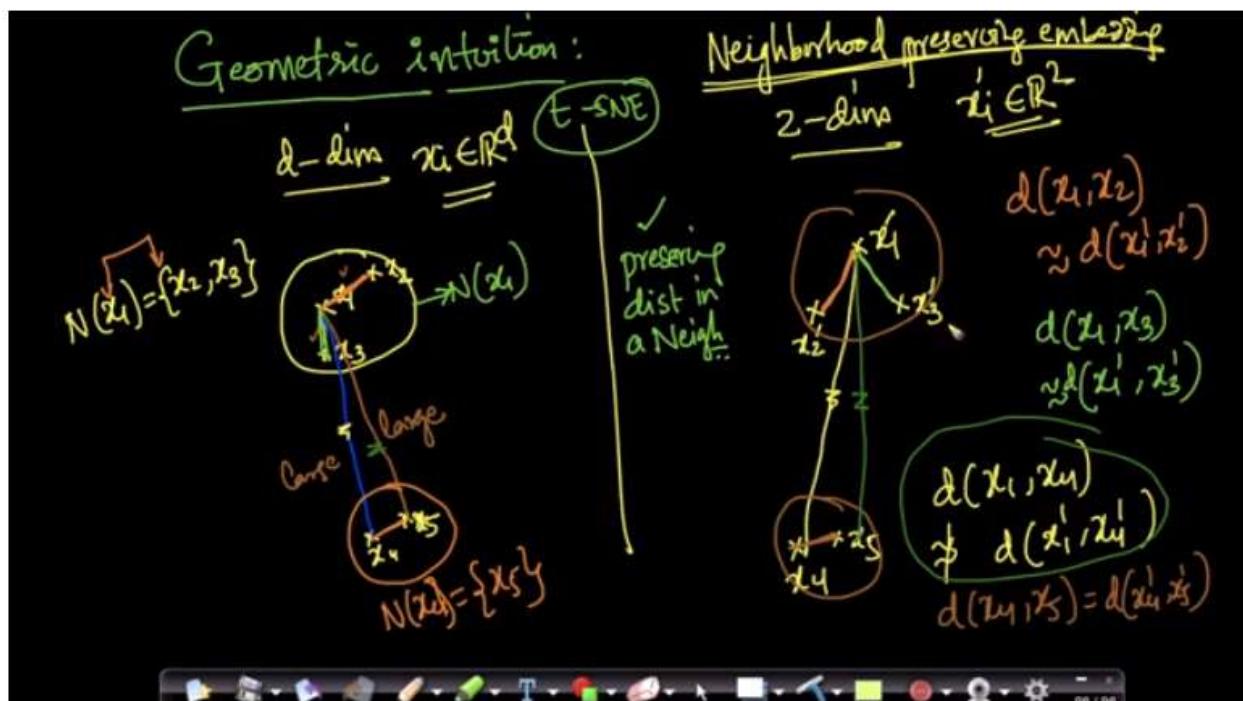
For any point, say x_1 , the points around x_1 are called the neighborhood of point x_1 . In our case the neighborhood of x_1 is the set of points x_2, x_3, x_4 , etc. The neighborhood of any point x_i is the set of all points that are geometrically close to the point x_i .



Timestamp: 6:14

For any point in the high dimensional space, if we find the corresponding point in the low dimensional space, then such a thing is called embedding.

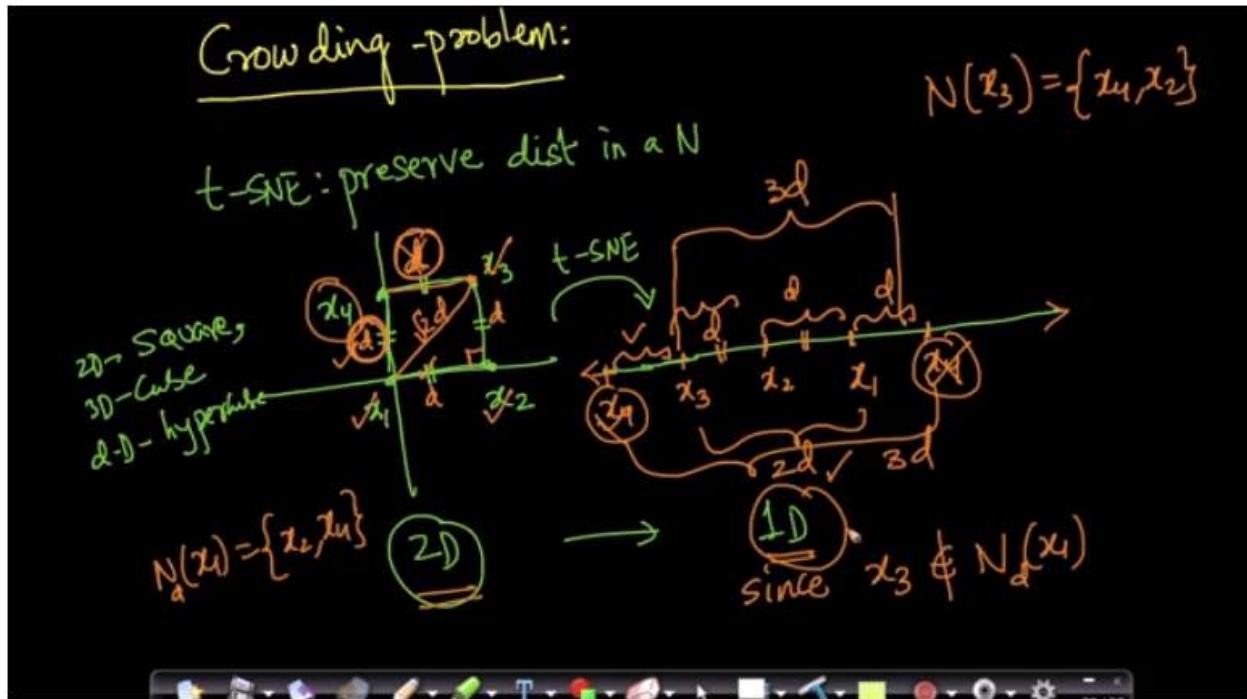
27.3 Geometric intuition of t-SNE



Timestamp: 6:18

Geometrically, what t-SNE does is, it tries to preserve the distance between the points in the same neighborhood even in the embeddings. What that means is that points that the distance between points belonging to the same neighborhood is almost similar to the distance between the points in the smaller dimension as well, the same is not guaranteed with the points that are not in the same neighborhood in the original space.

27.4 Crowding Problem

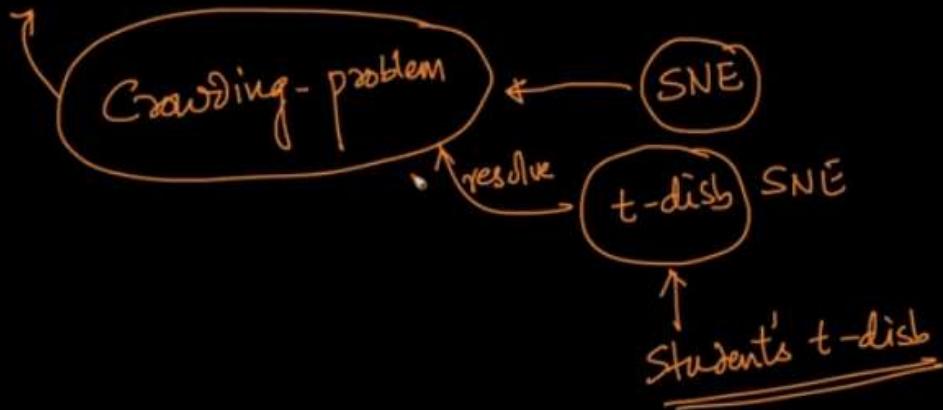


Timestamp: 5:47

Consider a scenario as above where we have 2-D dimensional data and we have to reduce it 1-D. Consider the points x_1, x_2, x_3 and x_4 in the original 2-D space and let us say that d is the neighborhood distance i.e two points with distance less than or equal to d , are said to be in the same neighborhood. So in the above figure x_4 has neighbors x_1 and x_3 , x_3 have neighbors x_4 and x_2 ,etc.

So in our 1-D embedding, we have to place x_2 and x_1 within a distance of d , since we have to preserve distance between points that belong to the same neighborhood, next we place x_3 within d distance of x_2 since x_2 and x_3 belong to same neighborhood, now we have to find a place for x_4 in the 1-D, but we cannot place it correctly. This is because if we place it in d distance from x_3 , then it would be of distance $> d$ from x_1 . Hence we won't be able to preserve the distance between x_4 and x_1 even though they belong to the same neighborhood. The same will be the case even if we try to place it at d distance from x_1 . Here we are suffering from crowding problem.

\times sometimes.
It is impossible to preserve dist in all the N



Timestamp: 7:44

Sometimes, it is impossible to preserve distances of all points in the neighborhood. This problem is called the crowding problem. SNE used to suffer from this crowding problem, later t-SNE was introduced to resolve this issue, t-SNE doesn't completely solve the problem, but it tries its best to minimize the errors due to this problem.

27.5 How to apply t-SNE and interpret its output

1. Those hyperparameters really matter *(perplexity, dev.)*

Let's start with the "hello world" of t-SNE: a data set of two widely separated clusters. To make things as simple as possible, we'll consider clusters in a 2D plane, as shown in the lefthand diagram. (For clarity, the two clusters are color coded.) The diagrams at right show t-SNE plots for five different perplexity values.

$n=100$

With perplexity values in the range (5 - 50) suggested by van der Maaten & Hinton, the diagrams do show these clusters, although with very different separations.

Handwritten notes on the right side of the screen:

- ① with multiple perplexity values ✓
- ② $\text{perp} = \# \text{data points}$
 $\text{perp} < \# \text{adjs}$ ↗ MESS

Timestamp: 15:30

Note: In all the examples of this video, even though t-SNE is used to visualize higher dimensional data in 2-D, we try to visualize the data from 2-D itself post applying t-SNE, to give us the intuition of how t-SNE works.

- 1) Notice that with multiple values of perplexity you get different visualizations and the visualizations are close to the original data only for perplexities 30 and 50. The lesson here is to run t-SNE with different values of perplexity to get the right visualization
- 2) Never run t-SNE with $\text{perplexity} = \text{total_number_of_points}$, running t-SNE with perplexity as total number of points will create a mess as it tries to preserve the distances of all the points not only the neighborhood when ran with perplexity as total number of points

points. Implementations can give unexpected behavior otherwise.

Each of the plots above was made with 5,000 iterations with a learning rate (often called "epsilon") of 10, and had reached a point of stability by step 5,000. How much of a difference do those values make? In our experience, the most important thing is to iterate until reaching a stable configuration.

iter | Step →

The images above show five different runs at perplexity 30. The first four were stopped before stability. After 10, 20, 60, and 120 steps you can see layouts with seeming 1-dimensional and even pointlike images of the clusters.

Timestamp: 16:49

Always run t-SNE for multiple iterations(step), until unless you see that the shapes are stable even with further iterations as shown in the figure.

points. Implementations can give unexpected behavior otherwise.

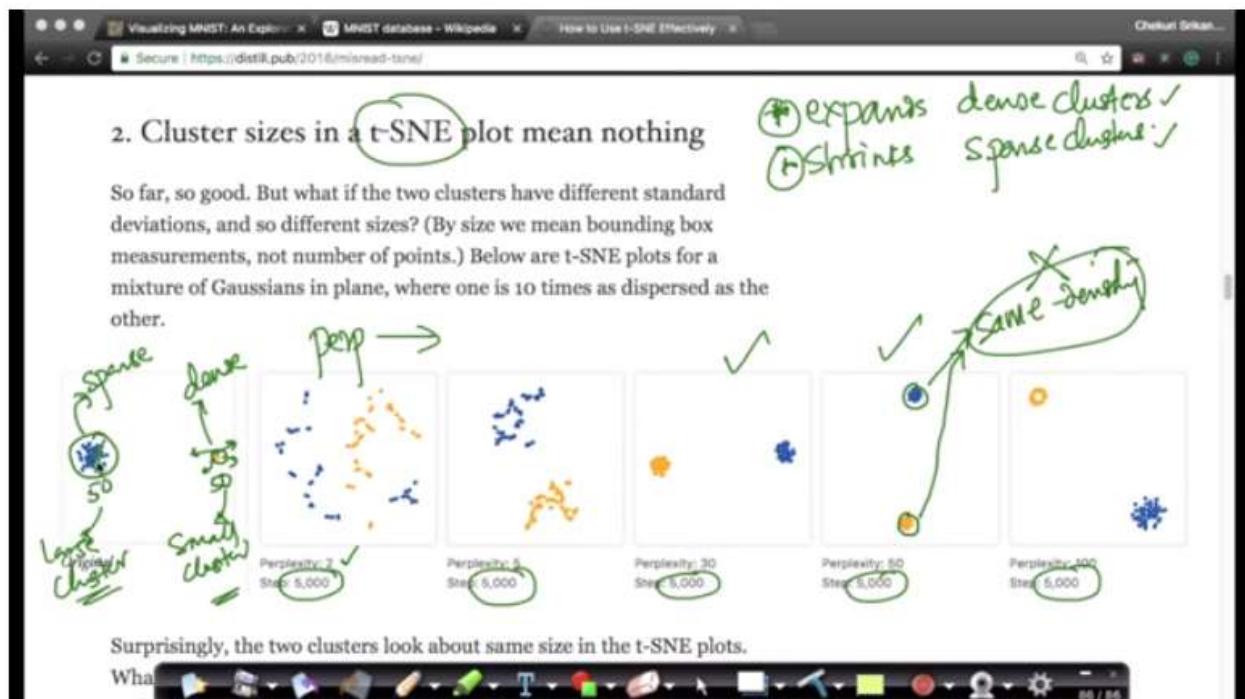
Each of the plots above was made with 5,000 iterations with a learning rate (often called "epsilon") of 10, and had reached a point of stability by step 5,000. How much of a difference do those values make? In our experience, the most important thing is to iterate until reaching a stable configuration.

D → P → ✓

The images above show five different runs at perplexity 30. The first four were stopped before stability. After 10, 20, 60, and 120 steps you can see layouts with seeming 1-dimensional and even pointlike images of the clusters.

Timestamp: 18:45

Always run t-SNE multiple times by fixing the perplexity and number of iterations. Since t-SNE internally is a stochastic algorithm, even with the same perplexity and number of iterations it may give slightly different results.



Timestamp: 22:07

You can't come with the same density conclusion with t-SNE. If two clusters have same density in the embeddings then there is no guarantee that they have the same density in the original space. t-SNE typically tends to expand dense clusters and shrink sparse clusters

increase to compensate. Here are the t-SNE diagrams for three Gaussian clusters with 200 points each, instead of 50. Now none of the trial perplexity values gives a good result.

It's bad news that seeing global geometry requires fine-tuning perplexity. Real-world data would probably have multiple clusters with different numbers of elements. There may not be one perplexity value that will capture distances across all clusters—and sadly perplexity is a global parameter. Fixing this problem might be an interesting area for future research.

Timestamp: 22:59

t-SNE doesn't preserve the distance between clusters. You can see in the above figure that cluster1 and cluster2 are closer than cluster3 but after running t-SNE , it is showing that they are of equal distances.

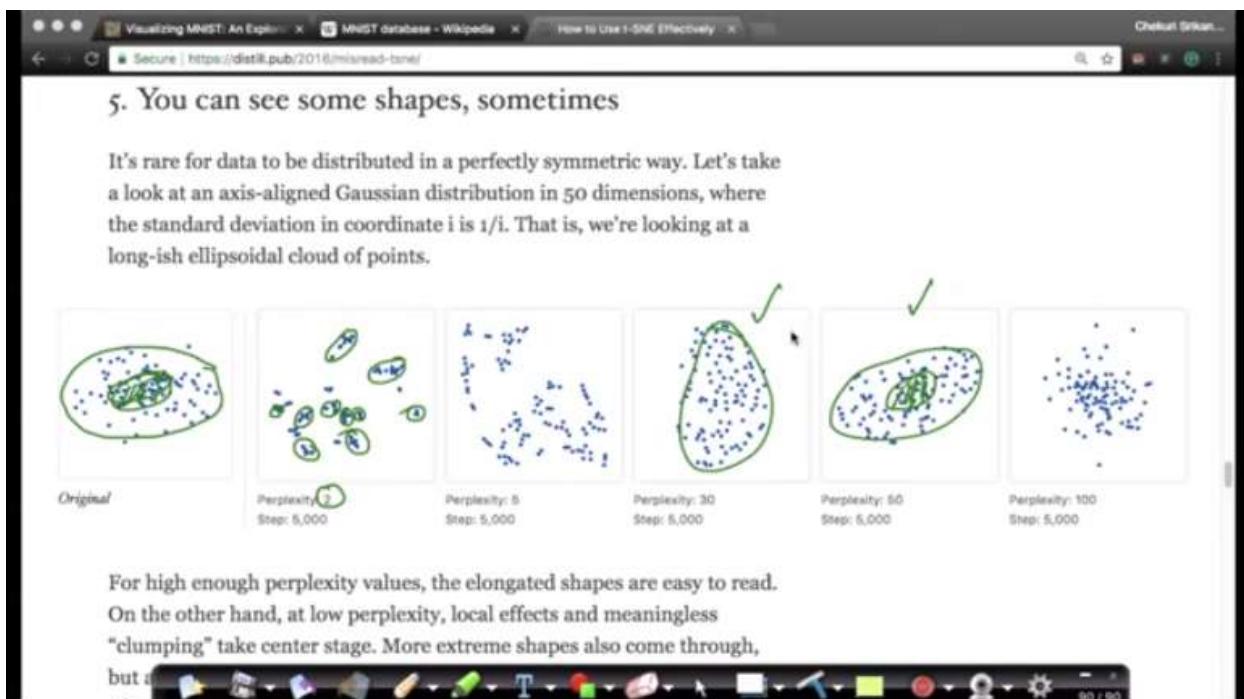
show genuinely random data, 500 points drawn from a unit Gaussian distribution in 100 dimensions. The left image is a projection onto the first two coordinates.

The plot with perplexity 2 seems to show dramatic clusters. If you were tuning perplexity to bring out structure in the data, you might think you'd hit the jackpot.

Of course, since we know the cloud of points was generated randomly, it has no statistically interesting clusters: those "clumps" aren't meaningful. If you look back at previous examples, low perplexity values often lead to this!

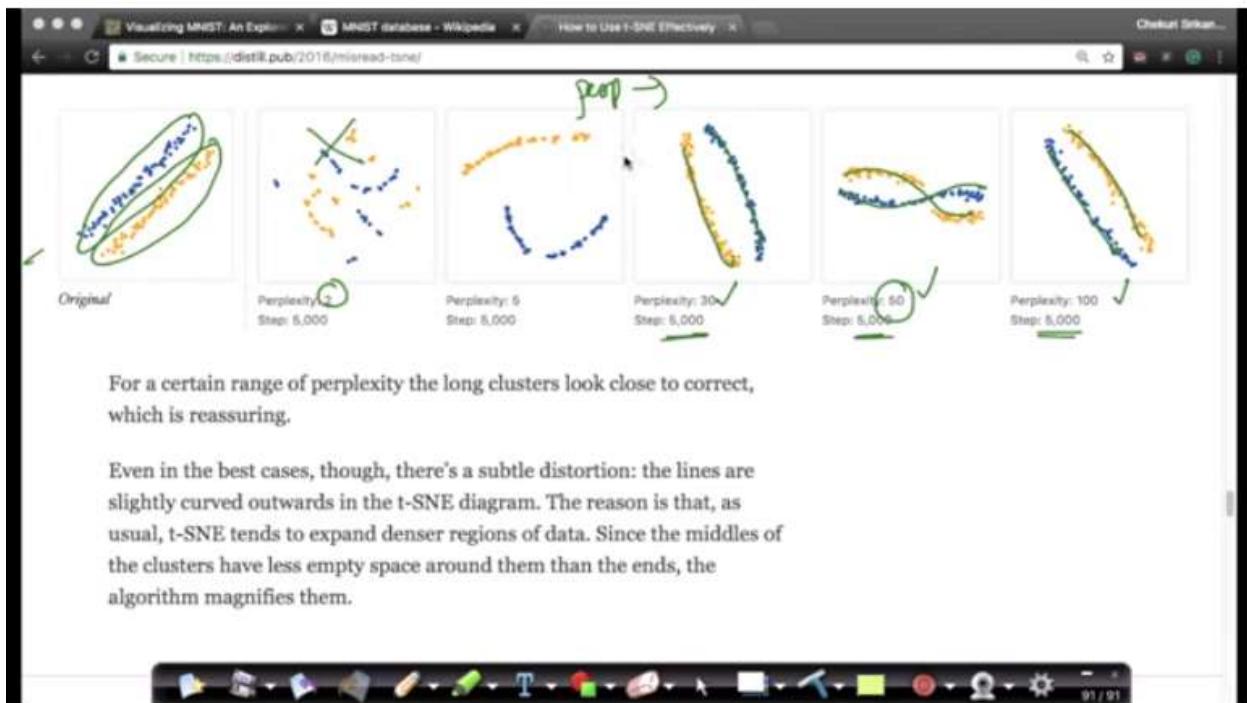
Timestamp: 27:51

With t-SNE you might come up with wrong conclusions if you run with a single perplexity value. In the above picture, we can see some clusters when run with perplexity=2, but the original data is completely random. Running with other perplexity values shows that the data is indeed random. Hence running t-SNE with different perplexity values is preferred.



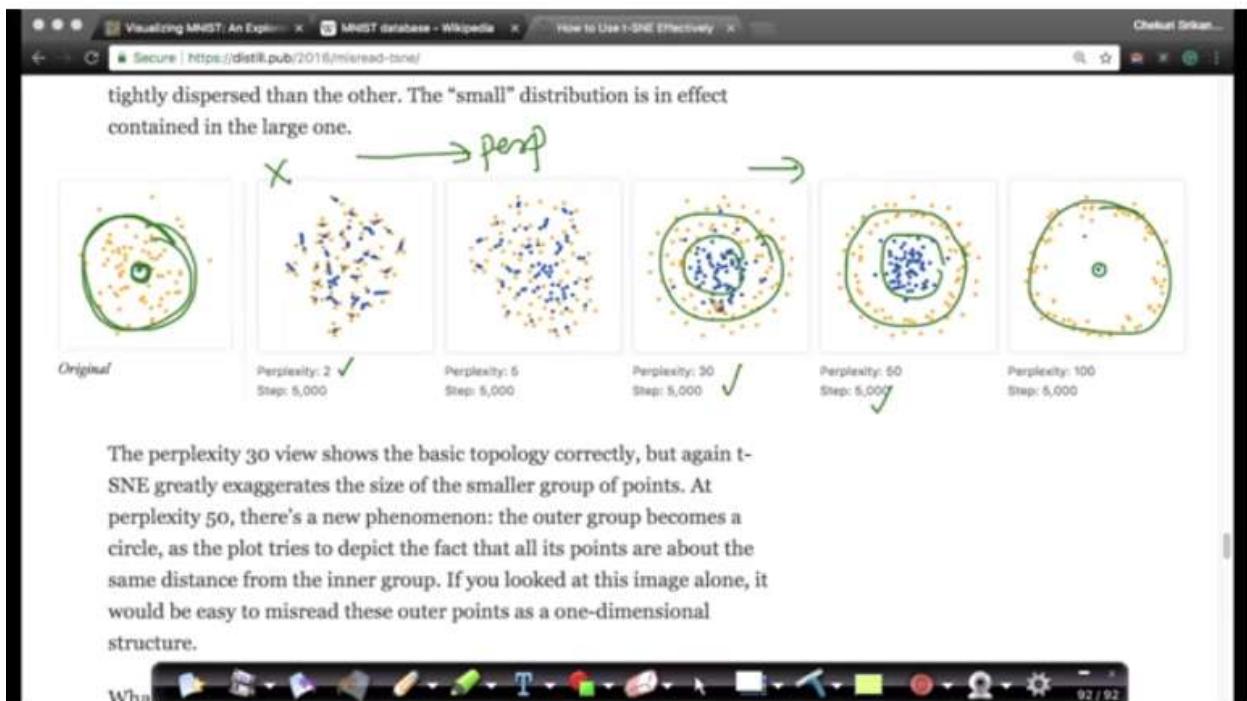
Timestamp: 28:52

Similar example, do not run t-SNE with single perplexity values rather run with different perplexity values as there is a risk of concluding that there are certain shapes of clusters with the points in our data when they are not.



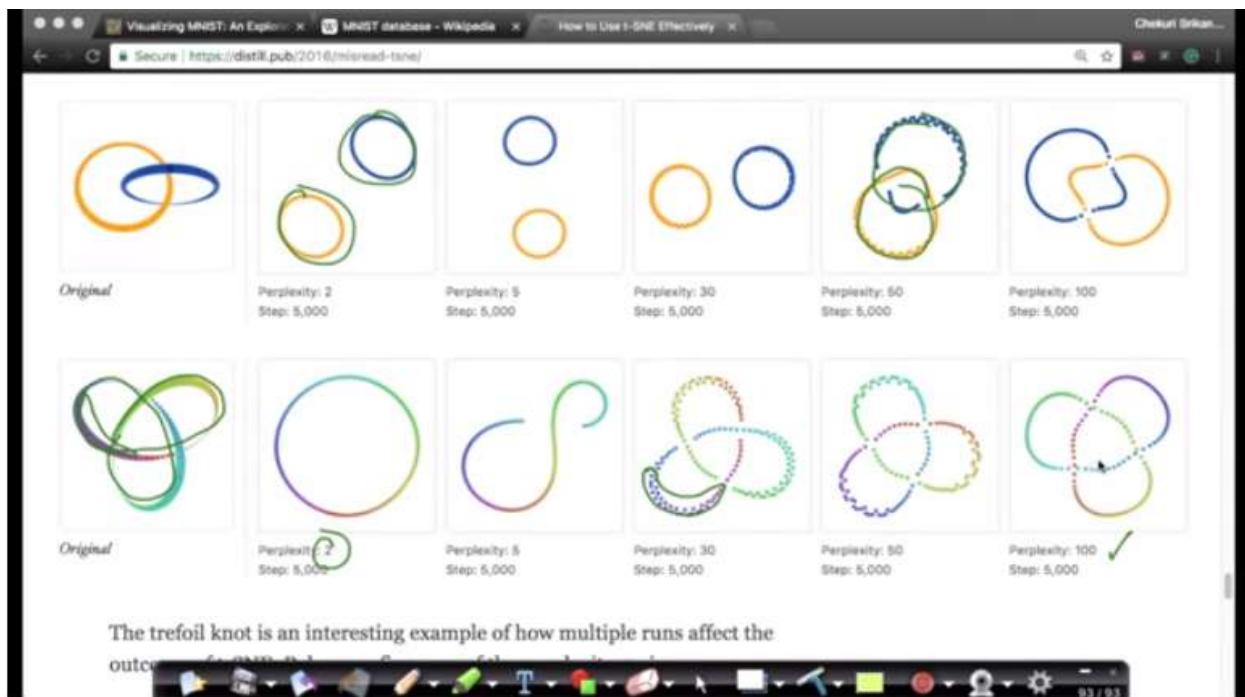
Timestamp: 30:36

Again, never run t-SNE with a single perplexity value, sometimes with some perplexity values we may get different shapes than the original data. Another takeaway is to run t-SNE multiple times with the same perplexity value and number of iterations.



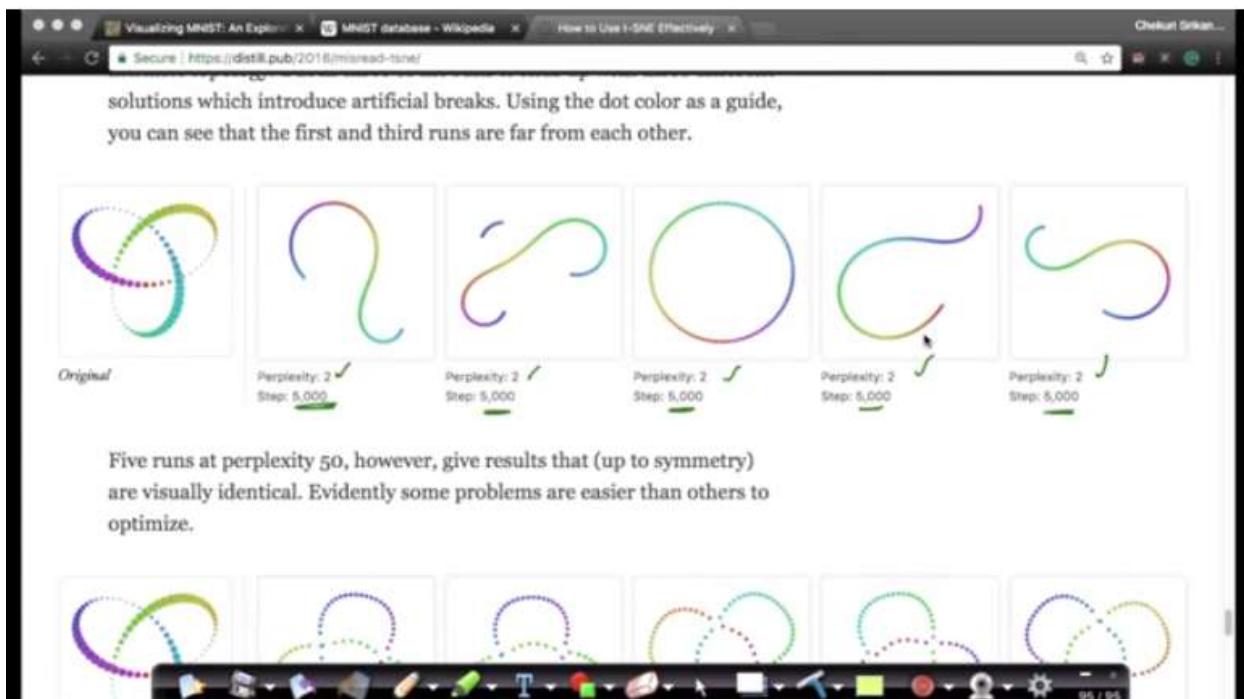
Timestamp: 31:59

Running with different perplexity values gives us the sense of overall topology of our data. We can see from the above data that as we increase the perplexity values we begin to realize that blue points lie inside orange points in the original data.

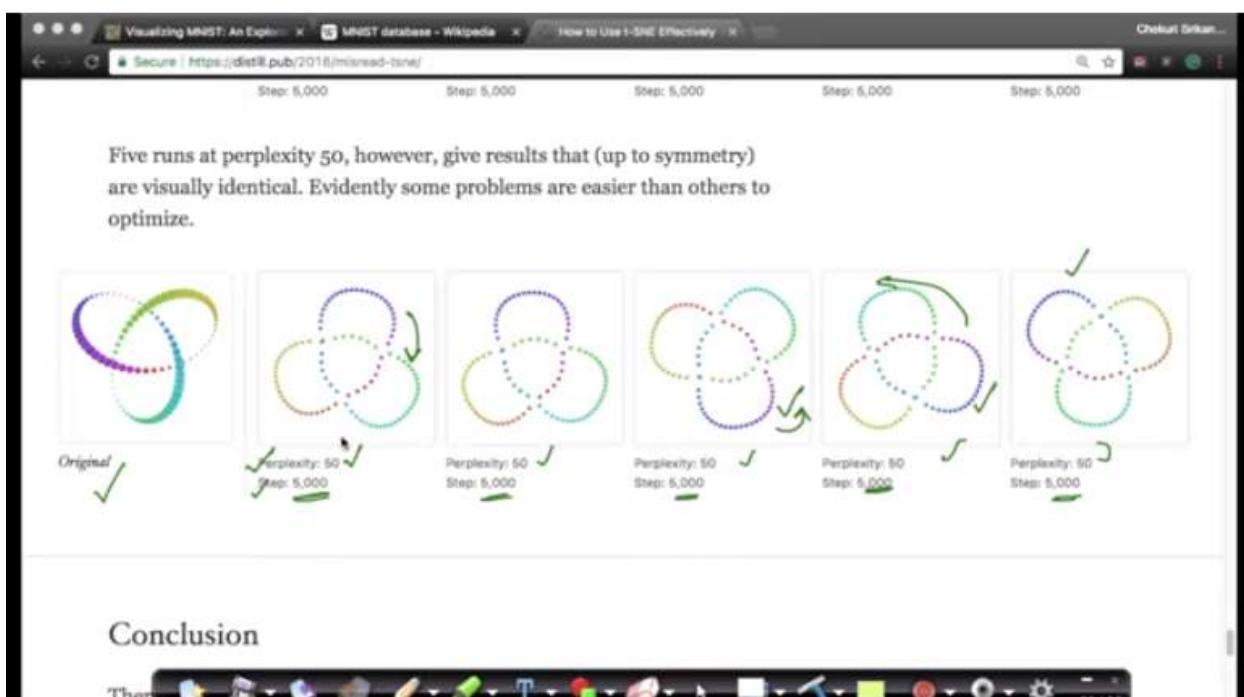


Timestamp: 32:38

From the above figure we can observe that with increasing perplexity, the topology of the visualization looks more and more similar to the topology of the original data.



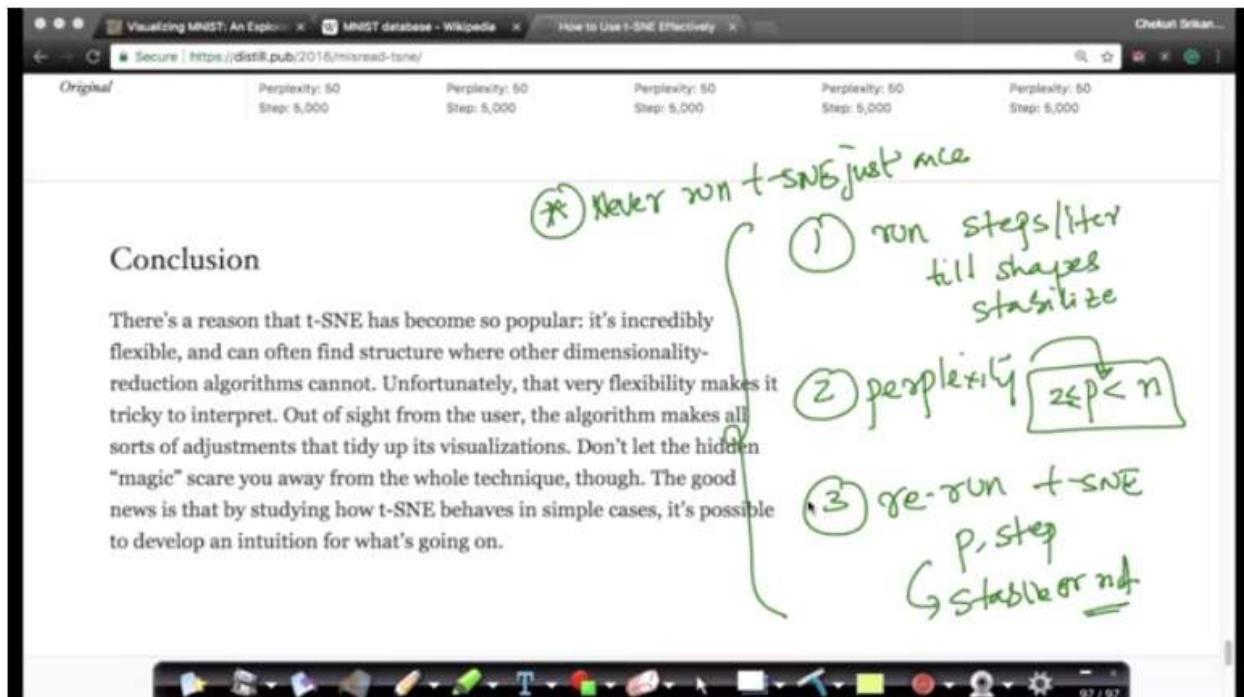
Timestamp: 33:39



Timestamp: 34:25

In the first figure we can see that with the same perplexity=2 and same number of iterations, we are seeing different shapes and the shapes are not stable. This means perplexity=2 is not the right perplexity value for the dataset.

Instead if we run t-SNE with perplexity=5 and same number of iterations, we see that the shapes are more or less stable. Hence it is always necessary to run t-SNE multiple times with the same perplexity value and number of iterations.



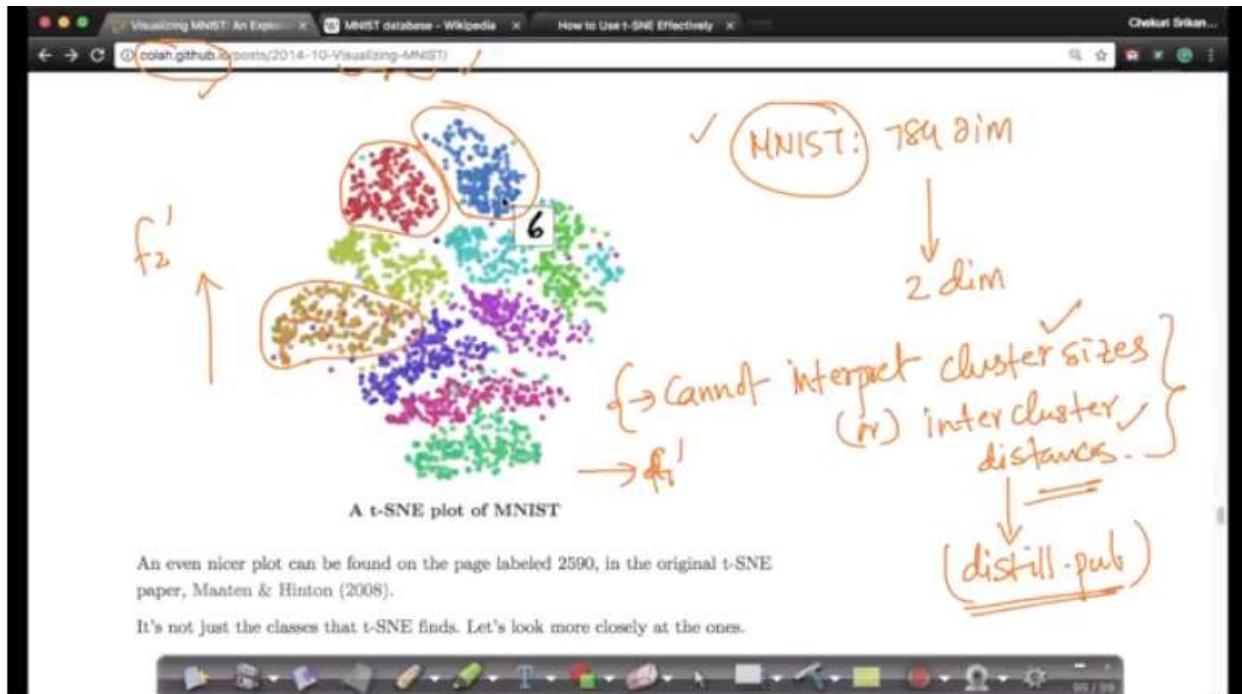
Timestamp: 36:03

To conclude,

- 1) Always run t-SNE for more iterations until the shapes stabilize
- 2) Try with different perplexity values in the range 2 to n (not including n as it creates mess).
- 3) Rerun t-SNE with the same number of iterations and perplexity value and check if the shapes are stable or not

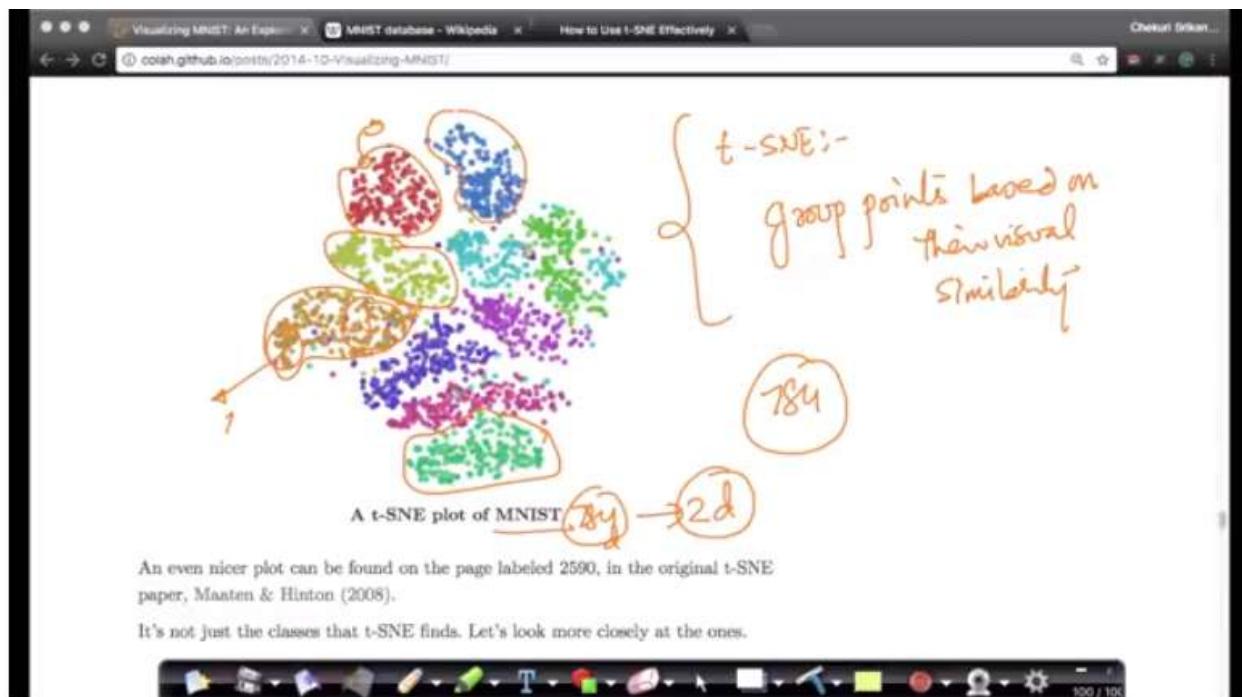
Never run t-SNE just once.

27.6 t-SNE on MNIST



Timestamp: 3:47

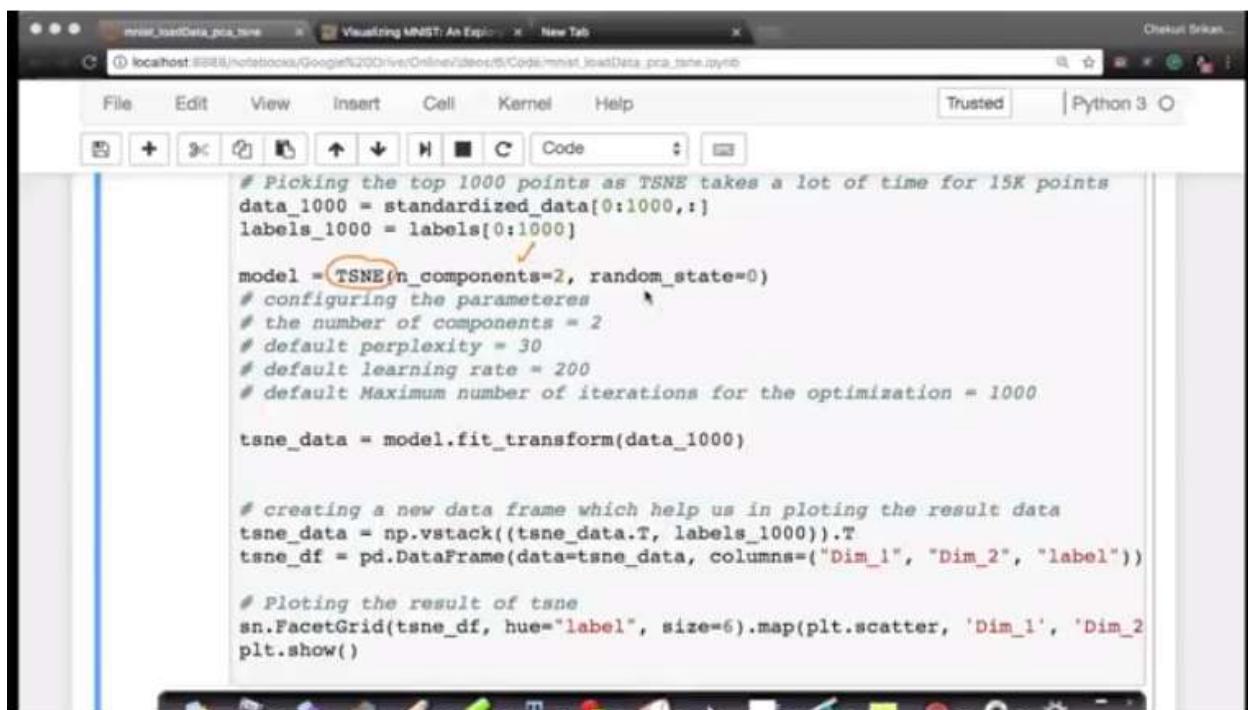
The above plot shows t-SNE applied on MNIST data. We cannot interpret cluster sizes or inter cluster distances using t-SNE.



Timestamp: 6:11

If we visualize the data points that are close together we can see that they are grouped based on visual similarity. If we see the visualization above we can see that most of the points of each class are well separated and grouped together, therefore we can easily fit simple machine learning models to separate these points.

27.7 code example of t-SNE



The screenshot shows a Jupyter Notebook interface with a single code cell containing Python code for performing t-SNE on the MNIST dataset. The code is annotated with several yellow arrows pointing to specific parts of the code, likely highlighting steps or parameters being explained in the video.

```
# Picking the top 1000 points as TSNE takes a lot of time for 15K points
data_1000 = standardized_data[0:1000,:]
labels_1000 = labels[0:1000]

model = TSNE(n_components=2, random_state=0)
# configuring the parameters
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

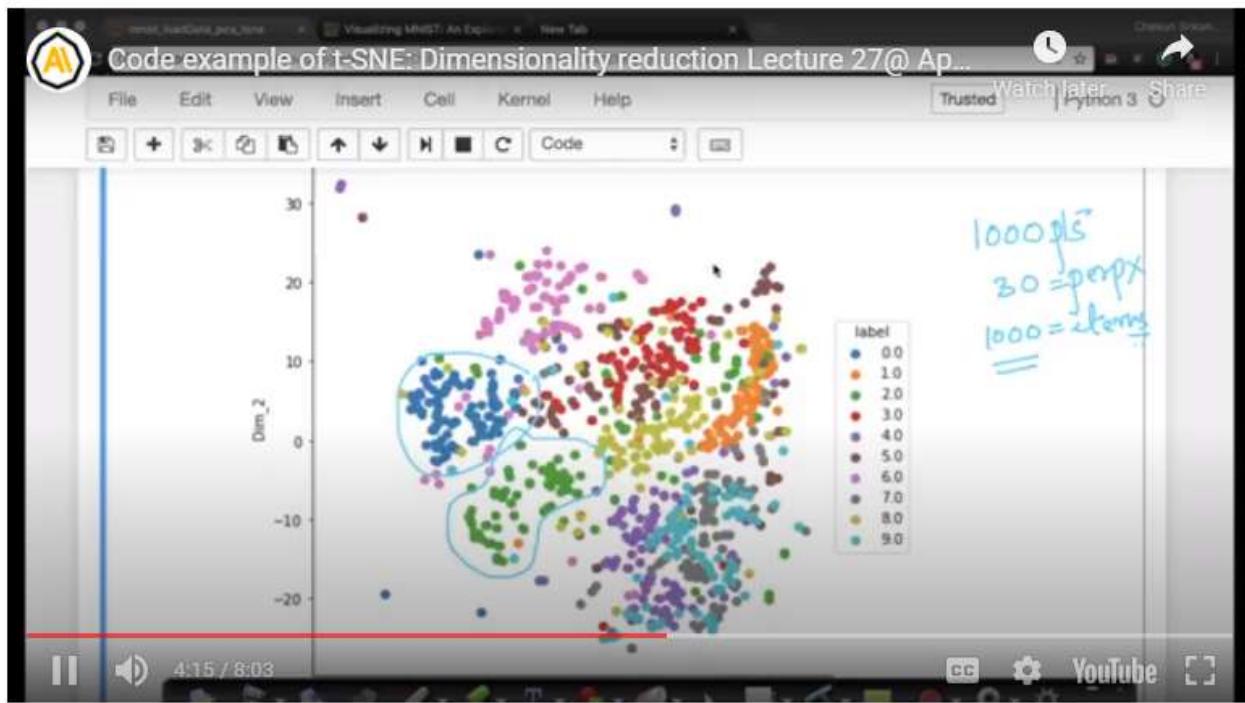
tsne_data = model.fit_transform(data_1000)

# creating a new data frame which help us in plotting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=["Dim_1", "Dim_2", "label"])

# Plotting the result of tsne
sn.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2')
plt.show()
```

Timestamp: 1:39

t-SNE was tried in the first 1000 data points of MNIST dataset using the code above with default values of perplexity and number of iterations which are 30 and 1000 respectively, we got the below plot.



Timestamp: 4:14

Now t-SNE is run with perplexity=50, with 1000 iterations using the below code

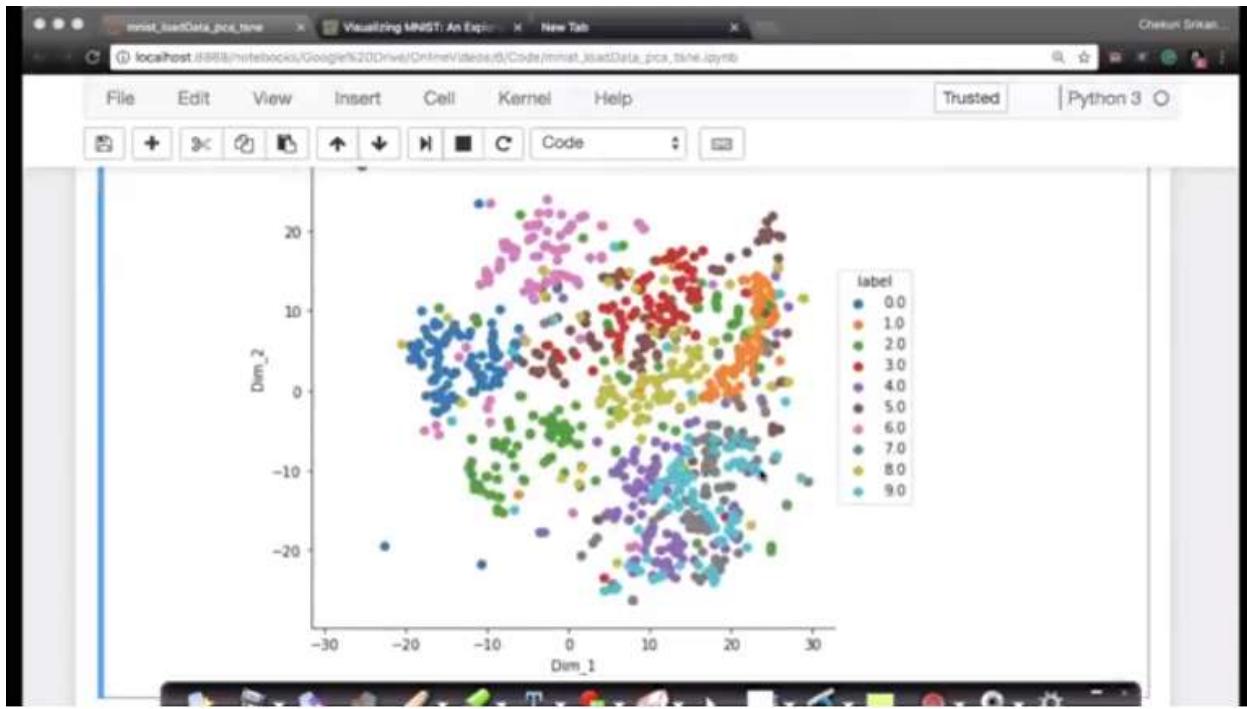
```
In [21]: model = TSNE(n_components=2, random_state=0, perplexity=50)
tsne_data = model.fit_transform(data_1000)

# creating a new data frame which help us in plotting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=["Dim_1", "Dim_2", "label"])

# Plotting the result of tsne
sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2')
plt.title('With perplexity = 50')
plt.show()
```

Timestamp: 4:38

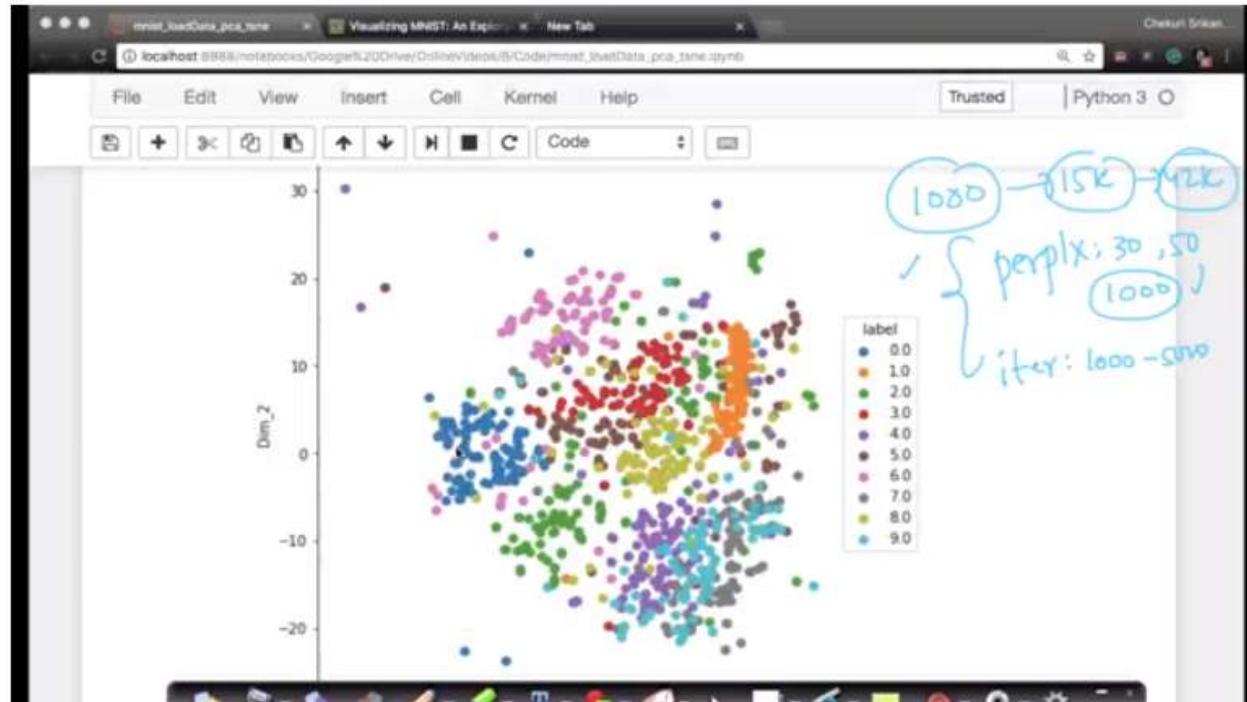
We got the below visualization for the same as below



Timestamp: 5:00

This plot looks similar to the plot with perplexity=30, the visualization is more or less similar.

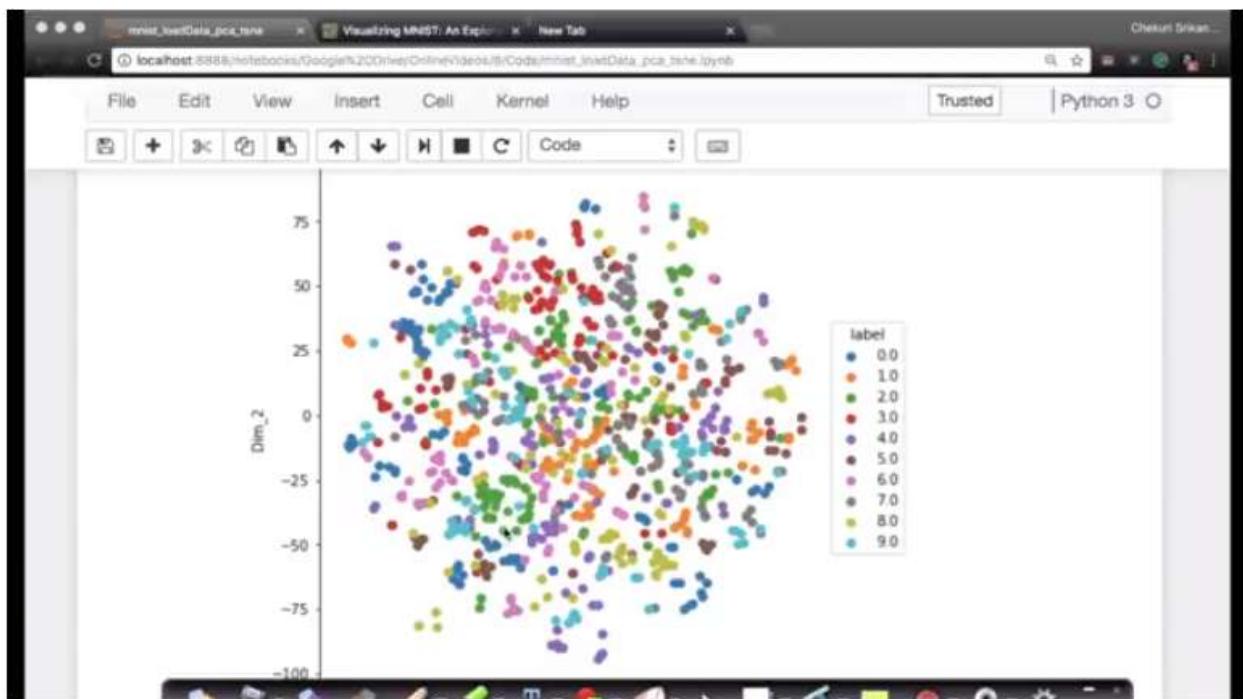
Now t-SNE is run with perplexity=50 and the number of iterations to be 5000, we got the below visualization



Timestamp: 5:26

Even with 5000 iterations we got similar visualization meaning that our visualization is stable. Notice that with only using 10K data points we got the above visualizations, with using more data points the visualization would be much more better.

Now we try with perplexity=2, then we get the below visualization.



Timestamp: 6:49

From the above visualization, we can see that perplexity=2 may not work well as it fails to separate the data points.

28.1 Dataset Overview: Amazon Fine Food Reviews (EDA)

Important Links

Data Source

<https://www.kaggle.com/snap/amazon-fine-food-reviews>

Blog on EDA

https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualizatio_n/

Ipython Notebook

https://colab.research.google.com/drive/1FCQfzaYb-yyDluF5RKTFa0VPnsYJZyR8?usp=drive_open

Note

In order to download the dataset from Kaggle, you first have to login to Kaggle and then download it.

Dataset Description

The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454

Number of users: 256,059

Number of products: 74,258

Timespan: Oct 1999 - Oct 2012

Number of Attributes/Columns in data: 10

Attribute Information

1. Id - Unique Identifier given to each review
2. ProductId - Unique Identifier for the product
3. UserId - Unique Identifier for the user
4. ProfileName - Customer/User's name
5. HelpfulnessNumerator - number of users who found the review helpful
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
7. Score - rating between 1 and 5
8. Time - timestamp at which the review was posted on the website
9. Summary - brief summary of the review
10. Text - text of the review

Problem Objective

We have to determine if the given review is positive or negative.

How to categorize a given review as positive/negative?

Categorization of the reviews is done on the basis of the 'Score' column values. If a given review has a score value of 4 (or) 5, then it is considered as a Positive review, and if the score value is 1 (or) 2, then it is considered as a Negative Review.

A score value of 3, is considered to be a neutral review, but for the time being, we shall ignore the neutral reviews because, considering the neutral reviews will pose our problem as a multi-class classification. But as we are looking to pose our problem as a binary classification problem, we consider only the positive and the negative reviews.

Loading the Dataset

The dataset is given in two forms.

- 1) CSV File format (with the name Reviews.csv)
- 2) SQLite Database

The dataset is already present in the SQLite database, so that it can be directly loaded into a pandas dataframe, and from there we can perform analysis and visualizations on the data easily.

But in case, if you do not want to perform any analysis or visualizations on the data, but just want to have a look at the data, then instead of loading the data into a pandas dataframe every time, you can directly look into the 'Reviews.csv' file.

Below is the line of code that was discussed at the timestamp 18:40, which establishes the connection with the database.

```
con = sqlite3.connect('database.sqlite')
```

Note: If the 'database.sqlite' file is present in the same folder where your ipython notebook is present, then you can pass only the file name as a value to the sqlite3.connect(). If not, then you have to specify the entire path as well.

Below is the line of code that was discussed at the timestamp 19:15, which loads the dataset into a pandas dataframe.

```
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 LIMIT 5000""", con)
```

Note: Here as we are loading the dataset from the database, we are using the method `pandas.read_sql_query()` and are passing the SQL query and the sqlite connection object. If we had to load the dataset from a CSV file, then we would have used `pandas.read_csv()` with the path to the CSV file as an argument.

The data in our database is stored in the table ‘Reviews’, and we are selecting only those rows whose score value is not equal to 3.

The below code snippet was discussed starting from the timestamp 20:33 and here we are relabelling the scores 4 and 5 as ‘Positive’, and 1 and 2 as ‘Negative’.

```
# Give reviews with Score>3 a positive rating, and reviews with a score<3 a negative rating.
def partition(x):
    if x < 3:
        return 0
    return 1

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print("Number of data points in our data", filtered_data.shape)
filtered_data.head(3)
```

We are first loading the values of the ‘Score’ column into a series variable ‘actualScore’, and then using the `map()` function, we apply relabelling the values. The code for relabelling is defined in the function `partition()`.

28.2 Data Cleaning: Deduplication

It is observed that the given dataset had many duplicate entries. Hence it is necessary to remove duplicates in order to get unbiased results for the analysis of the data.

In our data, we are terming two or more reviews as duplicates, if they have the same values for the columns ‘UserId’, ‘ProfileName’, ‘Time’, and ‘Text’. Before we delete the duplicate entries, we first have to sort all the reviews on the basis of the ‘ProductId’ column using `pandas.sort_values()`. After sorting the reviews, we drop the duplicates using `panda.drop_duplicates()`.

Below is the line of code that was discussed starting from the timestamp 7:00 which sorts all the reviews on the basis of the ‘ProductId’ column. After sorting the reviews, we are dropping the duplicates and it was discussed at the timestamp

```
#Sorting data according to ProductId in ascending order
sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')

#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='first', inplace=False)
final.shape

(4986, 10)
```

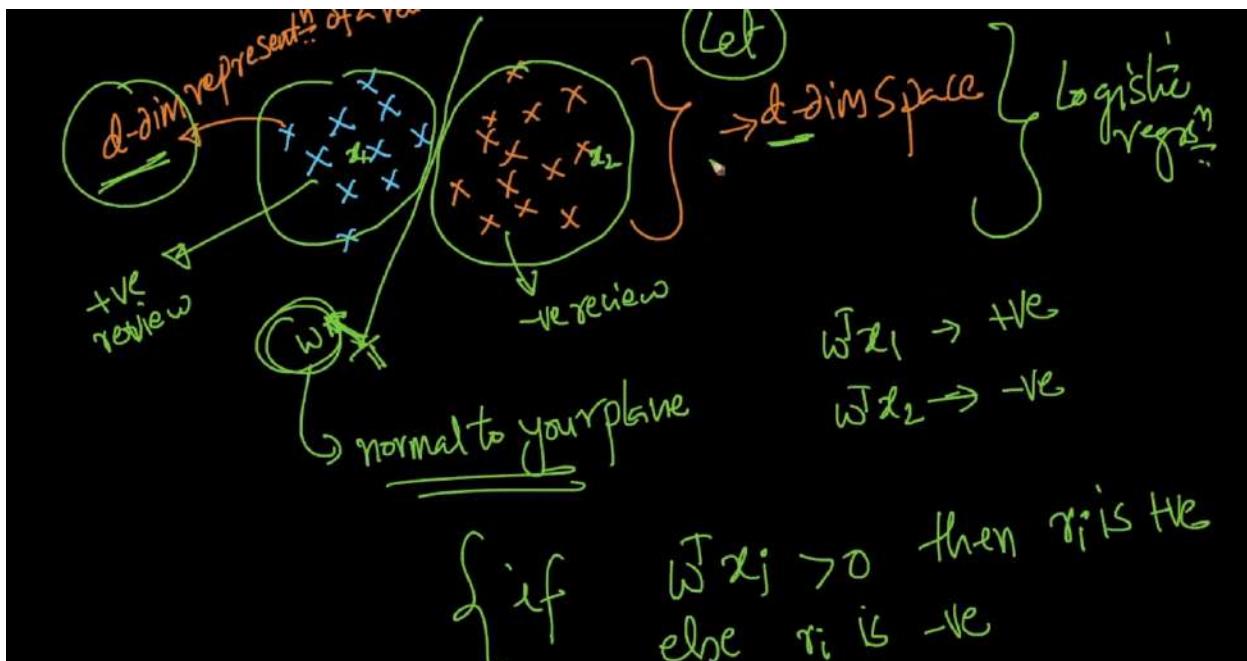
Deduplication is a technique used to improve storage utilization and can also be applied to the network data transfers to reduce the number of bytes that must be read.

28.3 Why convert a text to a vector?

Given any problem, if we are able to convert the data into vector form, we can leverage the whole power of Linear Algebra.

When we are given the text data, if we could convert it into d-dimensional vector format, and plot those vectors/points in d-dimensional coordinate space, we can find a hyperplane that could separate the points/vectors belonging to different classes.

Below is the representation that was explained starting from the timestamp 3:25.



This is a d-dimensional representation of each vector. Each vector represents a d-dimensional review. Here the vector 'w' is normal to the hyperplane ' π '.

If $w^T x_1 > 0$, then we can say that the vector 'w' is in the same direction as that of ' x_1 ' and. If $w^T x_2 < 0$, then we can say that the vector 'w' is in the direction opposite to that of ' x_2 '.

For a given query point ' x_i ',

If $w^T x_i > 0$, then the point ' x_i ' is classified as positive.

If $w^T x_i < 0$, then the point ' x_i ' is classified as negative.

Properties required to convert a text into a d-dimensional vector

If we have 3 vectors ' r_1 ', ' r_2 ' and ' r_3 ' which are semantically similar, then

If $\text{similarity}(r_1, r_2) > \text{similarity}(r_1, r_3)$, then $\text{distance}(v_1, v_2) < \text{distance}(v_1, v_3)$

$v_1 \rightarrow$ Vector form of the review ' r_1 '.

$v_2 \rightarrow$ Vector form of the review ' r_2 '.

It means, if the reviews ' r_1 ' and ' r_2 ' are similar, then the vectors ' v_1 ' and ' v_2 ' must be close.

If $\text{similarity}(v_1, v_2) > \text{similarity}(v_1, v_3)$, then $\text{length}(v_1-v_2) < \text{length}(v_1-v_3)$.

28.4 Bag of Words (BOW)

The below techniques are used to convert a text to a vector.

- 1) Bag of Words (BOW)
- 2) Term Frequency - Inverse Document Frequency (TF-IDF)
- 3) Average Word2Vector
- 4) TF-IDF Weighted Average Word2Vector

Bag of Words (BOW)

Consider the below 4 reviews as an example.

- r_1 : this pasta is very tasty and affordable
 r_2 : this pasta is not tasty and is affordable
 r_3 : this pasta is delicious and cheap
 r_4 : pasta is tasty and pasta tastes good

Steps in Bag of Words

- 1) Construct a dictionary (not Python data structure dictionary) which is going to be a set of all the words that are present in all the reviews. For this example, all the unique words present in all the reviews are to be added to a new set. (All these words should be taken only once)
- 2) Construct a vector for every review. In this problem, we are using the word "review", but the official terminology in NLP is a "**document**".

For example, we have to construct the vector ' V_1 ' from the review ' r_1 '. Let us assume we have 'n' documents/reviews and the total number of unique words in all the 'n' documents is 'd'.

We then have to construct a d-dimensional vector for every document. The vector has to be initialized with '0' in all dimensions and each cell is associated with each dimension and has to be filled with the number of times the corresponding word occurs in the given document/review. Each word is considered as a different dimension here.

Let us look at converting the above given 4 reviews into vector form. We'll have the dimensions as shown below.

this	pasta	is	very	tasty	and	affordable	not	delicious	cheap	tastes	good

All these features/words are present across the corpus. Now we shall vectorize the given 4 reviews according to these dimensions.

	this	pasta	is	very	tasty	and	affordable	not	delicious	cheap	tastes	good
r1:	1	1	1	1	1	1	1	0	0	0	0	0
r2:	1	1	2	0	1	1	1	1	0	0	0	0
r3:	1	1	1	0	0	1	0	0	1	1	0	0
r4:	0	2	1	0	1	1	0	0	0	0	1	1

In the above vectors of the reviews, the numbers denote the number of occurrences of that particular word in that review.

Note: The collection of all the documents(here reviews) is called **Corpus**.

The main objective of bag of words is that if documents are more similar semantically, then their corresponding vectors will be closer.

The result of Bag of Words is always a **Sparse Vector**.

Sparse Vector and Dense Vector

A vector is said to be a sparse vector, if most of the dimensions in it have 0 as value.

A vector is said to be a dense vector, if most of the dimensions in it have non zero values.

Note: Let us calculate the length of the difference between two vectors.

$$\text{length}(\mathbf{V}_1 - \mathbf{V}_2) = \|\mathbf{V}_1 - \mathbf{V}_2\| = \sqrt{(2-1)^2 + (0-1)^2 + (1-0)^2} = \sqrt{1 + 1 + 1} = \sqrt{3}$$

Note: One of the disadvantages with the Bag of Words approach is that, even after vectorization, if we observe a difference of a few bits between the two given reviews, but still the semantic meanings could be quite opposite. For example, if we clearly observe the reviews 'r₁' and 'r₂', we see a difference of only two words among them, but their semantic meanings are quite opposite.

Variations of Bag of Words

The so far discussed approach of Bag of Words is also known as **Count Bag of Words**. There is another variation of Bag of Words, which is called **Binary Bag of Words** (or) **Boolean Bag of Words**.

In the Binary/Boolean Bag of Words, unlike the Count Bag of Words, we do not have the count of occurrences of each word in the corpus as a dimension magnitude in

their vectors. Here it just indicates whether the words in the corpus occur in the documents or not in the form of '1' and '0'.

In the case of the Binary Bag of Words, the distance between the vectors is the square root of the total number of different values in the magnitude of each dimension in both the vectors.

length($V_1 - V_2$) = $\|V_1 - V_2\| = \sqrt{\text{number of different values in the magnitudes of each dimension}}$

Note: If there is no duplicate word occurring in a vector form among all the reviews/documents in the corpus, then both Count Bag of Words and Binary Bag of Words will give the same performance and the same result.

When to choose Count Bag of Words and when to choose Binary Bag of Words?

Choosing one among the two is problem specific. In some contexts, the information of presence or absence of a word is sufficient for the model to work well. In such a case, Binary Bag of Words is a good choice.

In other contexts, the count of the number of times a word occurs matters a lot. Here the Count based Bag of Words carries more information than the Binary Bag of Words. So the count Bag of Words typically tends to outperform the Binary Bag of Words.

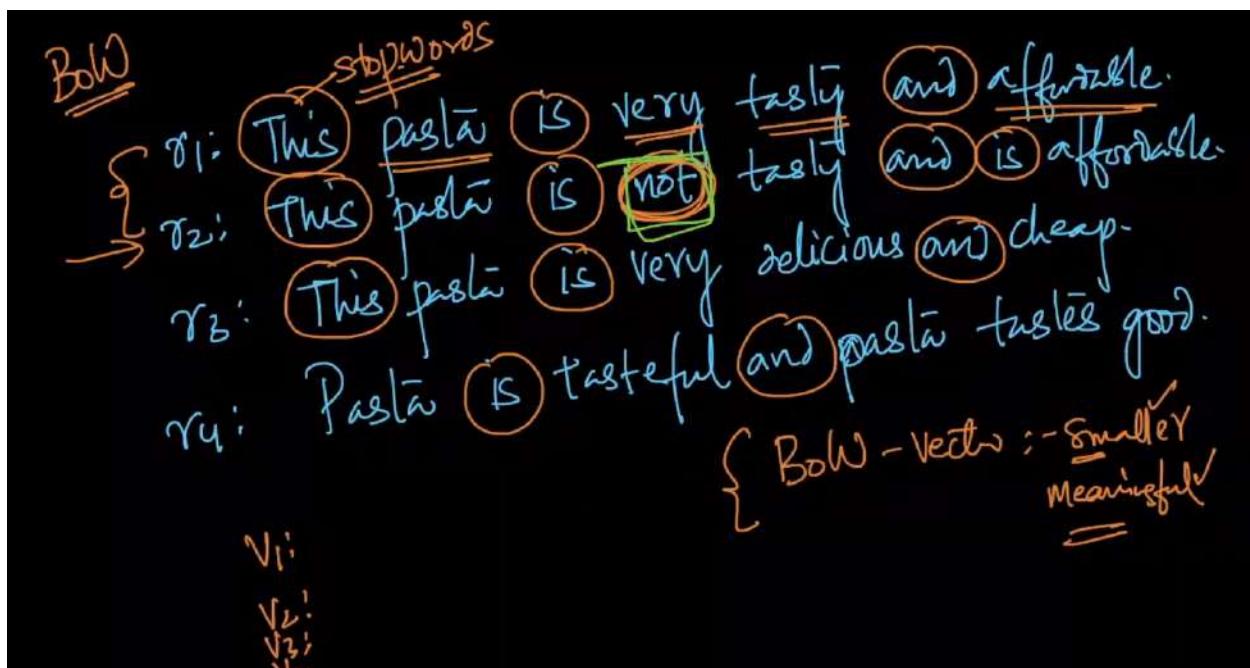
If we want to determine if a text review is "positive" or "negative", the presence of words like "bad" or "terrific" is good enough to predict the class label. This context is one of the best examples where Binary BOW based encoding can be used.

On the other hand, if we have more classes like "Very Positive", "Positive", "Negative", "Very Negative", then the number of times words like "terrific", "bad", "great", "terrible" occur could help us determine the extent of positivity and negativity helping us classify better. This context is one of the best examples where Count BOW based encoding can be used.

28.5 Text Preprocessing: Stemming, Stop-word Removal, Tokenization, Lemmatization

In a document, there will be certain words which make more sense about the data and there will be certain words which do not make much sense about the data. These words that do not make much sense about the data are present in the document just for sentence completion. Such words are called **Stopwords**.

These stopwords can be removed in order to make the bag of words vector smaller and more meaningful. Removal of stopwords is one of the text preprocessing steps and it has to be performed only when needed.



Steps in Text Preprocessing

- 1) Stopword Removal
- 2) Conversion of all words into lowercase
- 3) Stemming

In Stemming, the words like 'taste', 'tasteful', 'tastes' are reduced to the root form 'tast'. There are many stemming algorithms in Natural Language Processing (NLP). Two of the majorly used Stemming algorithms are PorterStemmer and SnowballStemmer. SnowballStemmer is much more powerful than PorterStemmer.

4) Lemmatization

Lemmatization is the algorithmic process of determining the lemma of a word based on its intended meaning. Unlike stemming, lemmatization depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document.

Note:

Let us look at the reviews 'r1' and 'r3' given below.

r₁: this pasta is very tasty and affordable

r₃: this pasta is delicious and cheap

These two reviews give the same meaning, but in BOW vectorization, the words 'tasty' and 'delicious' are treated as two different features. Similarly the words 'affordable' and 'cheap' are treated as two different features. This is the main disadvantage with the Bag of Words approach, as it doesn't preserve the semantic meaning.

The semantic meanings of the words are taken into consideration in the **Word2Vec vectorization techniques**. So finally using **Text Preprocessing + Bag of Words**, we are converting the text into a 'd' dimensional vector that could not guarantee the semantic meanings of the words. Bag of words doesn't take the semantic meanings into consideration.

References:

Refer to the below blogs to learn about the differences between Stemming and Lemmatization.

<https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/>

<https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221>

Tokenization

Tokenization is the process of splitting a given string into a sequence of sub-strings. There are two types of Tokenization. They are **Word Tokenizer** and **Sentence Tokenizer**.

The Word Tokenizer splits the given sentence into a sequence of words, on the basis of space whereas the Sentence Tokenizer splits the given sentence into a sequence of words/sentences on the basis of dot(.)

Example: "Hello Mr.Rajeev, How are you doing today?"

Word Tokenizer Output:

[“Hello”, “Mr.”, Rajeev”, “How”, “are”, “you”, “doing”, “today”]

Sentence Tokenizer Output:

[“Hello Mr”, “Rajeev How are you doing today?”]

28.6 Uni-gram, Bi-gram and n-grams

Let us assume we have two reviews 'r1' and 'r2'.

r₁: this pasta is very tasty and affordable

r₂: this pasta is not tasty and is affordable

After removing the underlined stopwords, the reviews become

r₁: pasta tasty affordable

r₂: pasta tasty affordable

Now both 'r₁' an 'r₂' have become exactly the same and the distance between their corresponding vectors 'v₁' and 'v₂' is zero, as they both are the same. But here, now we are forced to conclude that both the reviews are similar. But these two reviews, in their original form(ie., before removing the stopwords) are quite opposite. So we shouldn't go with false conclusions. In order to solve this type of problem, we need bi-grams, tri-grams, n-grams, etc.

Let us consider the reviews in their original form again.

r₁: this pasta is very tasty and affordable

r₂: this pasta is not tasty and is affordable

Uni-grams

We have a d-dimensional vector for all the words in the corpus. The presence of each word is indicated by non zero values in each dimension.

	this	pasta	is	very	tasty	and	affordable	not
r1	1	1	1	1	1	1	1	0
r2	1	1	1	0	1	1	1	1

Bi-grams

Here we create a vector for each review, with a pair of words as each dimension and these words will be the consecutive words of both the reviews.

	this	pasta	pasta	is	is	very	very	tasty	tasty	and	and	affordable	is	not	not	tasty	tasty	and	is	is	affordable
r1			1		1		1		1		1		1	0	0	0	0	0	0	0	
r2			1		1		0		0		1		0	1	1	1	1	1	1	1	

Tri-grams

Each dimension is obtained by taking 3 consecutive words at a time.

	this	pasta	is	very	tasty	and	affordable	pasta	is	not	is	tasty	and	is	affordable
r1			1		1		1		1	0	0	0	0	0	0
r2			1		0		0		0	1	1	1	1	1	1

28.7 TF-IDF (Term Frequency - Inverse Document Frequency)

Term Frequency (TF)

Let us assume we have the words ' w_1 ', ' w_2 ', ' w_3 ', ' w_4 ', ' w_5 ' and ' w_6 ' and there are ' N ' documents in the corpus.

Term Frequency(w_i, r_j) = (Number of times ' w_i ' occurs in ' r_j ')/(Total Number of Words in ' r_j ')

Let us assume we have 2 reviews ' r_1 ' and ' r_2 ' and the words in them be

r_1 : $w_1 w_2 w_3 w_2 w_5$

r_2 : $w_1 w_3 w_4 w_2 w_6 w_5$

So $TF(w_2, r_1) = \frac{2}{5}$

$TF(w_2, r_2) = \frac{1}{6}$

The term frequency of any word in general lies in between 0 and 1 (inclusive). So as this value lies in between 0 and 1, we can interpret it as probability. So $TF(w_i, r_j)$ can also be called as probability of occurrence of the word ' w_i ' in ' r_j '.

Inverse Document Frequency (IDF)

Let $D_c \rightarrow$ Data of corpus $\rightarrow \{r_1, r_2, \dots, r_n\}$

Inverse Document Frequency of a word is defined over the corpus, but not over a document.

IDF(w_i, D_c) = $\log_e(\text{Total Number of Documents}(N)/\text{Total Number of reviews containing } 'w_i')$

So $IDF(w_i, D_c) = \log_e(N/n_i)$

We know that $n_i \leq N$, so $N/n_i \geq 1$.

So it means $\log_e(N/n_i) \geq 0$

If ' n_i ' increases, ' N/n_i ' decreases and ultimately $\log_e(N/n_i)$ decreases.

$\log_e(N/n_i)$ is a monotonically decreasing function in ' n_i '. The more the word ' w_i ' across the reviews in the corpus, the lesser is the IDF score.

If ' w_i ' is more frequent, $IDF(w_i, D_c)$ will be low.

If ' w_i ' is a rare word, $IDF(w_i, D_c)$ will be more.

Vector Creation for each document using TF-IDF

In vector creation using TF-IDF, the magnitude of each dimension ' w_i ' is given by

Magnitude of ' w_i ' = $TF(w_i, r_i) * IDF(w_i, D_c)$

In a nutshell, we are giving more priority to the words which occur most frequently (whose TF value is high), and the words which occur rarely (whose IDF value is very low). By this, we can maximize the value of the product $TF * IDF$.

Drawbacks of TF-IDF

One of the drawbacks of TF-IDF is, it also doesn't take the semantic meaning into consideration. (ie., words like (cheap, affordable), (tasty, delicious), (valuable, precious) are considered as separate dimensions)

Difference between BOW and TF-IDF

In both BOW and TF-IDF, we convert each of the reviews into a d-dimensional vector where 'd' is the number of unique words in the total text across all the reviews.

The key difference between BOW and TF-IDF is that instead of using frequency counts as the values in the d-dimensional vector for each word as in BOW, we use the TF-IDF score for the words in TF-IDF vector representation.

28.8 Why do we use 'log' in the IDF?

The Inverse Document Frequency of a word ' w_i ' across the document corpus ' D_c ' is given as

$$\text{IDF}(w_i, D_c) = \log(N/n_i)$$

Where $N \rightarrow$ Total number of documents in the corpus

$n_i \rightarrow$ Total number of documents containing the word ' w_i '

In order to know why we use logarithm in the IDF, we shall go through Zipf's law first.

Zipf's Law

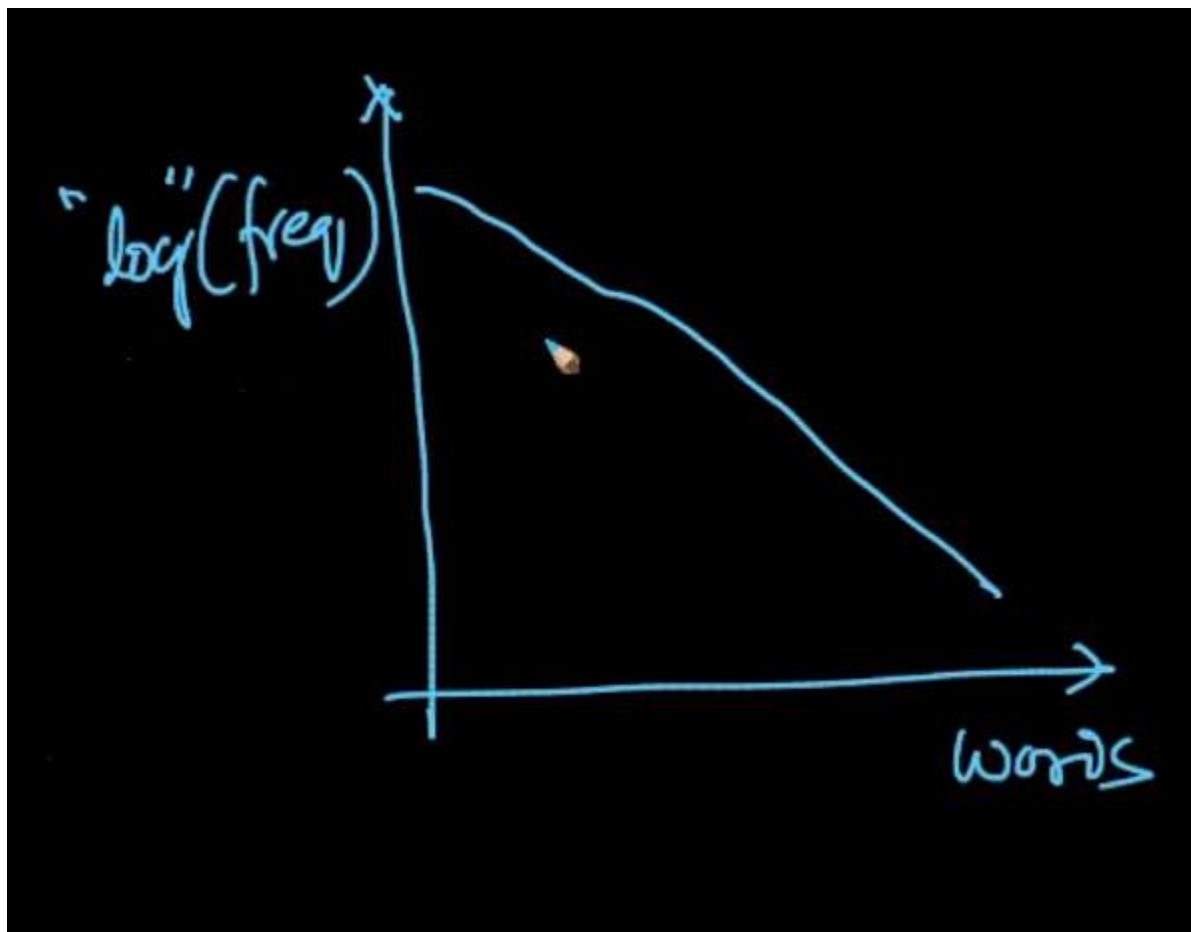
As mentioned in the video, starting from the timestamp 6:35, if we have all the words on the 'X' axis and the frequency of occurrence of each word on 'Y' axis, then if we plot a histogram, it looks like below



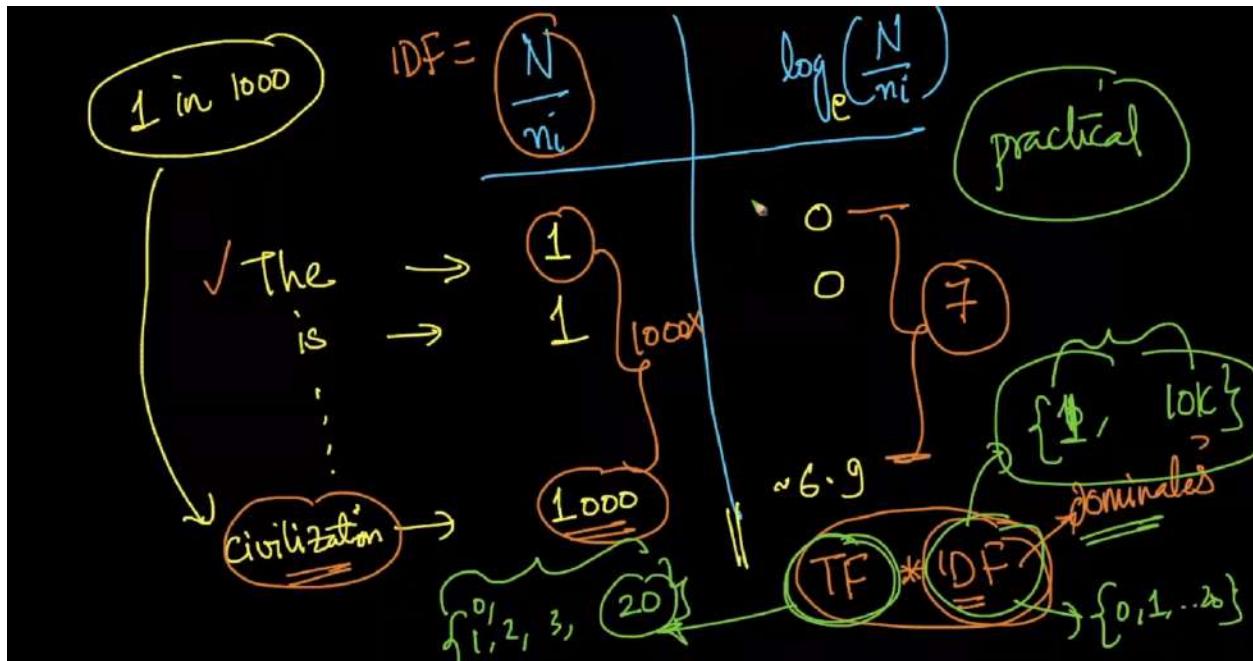
In the above plot, the most frequently occurring words are present towards the origin and the rare words are present away from the origin on the 'X' axis. The curve is in the decreasing order of the frequency. This is an example of Power Law.

If we have a random variable 'X' which follows Power Law, then we can convert it into a gaussian distribution by applying a box-cox transform. Also from the definition of Power Law, we also know that if random variables 'X' and 'Y' follow the power law, then the plot of $\log(X)$ vs $\log(Y)$ will be a straight line. Similarly, even if one feature(ie.,

frequency) follows power law while the other feature(words) is discrete, then if we apply logarithm to the frequency, the curve gets transformed into a line as shown below.



Let us now apply logarithm to the values of (N/n_i) and check how the transformed values look like, as discussed in the video starting from the timestamp 8:00



The range of (N/n_i) is 1 to 10000, whereas the range of $\log(N/n_i)$ is 1 to 7. So if we take (N/n_i) , the range is very high, the IDF will dominate TF and result in a huge value. Hence we choose $\log(N/n_i)$ to get reasonable values. We apply logarithm to reduce the scale.

28.9 Word2Vec

We have seen the BOW and TF-IDF techniques for converting a text into a vector. But these techniques do not take the semantic meanings into consideration, whereas Word2Vec is a state of the art technique used to convert the text into a vector, and also takes the semantic meaning into consideration.

So far in BOW/TF-IDF, we have seen a text is given as an input and the output is a sparse vector. But Word2Vec takes a word as an input and gives a d-dimensional vector as an output which is dense.

If the words ' w_1 ' and ' w_2 ' are semantically similar, then their vectors ' v_1 ' and ' v_2 ' are closer. This is the principle, Word2Vec tries to achieve. Word2Vec also tries to satisfy the relationships between the words.

For example, if we have the words ' w_1 ', ' w_2 ', ' w_3 ' and ' w_4 ' as

w_1 = "man"

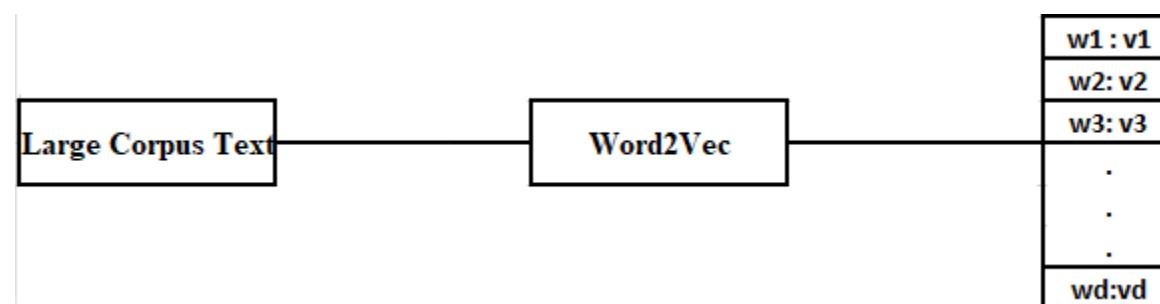
w_2 = "woman"

w_3 = "king"

w_4 = "queen"

Here the vector difference ($V_{\text{man}} - V_{\text{woman}}$) is parallel to ($V_{\text{king}} - V_{\text{queen}}$). This is the relationship it holds. Here it is a male-female relationship. Similarly, it holds country-capital, country-currency relationships, etc. It also holds verb-tense relationships. All these relationships are learnt by Word2Vec automatically from the raw text, without being explicitly programmed.

Top Level View of how Word2Vec works



Here a large corpus text is given as an input to Word2Vec and then Word2Vec creates a vector for each word. These vectors are high-dimensional. The more the dimensions we have in our vectors, the more rich the information is going to be. In order to get as many dimensions in the vectors, we need to give as much large corpus text as input.

In the core nutshell, for every word, the Word2Vec checks for the neighborhood of that word and if the neighborhood of this word is similar to the neighborhood of other words, then the vector of this word and other words is similar.

If we have two words ' w_i ' and ' w_j ', then if the neighborhood of the word ' w_i ' is the same as the neighborhood of ' w_j ', then the vectors of ' w_i ' and ' w_j ' are similar.

So far in BOW and TF-IDF we have converted documents into vectors. In Word2Vec, we are converting each word of the corpus into a vector.

Note:

Word2Vec could not give the correct results for stemmed words. For example, for words like 'tasti', we do not get appropriate results as the stemmed words are not present in the text directly. So we should not perform stemming on the data if we want to go for Word2Vec.

28.10 Avg Word2Vec, TF-IDF Weighted Word2Vec

Average Word2Vec

Let us assume we have a document/review ' r_1 '. Let its corresponding average vector form be denoted as ' v_1 '. Let the total number of words in ' r_1 ' be ' n_1 '.

$r_1: w_1 \ w_2 \ w_1 \ w_3 \ w_4 \ w_5$

So now the average word2vec for ' r_1 ' is

$$v_1 = (1/n_1)[w2v(w_1) + w2v(w_2) + w2v(w_1) + w2v(w_3) + w2v(w_4) + w2v(w_5)]$$

$w2v(w_i)$ represents the word vector of the word ' w_i '. This is called the Average Word2Vec representation of ' r_1 '. Average Word2Vec fairly works well in practice, but is not perfect all the time. It still works well enough.

Average Word2Vec is a simple way to leverage Word2Vector to build sentence vectors.

TF-IDF Weighted Word2Vec

Let us assume we have 7 words (say ' w_1 ', ' w_2 ', ' w_3 ', ' w_4 ', ' w_5 ', ' w_6 ', ' w_7 ') and the review/document ' r_1 '.

$r_1: w_1 \ w_2 \ w_1 \ w_3 \ w_4 \ w_5$

The TF-IDF representation of the ' r_1 ' vector is given as below

w1	w2	w3	w4	w5	w6	w7
t1	t2	t3	t4	t5	t6=0	t7=0

Here $t_i \rightarrow \text{TF-IDF}(w_i, r_1)$

Now, let ' v_1 ' be the TF-IDF Weighted Word2Vec representation of the vector ' r_1 ', then

$$v_1 = (t_1 * w2v(w_1) + t_2 * w2v(w_2) + t_3 * w2v(w_3) + t_4 * w2v(w_4) + t_5 * w2v(w_5)) / (t_1 + t_2 + t_3 + t_4 + t_5)$$

It can simply be written as

$$\text{TFIDF Weighted Word2Vec } (r_1) = \sum_{i=1}^n (t_i * w2v(w_i)) / (\sum_{i=1}^n t_i)$$

Note - Special Case:

If $t_i=1$ (ie., $t_1 = t_2 = t_3 = \dots = 1$), then the TF-IDF Weighted Word2Vec is the Average Word2Vec.

Average Word2Vec and TF-IDF Weighted Word2Vec are two simple weighting strategies to convert sentences into vectors. They both serve the same purpose. TF-IDF Weighted Word2Vec weights each word differently as compared to Average Word2Vector. In practice, we try both the options and choose the one that performs better at our task.

29.1 How Classification works?

Let us consider the Amazon Fine Food reviews dataset. For each review, we have a text associated with it, and this text is converted into a vector form through one of the vectorization techniques like BOW, TF-IDF, Avg Word2Vec, TF-IDF Weighted Average Word2Vec, etc. Now for each review, we have a vector and also the data that tells us whether the review is positive or negative.

The classification task for this problem is, for a given review, we have to predict if it is positive or negative.

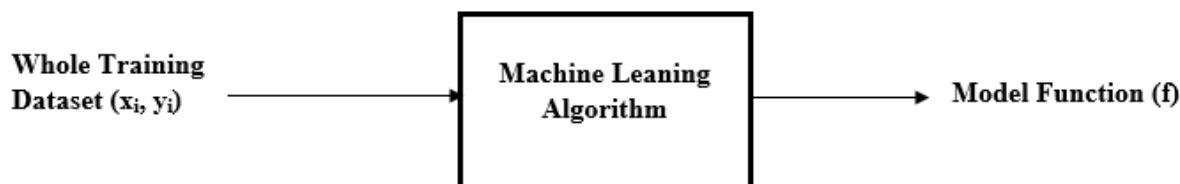
Let us assume each review in our dataset is denoted as ' x_i '. Classification here is given a review ' x_i ', we have to find a function ' f ' such that if we apply that function on ' x ', we get the output ' y ' either as 'Positive' or 'Negative'. This is the central concept of classification.

$$y_i = f(x_i)$$

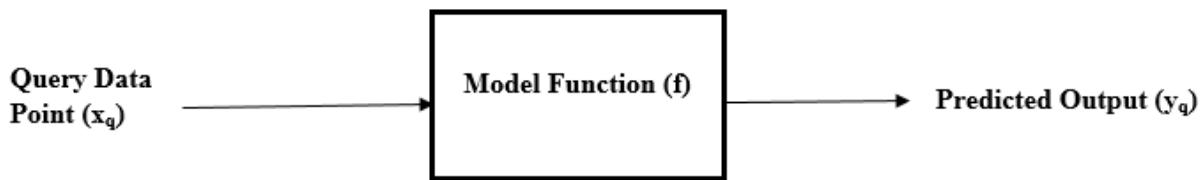
where $x_i \rightarrow$ query review (in vector form)

$y_i \rightarrow$ predicted class label

Training Stage



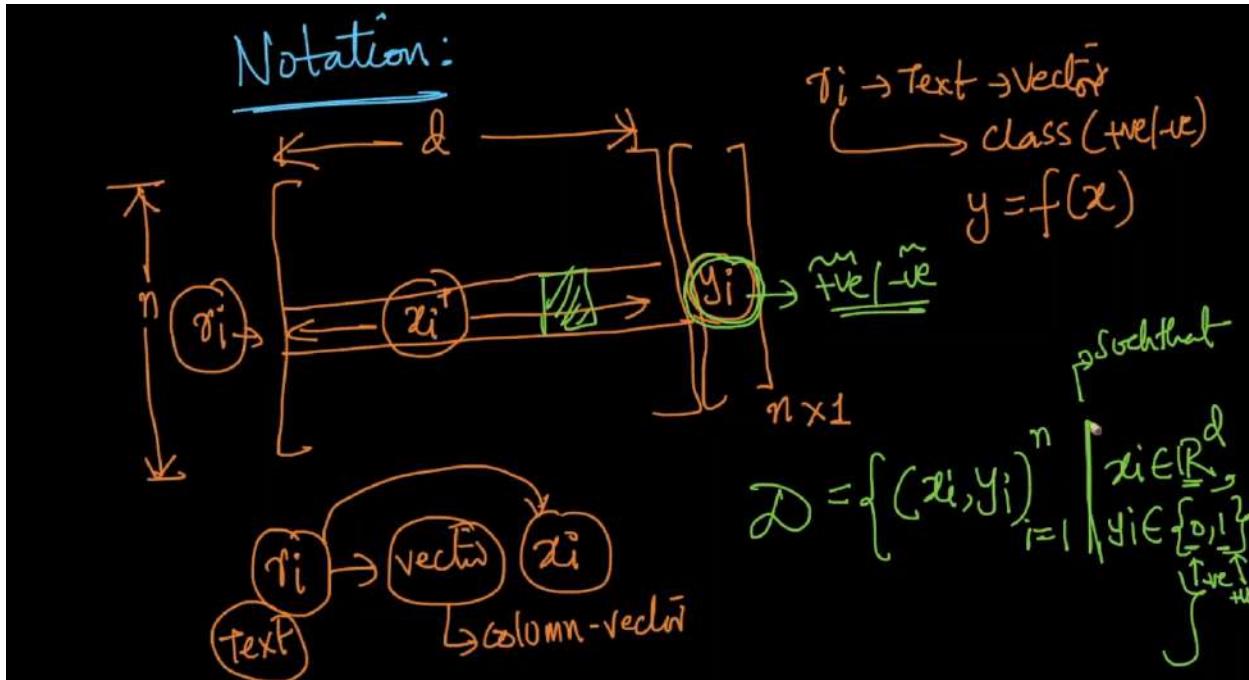
Testing/Validation Stage



The algorithm gets trained using the training data and computes the function ' f '. After computing the function ' f ', in the validation stage, this function is used to predict the output for the test data.

29.2 Data Matrix Notation

We have the reviews in Amazon Fine Food Reviews Dataset and let each review ' r_i ' be represented in a vector form ' x_i ' and the corresponding output class may be denoted by ' y_i '. Then each data point is represented in data matrix format as shown below.



Here the rows represent the data points and the columns represent the dimensions/features. Let us assume we have 'n' data points, and each data point has components in 'd' dimensions. Then the dataset is represented mathematically in the form of a set as $D = \{(x_i, y_i)_{i=1}^n | x_i \in \mathbb{R}^d, y_i \in \{0, 1\}\}$

Here we are converting the classes from 'Negative' and 'Positive' format to 0-1 format, because linear algebra couldn't understand the terms 'Positive' and 'Negative'.

Note: As there are 'd' dimensions in every data point ' x_i ', we have used the notation $x_i \in \mathbb{R}^d$. As ' y_i ' takes only two values (ie., 0 and 1), we have used the notation $y_i \in \{0, 1\}$. As there are only 2 classes in the dataset, we call this problem a **binary classification** problem.

Note: If we consider the MNIST dataset, there are 60000 data points, each data point is a 784-dimensional vector and the class labels are the numbers 0 to 9, then the MNIST dataset is represented mathematically in the form of a set as

$$D = \{(x_i, y_i)_{i=1}^{60000} | x_i \in \mathbb{R}^{784}, y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}\}$$

As there are 10 classes in the MNIST dataset, we call this problem a **10-class classification** (or) a **multi-class classification** problem.

29.3 Classification vs Regression

Classification - Definition

Classification is the problem of identifying to which of a set of categories a new observation belongs to, on the basis of a training set of data containing observations whose category is already known.

Classification deals with predicting a qualitative (or) categorical response.

Below are the examples of Classification problems that were discussed starting from the timestamp 0.03.

$$\mathcal{D} = \left\{ (\underline{x}_i, y_i) \right\}_{i=1}^n \quad | \quad \underline{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}$$

$y_i \in \{0, 1\}$

↓
ve ↓
+ve

2 classes

Amazon Food reviews

2 class - classification / binary

MNIST: $y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \rightarrow$ 10-class / Multi-class classification

In the dataset of Amazon Fine Food Reviews, the class labels are 0 and 1. If any query point is given, its label would be either 0 or 1. As it has only 2 classes, such a classification problem is called a **binary classification** problem.

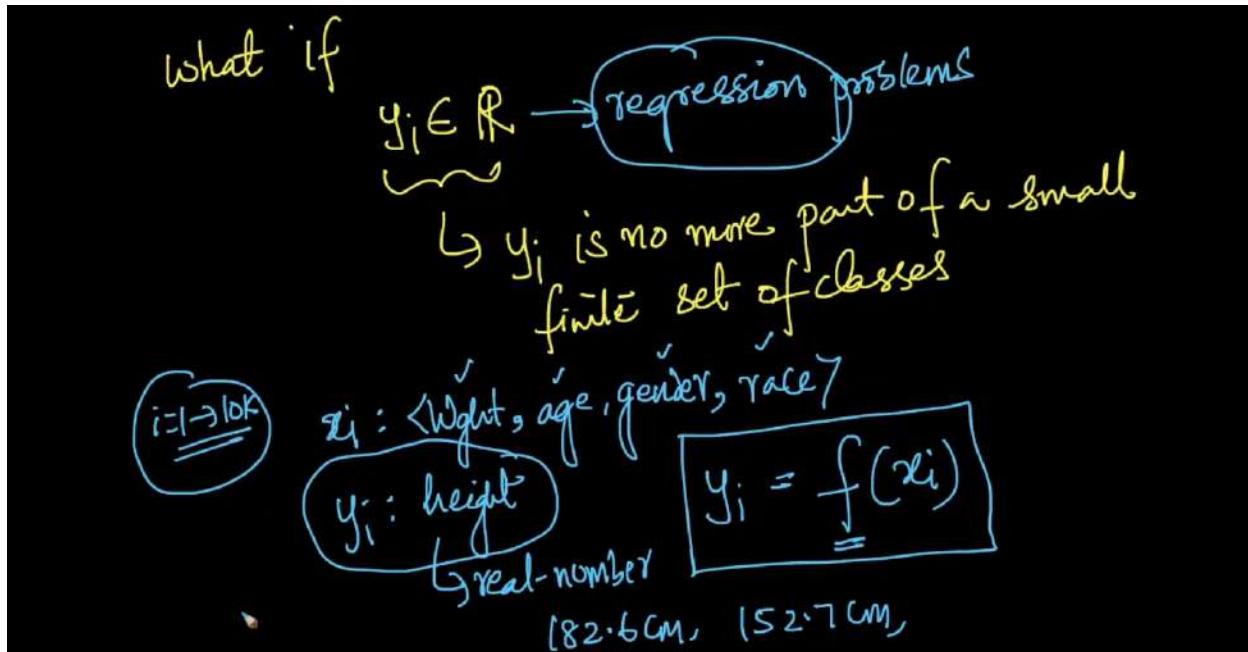
Similarly in the MNIST dataset, the class labels are the numbers from 0 to 9. If any query point is given, its label would be any one value from 0 to 9. As it has only 10 classes, such a classification problem is called a **10-class classification** problem (or) **multi-class classification** problem.

In both the examples mentioned above, the output class label comes from a finite set of values. Hence we call the problem of predicting such an output, a classification problem.

Regression - Definition

Regression is the problem of predicting a value for a new observation, on the basis of a functional relationship between two variables and on the basis of the training set of data containing observations whose value is already known.

Regression deals with predicting a quantitative (or) numerical response.



In the above example, we are predicting the height of an individual. The height value doesn't come from a finite set of values, but it comes from an infinite set of numerical values. As the output comes from an infinite set of numerical values, we call this problem a Regression problem.

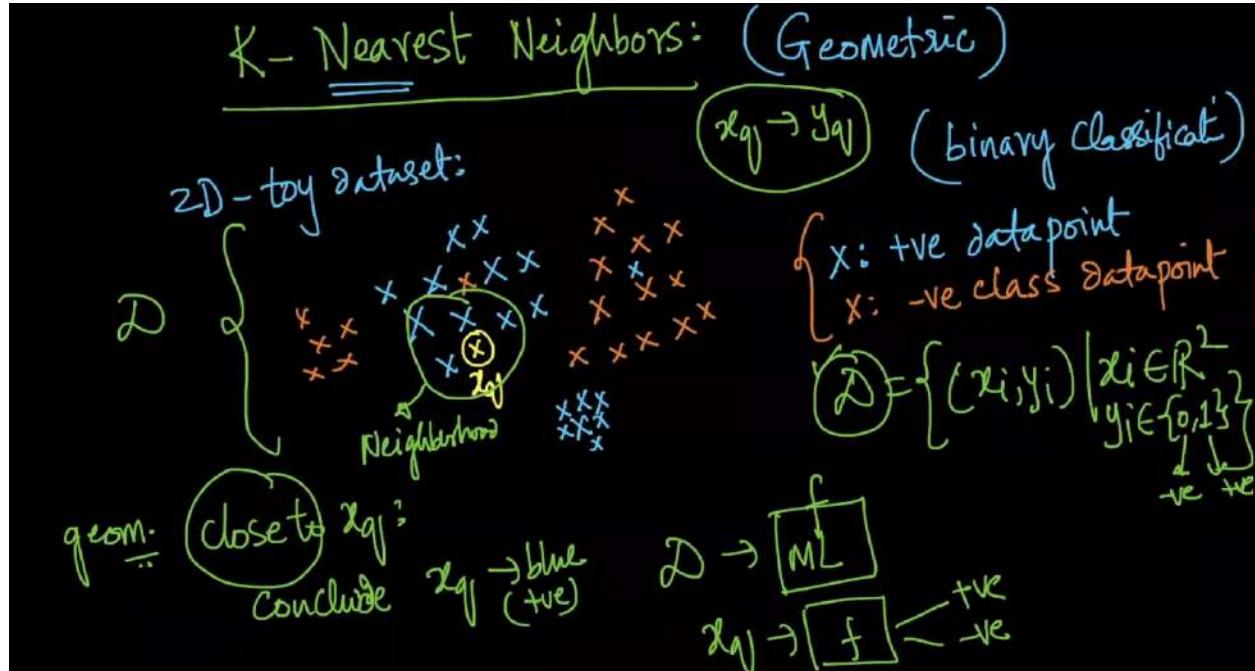
Note: As the output of a regression problem comes from an infinite set of numerical values, we denote it as $y_i \in \mathbb{R}$.

29.4 K-Nearest Neighbors Geometric Intuition with a toy example

Let us assume we are working on a binary classification problem and each observation belongs to either positive class or negative class. Let the dataset be in a 2-dimensional form. Here the dataset is represented as

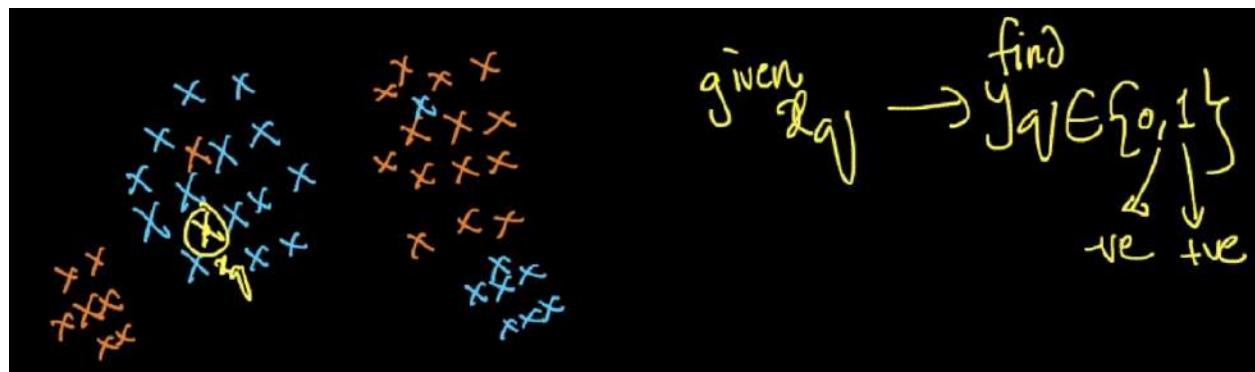
$$\mathcal{D} = \{(x_i, y_i)_{i=1}^n \mid x_i \in \mathbb{R}^2, y_i \in \{0, 1\}\}$$

The main purpose of classification is given a point ' x_q ', we have to predict the value of ' y_q '. Below is an example that was discussed starting from the timestamp 0:18.



Procedure of KNN

Given a query point ' x_q ', we have to predict the class label ' y_q '.



- 1) Compute the distance of the point ' x_q ' to all the points in the training dataset.
- 2) Sort all the distances in the ascending order, and then select the top 'K' nearest points to ' x_q '.

3) Let these 'K' points be $\{x_1, x_2, x_3, \dots, x_k\}$ and their corresponding outputs may be $\{y_1, y_2, y_3, \dots, y_k\}$.

Here for the point ' x_q ', the class to which majority of the class labels among $\{y_1, y_2, y_3, \dots, y_k\}$ belong to, will be predicted as the output for ' x_q '.

a) For example, if $K=3$ and let $\{y_1, y_2, y_3\} = \{\text{+ve}, \text{+ve}, \text{-ve}\}$, then ' x_q ' will be assigned to the '+ve' class, as the '+ve' class is in majority.

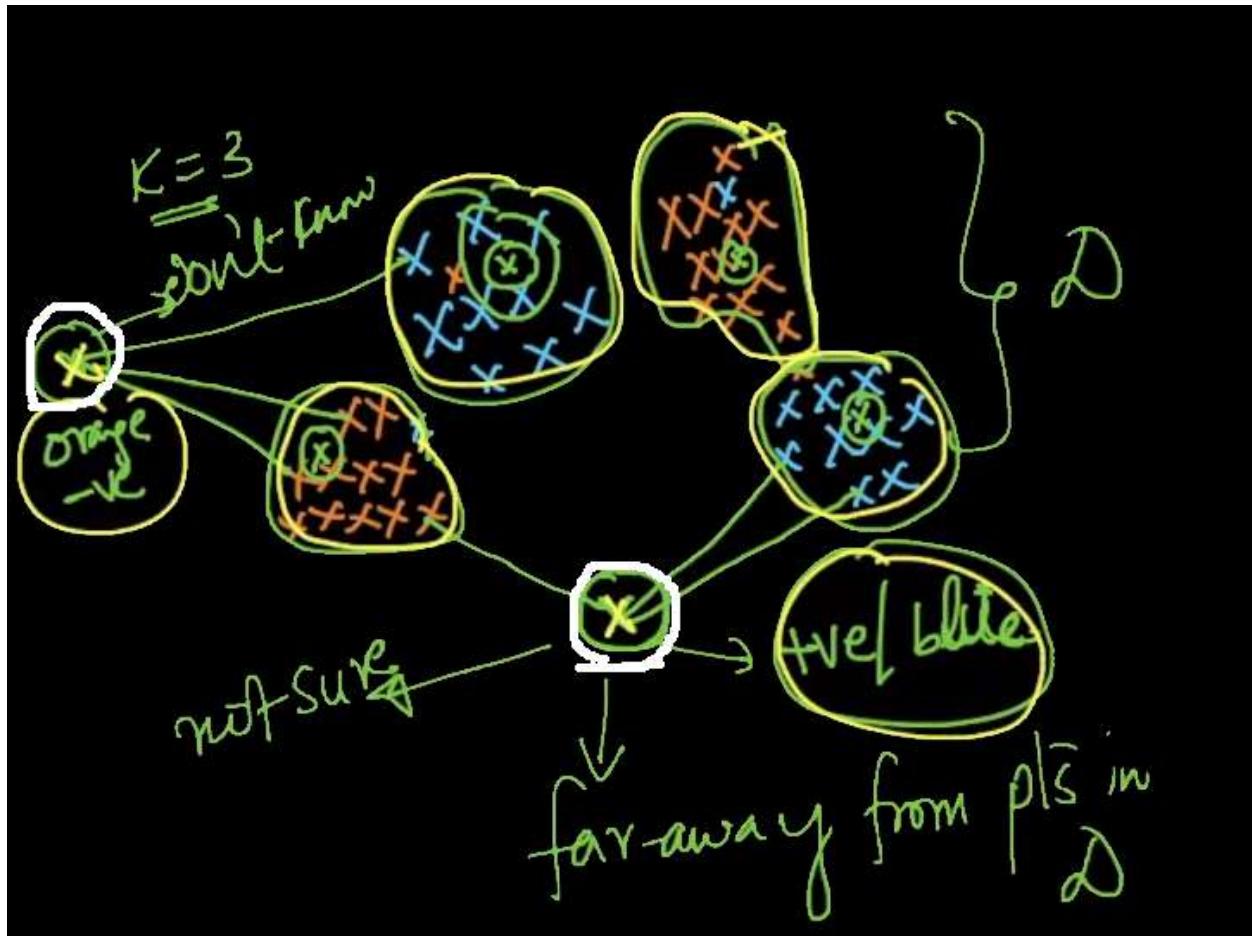
b) For example, if $K=5$ and let $\{y_1, y_2, y_3, y_4, y_5\} = \{\text{+ve}, \text{-ve}, \text{-ve}, \text{-ve}, \text{+ve}\}$, then ' x_q ' will be assigned to the '-ve' class, as the '-ve' class is in majority.

Here if 'K' is an even number, and if both the '+ve' and '-ve' class points in the neighborhood are equal in number, then it becomes difficult for the model to decide which class the datapoint ' x_q ' should be assigned to. So it is always better to avoid even values of 'K' in K-NN.

In cases where the 'K' value is even and the number of data points in '+ve' and '-ve' classes equal, then we should better increase the 'K' value, and repeat the same operation. Pick the class label with the majority of the points and assign it to ' x_q '.

29.5 Failure Cases of K-NN

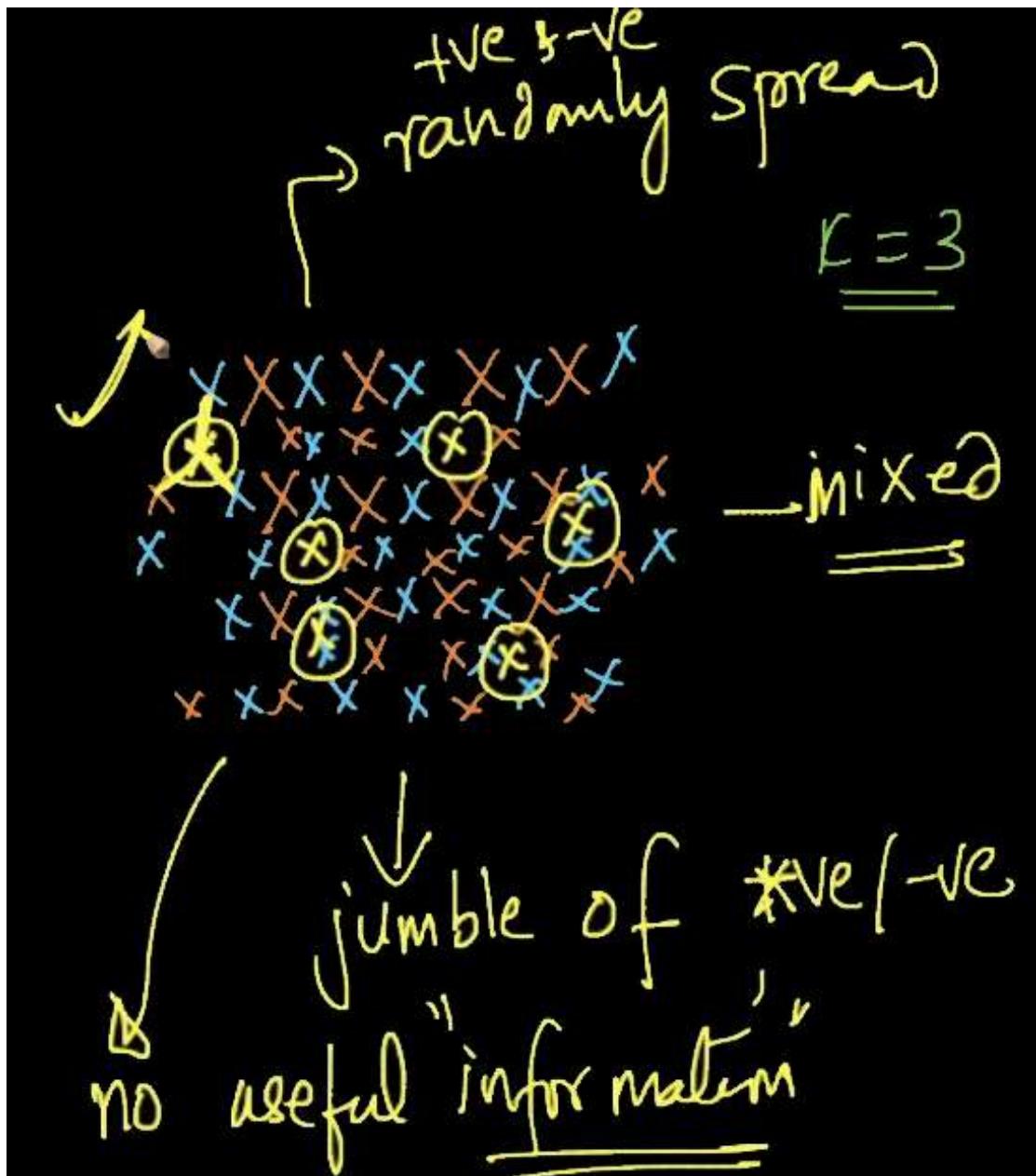
Case 1



If a point ' x_q ' is surrounded by (or) nearby more number of '+ve' points cluster or '-ve' points cluster, then we can easily classify this point into one of them, depending on the majority vote of the class labels of the points in the neighborhood.

Now what if our query points are far away from all the points in the dataset like the two given query points circled in white. When we have a query point that is far away from the rest of the points in the dataset, then we are not sure which class this point has to be classified. Though the majority of the points which are nearer belong to a particular class, we still couldn't say anything about the class accurately, because as this particular point is far away from the regular points, it might exhibit a different behaviour(ie., it doesn't belong to the pattern of the points)

Case 2



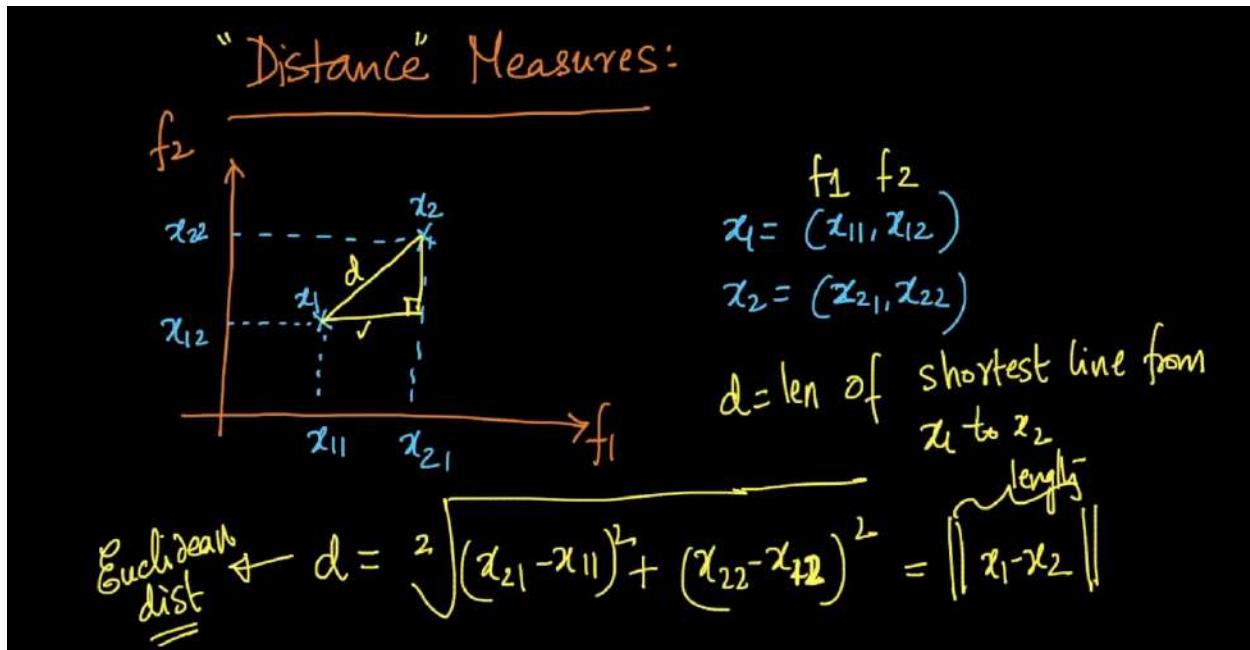
If the '+ve' and the '-ve' points are randomly spread, here we do not have exact and useful information. Here even if we predict the class label for the point ' x_q ' as '+ve' (or) '-ve', it will result in an error. In this kind of situation where there is no useful information, KNN fails.

The above 2 mentioned cases are the simplest cases, where KNN fails.

Note: When the data points of different classes are randomly jumbled up, no model can work well. All models will fail. If a real world dataset shows this sort of behavior, it is better to change the feature set, as the current features are useless for classification.

29.6 Distance Measures: Euclidean(L2), Manhattan(L1), Minkowski, Hamming

Euclidean Distance (L2)



Let us assume, we have a 2-dimensional space and we have two points ' x_1 ' and ' x_2 '.

$$x_1 = (x_{11}, x_{12})$$

$$x_2 = (x_{21}, x_{22})$$

$d \rightarrow$ Length of the shortest line from ' x_1 ' to ' x_2 '

$$d = \sqrt{(x_{21} - x_{11})^2 + (x_{22} - x_{12})^2} = \|x_1 - x_2\|$$

Here ' d ' \rightarrow euclidean distance between the points ' x_1 ' and ' x_2 '.

Let us assume we are working on a d -dimensional space.

$$x_1 \in \mathbb{R}^d, x_2 \in \mathbb{R}^d$$

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^d (x_{1i} - x_{2i})^2}$$

$$\|x_1 - x_2\|_2 \rightarrow L2 \text{ norm of the vector } (x_1 - x_2)$$

$$\|x_i\| \rightarrow \text{Distance of the point } 'x_i' \text{ from origin} = \sqrt{\sum_{i=1}^d x_i^2}$$

Manhattan Distance (L1)

✓ (Manhattan) dist: $\sum_{i=1}^d |\chi_{1i} - \chi_{2i}|$
 L₁-norm of vector $(\chi_1 - \chi_2)$ $\|(\chi_1 - \chi_2)\|_1$
 $\|\chi_1\|_1 = \sum_{i=1}^d |\chi_{1i}|$

The Manhattan distance (d) between $x_1(x_{11}, x_{12})$ and $x_2(x_{21}, x_{22})$ is given as

$$\text{Manhattan Distance} = \sum_{i=1}^d |\chi_{1i} - \chi_{2i}|$$

For a d-dimensional space, it is given as

$$\text{Manhattan Distance} = \sum_{i=1}^d |\chi_{1i} - \chi_{2i}|$$

$\|\chi_1 - \chi_2\| \rightarrow$ L1 Norm of the vector $(\chi_1 - \chi_2)$

$$\|\chi_1\|_1 = \sum_{i=1}^d |\chi_{1i}|$$

Minkowski Distance (Lp)

L_p -norms \nrightarrow Minkowski dist

$$\|(\chi_1 - \chi_2)\|_p = \left(\sum_{i=1}^d |\chi_{1i} - \chi_{2i}|^p \right)^{1/p}$$

$\left\{ \begin{array}{l} (-3)^2 = 9 \\ (1-3)^2 = 9 \end{array} \right.$

$= L_p$ -norm of $(\chi_1 - \chi_2)$

$p=2 \rightarrow$ Minkowski dist \rightarrow Eucl. dist
 $p=1 \rightarrow$ " \rightarrow Manhattan dist

The Minkowski distance is the ' L_p ' norm of the vector $(x_1 - x_2)$. The Minkowski distance in a d-dimensional space is given as

$$\|x_1 - x_2\|_p = \left(\sum_{i=1}^d |x_{1i} - x_{2i}|^p \right)^{1/p} \quad (\text{Here } p > 0 \text{ always})$$

If $p = 1$, Minkowski Distance \rightarrow Manhattan Distance

If $p = 2$, Minkowski Distance \rightarrow Euclidean Distance

Hamming Distance

✓ Hamming dist (boolean Vectr)

$x_1, x_2 \rightarrow$ boolean Vectr \rightarrow Binary Bow

$x_1 = [0, 1, 0, 1, 0, 1, 0, 1, \dots]$

$x_2 = [1, 0, 1, 0, 1, 0, 1, 0, \dots]$

Hamming dist $(x_1, x_2) = \# \text{ locations where binary vectors differ}$

↳ 3

Hamming Distance between the two given vectors is the number of dimensions/features in which the two vectors differ. It is recommended to use Hamming Distance only on Binary Vectors. One best example is to apply Hamming Distance on the Binary BOW data. You also can apply it on Count Based BOW, but still you'll get the same result, as that obtained on the BOW data.

strings:

$x_1 = a \underset{\text{P}}{l} c \underset{\text{A}}{d} e f g h i k \leftarrow \text{Gene code / Seq}$

$x_2 = a \underset{\text{C}}{l} b \underset{\text{A}}{d} e g j h i k$

$x_1 = A G T C T C A G \quad AGTC$

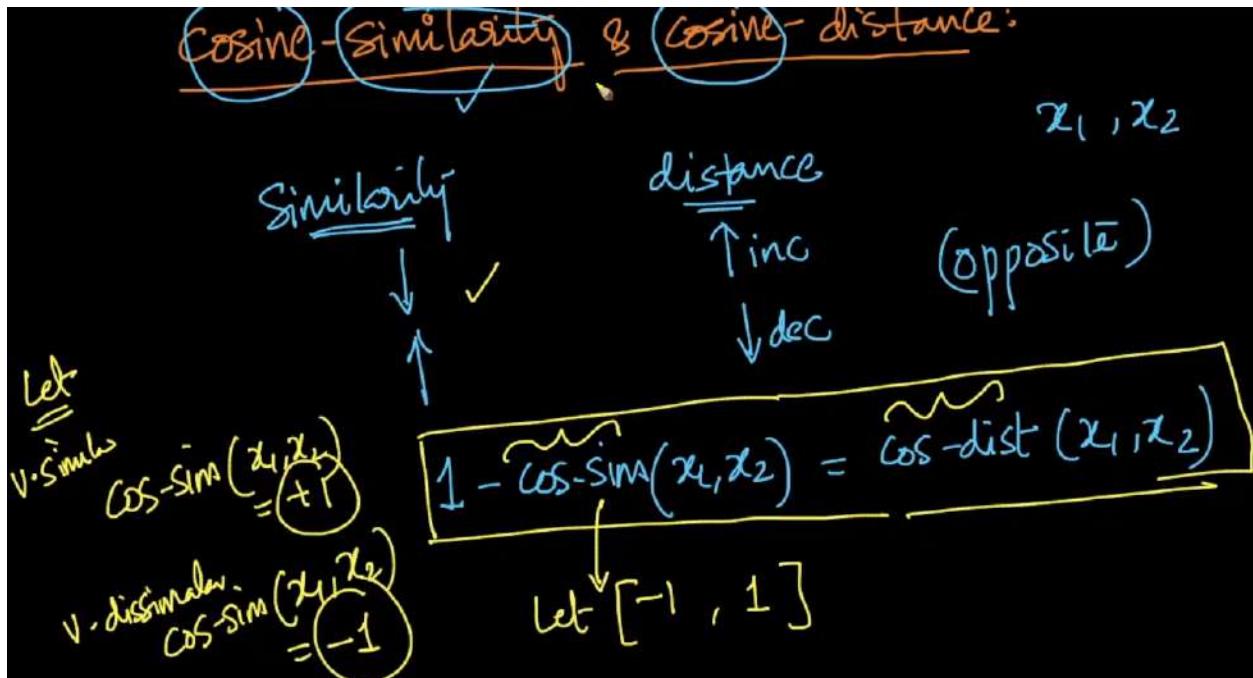
$x_2 = A G T A T C T C A$

hamming dist $(x_1, x_2) = 4$

Hamming Distance between the two given strings is the total number of differing components in those strings. In the above example, we see there are 4 characters differing ($2^{\text{nd}}, 3^{\text{rd}}, 7^{\text{th}}, 8^{\text{th}}$ locations). So the hamming distance is 4.

Note: Hamming Distance is no way related to the Minkowski Distance.

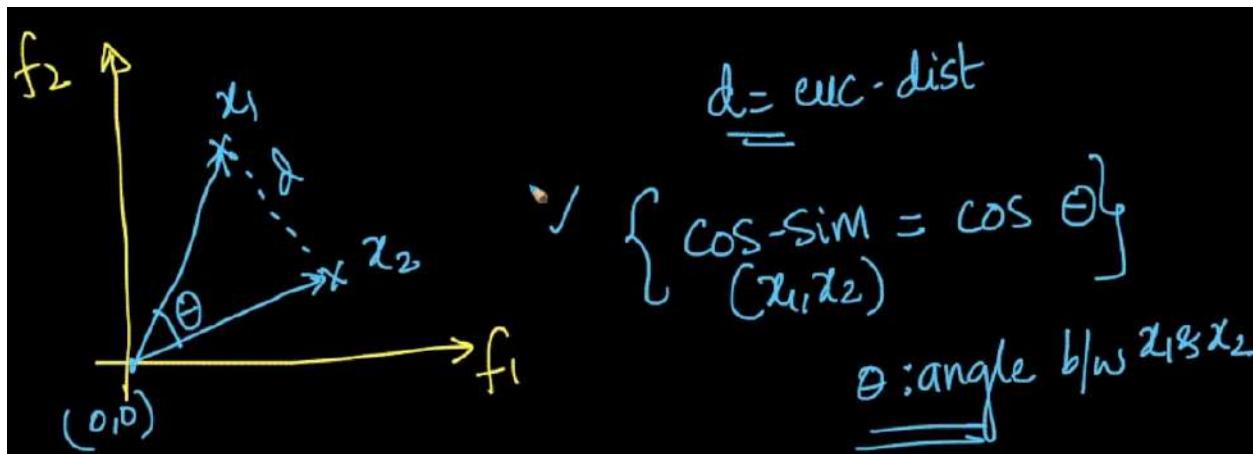
29.7 Cosine Distance & Cosine Similarity



Let us assume there are two vectors ' x_1 ' and ' x_2 '. As the distance between the two points increases, the similarity between them decreases. Similarly if the similarity between two points increases, then the distance between them decreases.

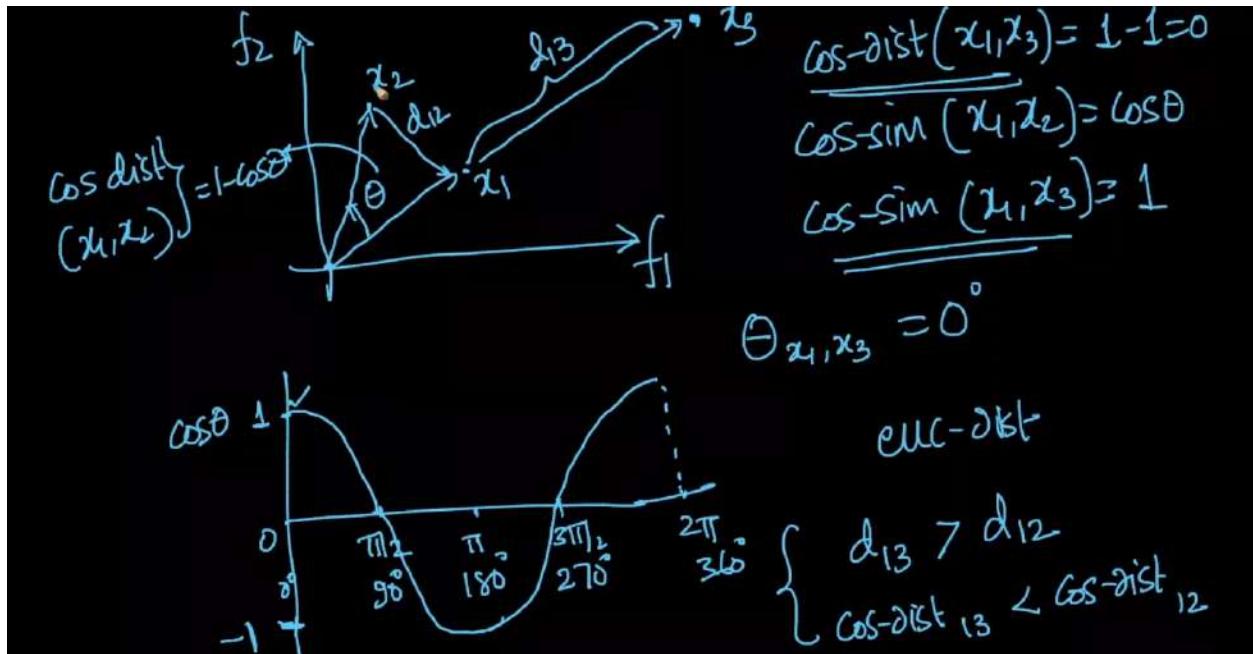
The relationship between cosine similarity and cosine distance is given as

$$1 - \cosine\text{-similarity}(x_1, x_2) = \cosine\text{-distance}(x_1, x_2)$$



The cosine similarity between ' x_1 ' and ' x_2 ' is given by the cosine of the angle between the vectors ' x_1 ' and ' x_2 '.

cosine_similarity(x_1, x_2) = $\cos\theta$ where ' θ ' is the angle between the vectors ' x_1 ' and ' x_2 '. Cosine Similarity is the measure of the similarity between two given non zero vectors.



Let us consider the 3 vectors ' x_1 ', ' x_2 ' and ' x_3 '. According to the figure above,

$$\text{cosine_similarity}(x_1, x_2) = \cos\theta$$

$$\text{cosine_similarity}(x_1, x_3) = \cos(0) = 1$$

$$\text{cosine_distance}(x_1, x_3) = 1 - \text{cosine_similarity}(x_1, x_3) = 1 - 1 = 0$$

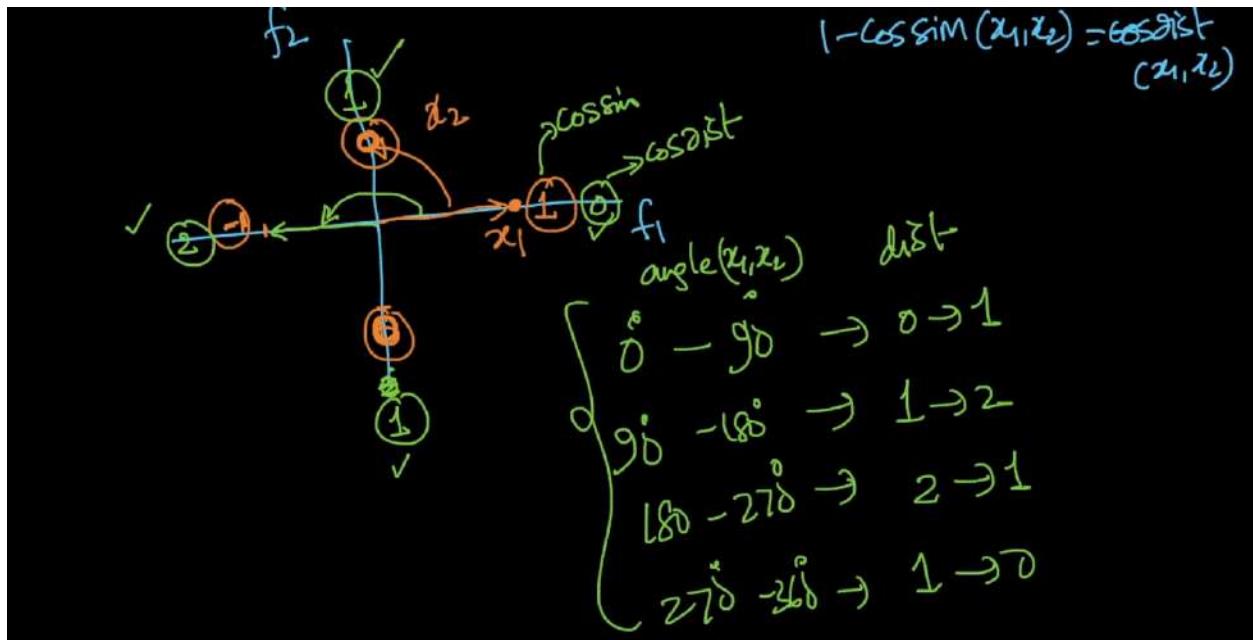
In case, if the angle between the vectors ' x_1 ' and ' x_2 ' is known, then

$$\text{cosine_similarity}(x_1, x_2) = \cos\theta$$

In case, if the angle between the vectors ' x_1 ' and ' x_2 ' is unknown, then

$$\text{cosine_similarity}(x_1, x_2) = \frac{x_1 \cdot x_2}{\|x_1\|_2 \cdot \|x_2\|_2}$$

If the vectors ' x_1 ' and ' x_2 ' are unit vectors, then $\|x_1\|_2 = \|x_2\|_2 = 1$



If the angle between the two given vectors ' x_1 ' and ' x_2 ' lies in between 0^0 and 90^0 (or) 270^0 and 360^0 (ie., $\theta \in [0, \pi/2]$ or $\theta \in [3\pi/2, 2\pi]$), then the cosine similarity lies in between 0 and 1. The cosine distance also lies in between 0 and 1.

If the angle between the two given vectors ' x_1 ' and ' x_2 ' lies in between 90^0 and 270^0 ((ie., $\theta \in [\pi/2, 3\pi/2]$)), then the cosine similarity lies in between -1 and 0. The cosine distance lies in between 1 and 2.

Note: Cosine Distance measures the angular difference between the vectors ' x_1 ' and ' x_2 '. Cosine Similarity is used as a metric for measuring distance when the magnitude of the vectors does not matter.

Relationship between Euclidean Distance and Cosine Distance

If ' x_1 ' and ' x_2 ' are the two given vectors, then

$$\|x_1 - x_2\|^2 = (x_1 - x_2)^T (x_1 - x_2) = \|x_1\|^2 + \|x_2\|^2 - 2x_1^T x_2 \quad \dots \dots \dots (1)$$

$x_1^T x_2$ is nothing but the dot product of the two vectors ' x_1 ' and ' x_2 '.

Formula: If we have two vectors 'A' and 'B', then

$A^T B = A \cdot B = \|A\| * \|B\| * \cos(\theta)$ where ' θ ' is the angle between the vectors 'A' and 'B'.

$$\text{So } x_1^T x_2 = x_1 \cdot x_2 = \|x_1\| * \|x_2\| * \cos(\theta) \quad \dots \dots \dots (2)$$

In the above equation, ' θ ' is the angle between the vectors ' x_1 ' and ' x_2 '.

After substituting (2) in (1), we get

$$\|x_1 - x_2\|^2 = \|x_1\|^2 + \|x_2\|^2 - (2 * \|x_1\| * \|x_2\| * \cos(\theta)) \quad \dots \dots \dots (3)$$

Let us assume the given two vectors ' x_1 ' and ' x_2 ' are the unit vectors, then

$$\|x_1\| = \|x_2\| = 1 \quad \dots \dots \dots (4)$$

After substituting (4) in the RHS of (3), we get

$$\|x_1 - x_2\|^2 = 1 + 1 - (2 * \cos(\theta))$$

$$\|x_1 - x_2\|^2 = 2 - (2 * \cos(\theta)) = 2(1 - \cos(\theta)) \quad \dots \dots \dots (5)$$

Here $\|x_1 - x_2\|^2 \rightarrow$ Euclidean Distance between the vectors ' x_1 ' and ' x_2 '.

$\cos(\theta) \rightarrow$ Cosine Similarity between the vectors ' x_1 ' and ' x_2 '.

$1 - \cos(\theta) \rightarrow$ Cosine Distance between the vectors ' x_1 ' and ' x_2 '.

So now the equation (5) becomes

$$[\text{euclidean_distance}(x_1, x_2)]^2 = 2 * \text{cosine_distance}(x_1, x_2)$$

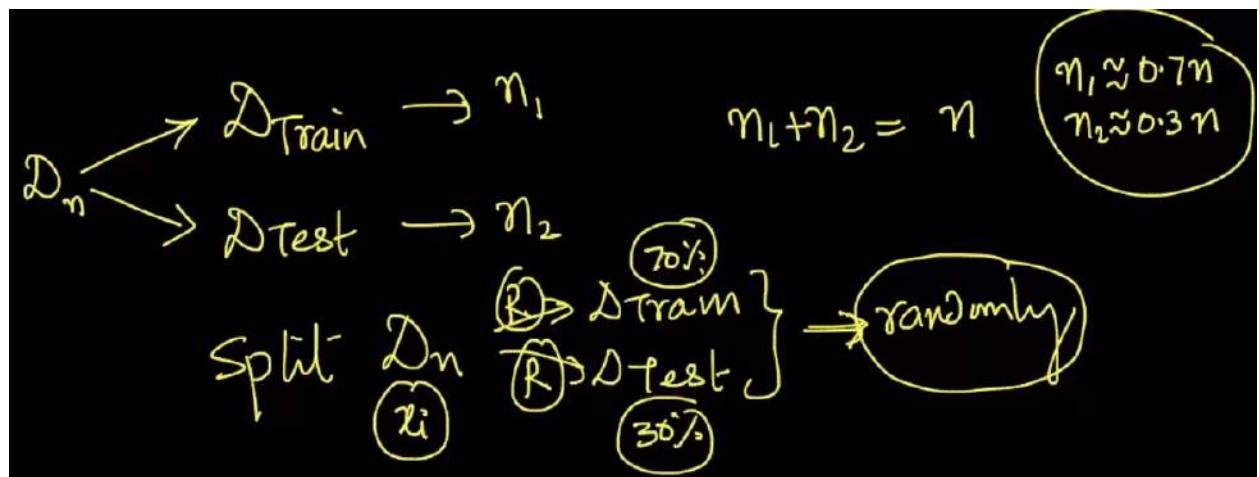
The above derived relationship is valid and applicable only if the two given vectors ' x_1 ' and ' x_2 ' are the unit vectors. (ie., $\|x_1\|_2 = \|x_2\|_2 = 1$)

29.8 How to measure the effectiveness of K-NN?

Let us consider the Amazon Fine Food Reviews Dataset which has got 364K reviews(after deduplication). For a given query point ' x_q ', we have to predict the class label ' y_q '. Each data point is represented in the form of a numerical vector, and each data point has its own class label.

Procedure to measure the effectiveness of K-NN

- 1) Let us assume we are given a dataset $\{D_n\}$ and our inputs are $\{x_i\}_{i=1}^n$ and the outputs are $\{y_i\}_{i=1}^n$
- 2) Divide the dataset $\{D_n\}$ into the training set $\{D_{Train}\}$ and the test set $\{D_{Test}\}$. Let ' n_1 ' be the number of points in $\{D_{Train}\}$ and ' n_2 ' be the number of points in $\{D_{Test}\}$. ($n_1 + n_2 = n$)



- 3) Now we have to fit the KNN model on ' D_{Train} ', so that the entire ' D_{Train} ' gets stored. Then for each point ' x_q ' in ' D_{Test} ', we have to make predictions using the same KNN model and predict the value of y_q .
- 4) Let us initialize a variable 'count = 0' and for every data point ' x_q ' in ' D_{Test} ', if $y_q == y_q$, then increment the 'count' value by 1.
- 5) Finally we have to compute the accuracy using the formula
Accuracy = count/(number of data points in ' D_{Test} ') = count/n₂
Accuracy value typically lies in between 0 and 1.

$\text{cnt} = 0$;
 for each pt in D_{test} : $x_1 \rightarrow y_1$
 $x_q = pt$
 use D_{Train} & K-NN to determine y_q
 if $y_q = y_{pt}$
 $\text{cnt} += 1$
 end
 $\text{cnt} = \# \text{ pts for which } D_{\text{Train}} + \text{KNN}$
 gave a correct class label

$$\text{Accuracy} = \frac{\text{cnt}}{n_2} \rightarrow \frac{\# \text{ pts for which } D_{\text{Train}} + \text{KNN}}{\# \text{ pts in } D_{\text{test}}} \text{ gave a correct class label}$$

$$0 \leq \text{Acc} \leq 1$$

$$\text{Acc} = 0.91 \Rightarrow 91\% \text{ of times}$$

$$x_1 \rightarrow y_1$$

D_{test}

Note: If accuracy = 0.92, it means in 92% of the cases, using the fit on ' D_{Train} ', the model predicts the output labels accurately.

29.9 Test/Evaluation Time and Space Complexity

For a data point ' x_q ' in KNN, in order to classify the label for the data points, we also need to take time and space complexities into consideration.

Here the data points are represented as ' x_q ' each and we have to predict their corresponding class label ' y_q '.

Inputs in KNN Algorithm: $D_{Train}, K, x_q \in \mathbb{R}^d$

Output: y_q

Pseudo Algorithm

KNN_points = []

for each x_i in D_{Train} :

Compute the distance between each x_i and ' x_q '. (Takes $O(d)$ as we have to compute the difference for all the ' d ' dimensions and square it.

Keep the smallest ' K ' distances and store them in KNN_points. It takes $O(K)$

Let us assume ' K ' is small. So let's ignore ' K ' in the Time Complexity. Then after this,

count_positive = 0, count_negative = 0

for each x_i in KNN_points:

if y_i is +ve:

count_positive += 1

else:

count_negative += 1

if count_positive > count_negative:

return $y_q = 1$

else:

return $y_q = 0$

Here the time complexity for the execution of the portion of code highlighted in yellow color is $O(nd)$.

(It is because the 'for' loop has to traverse through the entire training dataset ' D_{Train} ' of ' n ' points which takes $O(n)$ and again it has to compute the difference of the components and square them up, for all the ' d ' dimensions, which required $O(d)$. So the total time complexity for this task is $O(nd)$)

The time complexity for the execution of the portion of the code highlighted in blue color is $O(K)$. If ' K ' is very small, then it would be $O(1)$.

The time complexity for the execution of the portion of the code highlighted in green color is $O(1)$.

So now the total time complexity to run this code = $O(nd) + O(1) + O(1) = O(nd)$
If $d \ll n$, then this total time complexity = $O(n)$.

Now we shall check the space complexity. The space complexity is the total number of space locations needed to run the program. For the testing phase of KNN, the total space needed is $O(nd)$.

Conclusion

Test Time Complexity for KNN = $O(nd)$

Test Space Complexity for KNN = $O(nd)$

29.10 K-NN Limitations

a) High Space Complexity

Space Complexity of KNN = $O(nd)$

Where 'n' → number of data points, 'd' → number of dimensions

Let us now consider the Amazon Fine Food Reviews Dataset where n = 364K and d = 100K (let us assume, because we use techniques like BOW/TF-IDF)

Now the data matrix contains 364K * 100K elements.

So the total space needed = 364K * 100K = ~36GB of RAM is required. So the space complexity is so high.

b) High Time Complexity

In Amazon Fine Food Reviews Dataset, if a review is given, the system should be able to give the result within less than a millisecond.

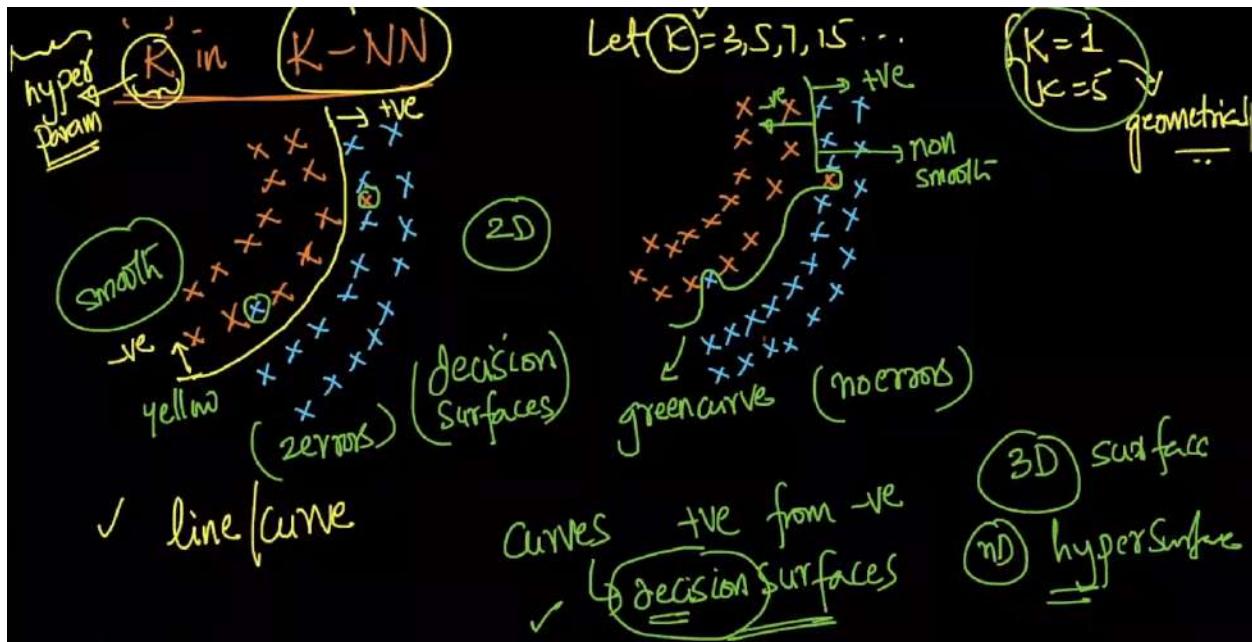
But here the system has to perform 36 billion computations. So the time complexity is so high.

Solutions to address this problem

- a) The simple implementation of KNN has a time complexity of $O(nd)$ and a space complexity of $O(nd)$. So we have to implement KNN with better time and space complexities. (**or**)
- b) Just change the algorithm. KNN is simply intuitive and elegant, but the reason for not being used is because of large time and space complexities.

Note: When we process such a huge amount of data on distributed systems, we'll have multiple threads processing the data. But still the time complexity can't be affected by the usage of multiple threads, whereas the total computation time reduces.

29.11 Decision Surface for K-NN as 'K' changes



'K' in K-NN is referred to as a Hyperparameter. Let us assume we have two different datasets and their training points are as shown above.

In the first dataset, we could see a smooth curve separating the '+ve' and the '-ve' classes. There are a few misclassifications, but still the curve is smooth. Whereas in the second dataset, the curve is not smooth, but all the points are classified perfectly. So when the curve is smooth, there are chances for a few classifications in this context, whereas if the curve is non-smooth, there are more chances for proper classifications.

The line/curve that separates the points belonging to two different classes is called a decision surface.

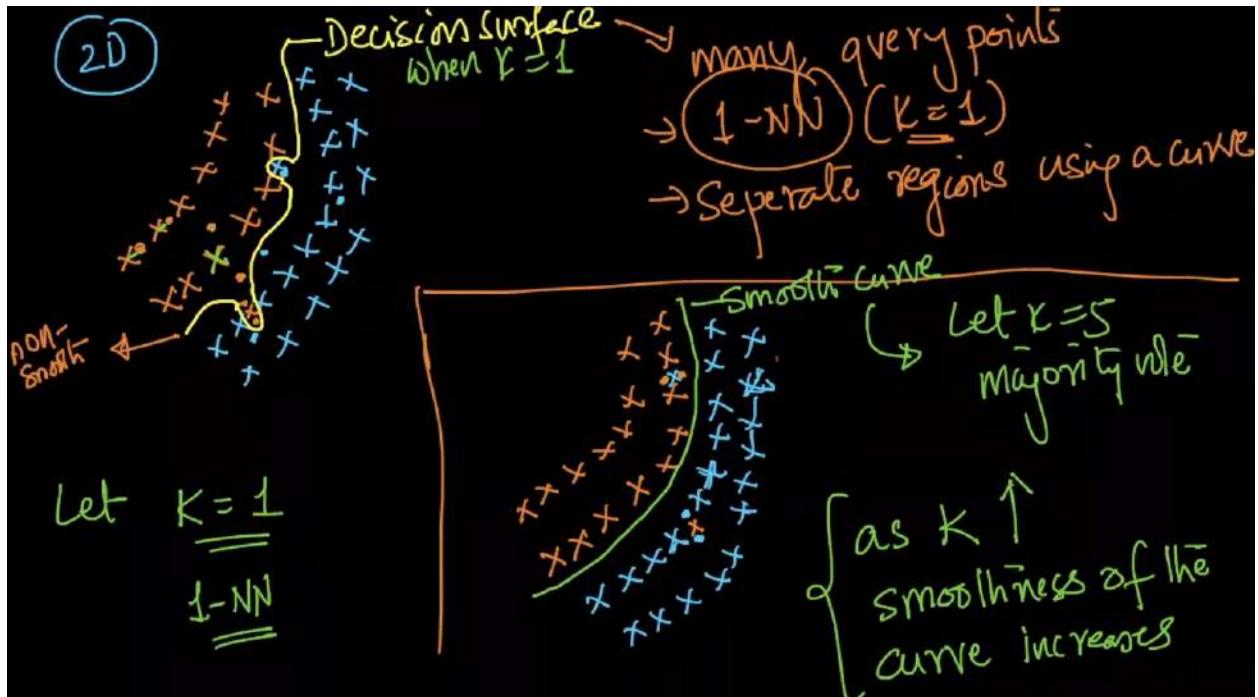
In 2-D space, we call the decision surface a line/curve.

In 3-D space, we call the decision surface a surface.

In n-D space, we call the decision surface a hyper-surface.

Let us examine the datasets in detail. When we look at the dataset, where the decision surface is non-smooth, we see all the points are classified correctly. In this case, if we consider K=1, as we have a '+ve' point lying in between a group of '-ve' points, and also a '-ve' point lying in between a group of '+ve' points.

So when K=1, if any query point has this exceptional blue point (ie., the blue point lying in between the range points) as its 1-nearest neighbor, then the class assigned for this query point will be the same as that of the exceptional blue point.

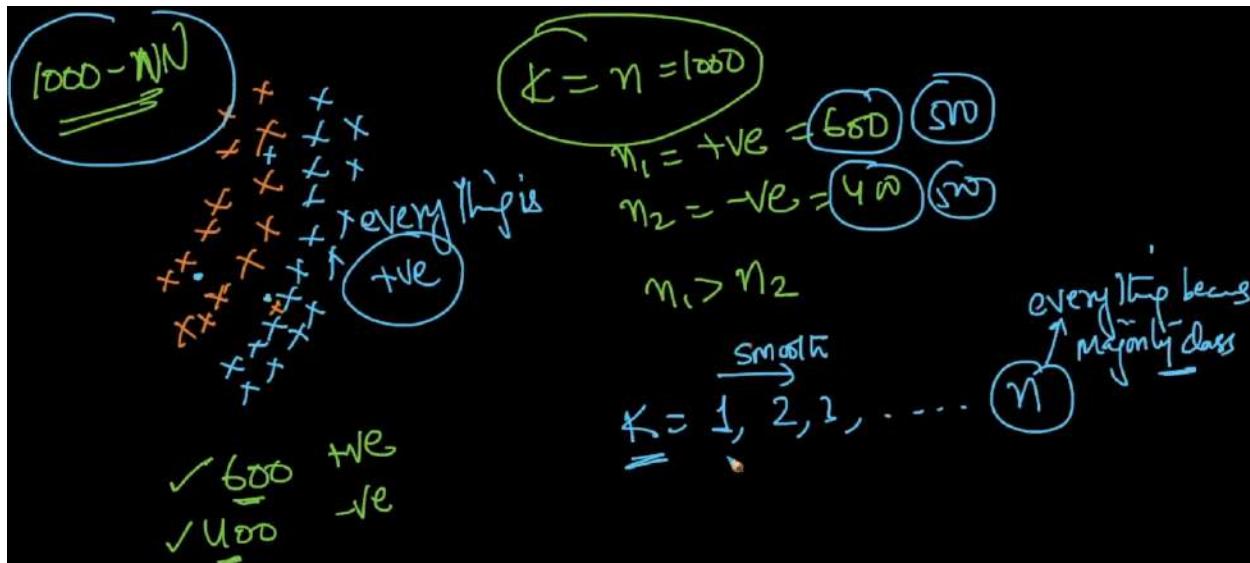


In case, if a new query point arrives with the exceptional orange point (ie., the orange point lying in between the group of blue points) as its i-nearest neighbor, then the class label assigned for this query point will be the same as this exceptional orange point.

In both the above mentioned scenarios, if these exceptional points are not the 1-nearest neighbors, then the assigned class labels would definitely change. So here we can say that, for small values of 'K', the class labels of the points keep changing as we get different points into the neighborhood, and also the decision surface is non-smooth.

In the second example, we see the decision surface is smooth. Let us assume the 'K' value as 5, then even if we have one or two exceptional points, we do not see much changes in the polarities of those points which have these exceptional points as their neighbors. Because having these one or two couldn't show much impact when we consider 5 nearest neighbors.

So we can say, as the 'K' value keeps increasing, the decision surface becomes smoother.



Let us now consider the case, where $K=n$ (where 'n' is the total number of points in the training dataset). Here we do not see the model putting much effort for predictions. It blindly goes with the majority class.

For example, if our dataset has 1000 data points, out of which 600 belong to the '+ve' class, and the remaining 400 belong to the 've' class, then for every query point, the KNN model assigns the majority class label. In this case, all the query points will be assigned with the '+ve' class label. This happens in case of an imbalance dataset(where the number of points belonging to a particular class differ from that of another classes)

For example, if our dataset has 1000 data points, out of which 600 belong to the '+ve' class, and the remaining 400 belong to the 've' class, then for every query point, the KNN model assigns the majority class label. In this case, all the query points will be assigned with the '+ve' class label.

For example, if our dataset has 1000 data points, out of which 500 belong to the '+ve' class, and the remaining 500 belong to the 've' class, then the model becomes indecisive and couldn't assign the accurate class label. It picks one of the labels randomly and assigns it to the query point.

29.12 Overfitting and Underfitting



In the above scenarios, we can clearly observe that the decision surface becomes complex, if the 'K' value is low. It is because the KNN tries to take the least number of neighbors into consideration while predicting the class label of a query point. As we are taking only a small number of points in the neighborhood for consideration, the prediction on the query point can easily get affected.

For $K=1$, we take the nearest point into consideration. If we slightly increase the 'K' value (say $K=3$), then if the other class points (not the class label that was assigned when $K=1$) are in majority, then the class label prediction changes. So for small 'K' values, the decision surface keeps changing easily. Hence we see a complex decision surface. As the value of 'K' keeps increasing, the decision surface keeps getting smoother.

For example, if we consider $K=5$, then we could see the decision surface has become smoother when compared to that of $K=1$. Here the prediction we make would be of more confidence when compared to $K=1$. So as the 'K' value keeps increasing, we can give our predicted result with more confidence.

Similarly, if $K = n$ (where 'n' is the total number of points in the training dataset), then it always predicts the majority class as the class label for a given query point. In this scenario, the model just gives the predictions blindly.

In overfitting, the decision surface does its maximum job to classify all the points accurately and hence it becomes non-smooth and more complex. In the case above where $K=1$, the exceptional points may be noise/outliers, but still the decision surface tries to classify them as accurately as possible. Hence the decision surface is working

extremely hard. In such a case, a small change in the 'K' value will affect the decision surface severely.

In underfitting, the decision surface is underworking and blindly gives the majority class label as the prediction. Hence even if you make a small change in the value of 'K', you do not find much difference in the predictions.

In case of a well-fit or best-fit, the decision surface is smoother and tries to classify the maximum number of points correctly (except the noise/outliers). Well-fit (or) Best-fit creates a balance between overfitting and underfitting.

Note:

If 'K' is small (say K=1), then the KNN model overfits

If 'K' is large (say K=n), then the KNN model underfits.

If 'K' is an intermediate value, then the KNN model fits perfectly.

	Train Accuracy	Test Accuracy	Variance	Bias
Overfitting	High	Low	High	Low
Best/Well Fit	Moderate/High	Moderate/High	Moderate	Moderate
Underfitting	Low	Low	Low	High

Note: In overfitting, if those exceptional points are noise/outliers, then the model makes a lot of errors because it has taken noise as the data, thereby creating a decision surface that is far from the truth. This decision surface is more prone(easily affected) to the outliers/noise.

In case of a well-fit model, those exceptional points are considered as noise and are ignored while building the decision surface. This decision surface is less prone to the outliers/noise when compared to the decision surface obtained with overfitting.

Q) When the model in overfitting is working more and harder to give perfect results, why don't we consider that?

Ans) It is because in overfitting, the model will work hard to give perfect results on the training data, but not on the test data. We actually care for the performance on the test data, rather than the performance on the training data. When the model overfits, it predicts well on the training data, but not on the test data.

When we have noise/outliers in our training data, if we consider overfitting, the model fits very well to all the points in the training data including the noise/outliers, but does not fit well on the test data points. Hence we ignore the models that overfit.

29.13 Need for Cross-Validation

The main purpose of performing cross-validation is to find out the optimal hyperparameter 'K' value for the K-NN model. Let our given dataset be represented mathematically in the form of a set as

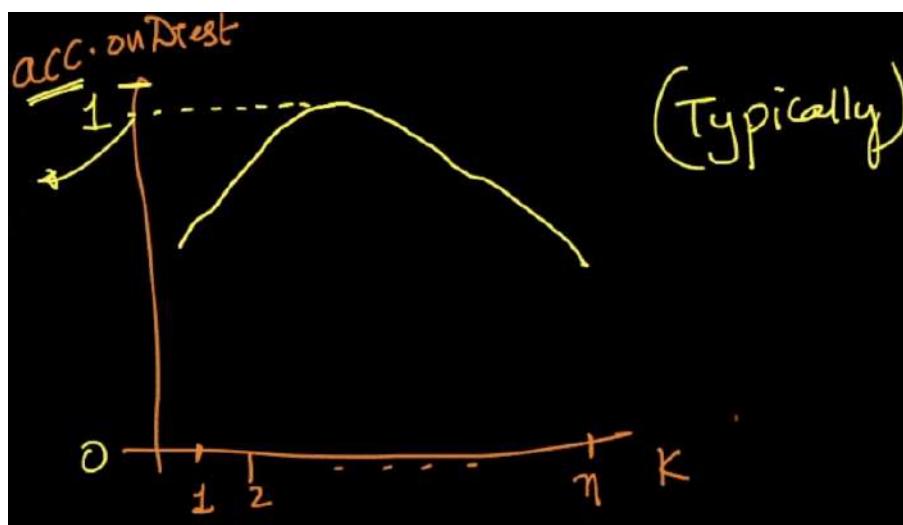
$$D = \{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{0, 1\}\}$$

How to choose the 'K' value?

- 1) Divide the dataset 'D' into the training set (D_{Train}) and the test set (D_{Test}).
- 2) For different values of 'K', fit the K-NN model on ' D_{Train} ' and make predictions on ' D_{Test} ' and then compute the test accuracy.

K	Accuracy on D_{Test}
K=1	accuracy ₁
K=2	accuracy ₂
.	.
.	.
K=5	accuracy ₅
.	.
.	.
.	.

- 3) Build a 2D line plot with all the 'K' values on the 'X' axis and the corresponding Test 'Accuracy' scores on the 'Y' axis, as shown below. Whichever value of 'K' yields the highest test accuracy score, that would be chosen as the optimal value.



But here we have an issue. The main goal of machine learning is to learn a function 'f' and give the best accuracy on the unseen data. If the model doesn't give the best accuracy on the unseen data, then the model itself is useless.

Here we are using ' D_{Train} ' to learn the function of finding out the nearest neighbors, ' D_{Test} ' to find out the optimal 'K' value. There needs to be an unseen dataset on which we can measure the accuracy(Here 'unseen' in the sense, it is not used either for finding out the nearest neighbors or for finding out the optimal 'K'). So this concept of splitting the dataset into two parts(ie., D_{Train} and D_{Test}) fails, as this is not serving the actual purpose of Machine Learning.

For example, if we got an accuracy of 0.96 for K=6 on ' D_{Test} ', it means that our model predicts the results correctly on ' D_{Test} ' in 96% of the cases, when K=6. But as ' D_{Test} ' has been used/seen by the model for finding out the optimal 'K', we couldn't say for sure that the same model could give the same accuracy on some unseen data.

When an algorithm works well for unseen future data, we call it **generalization**. The accuracy computed from the predictions made on such unseen data is called Generalization Accuracy. So we should split the dataset in such a way that we also get an unseen set of data points, to compute the generalization accuracy.

So we split the dataset into 3 parts. They are the training dataset (D_{Train}), cross-validation dataset (D_{cv}) and test dataset(D_{Test}).

D_{Train} → Used for learning the function 'f' to obtain the nearest neighbors

D_{cv} → Used for finding out the optimal 'K' value

D_{Test} → The unseen data used for computing the generalization accuracy.

Procedure for Cross-Validation (Simple Cross-Validation) on K-NN using D_{Train} , D_{cv} and D_{Test}

- 1) For different values of 'K', fit the K-NN model on ' D_{Train} '.
- 2) For every value of 'K', after fitting the model on ' D_{Train} ', make predictions on ' D_{cv} ' and compute the accuracy score for the predictions made on ' D_{cv} '.
- 3) Build a 2D line plot with all the 'K' values on the 'X' axis, and their corresponding cross-validation scores(ie., accuracies computed on ' D_{cv} ') on the 'Y' axis.
- 4) Pick the value of 'K' which yields the highest cross-validation accuracy, and that will be your optimal 'K' value.
- 5) After obtaining the optimal 'K' value, we have to fit the K-NN model on ' D_{Train} ' using the optimal 'K' value, and then make predictions on ' D_{Test} '(unseen data). Compute the accuracy for the predictions made on ' D_{Test} ' and this accuracy is called 'Generalization Accuracy.'

Q) Why do we need cross-validation?

Ans) Cross-Validation is used to assess the predictive performance of the models and to judge how well they perform on unseen data.

The motivation to use the cross-validation mechanism is that when we fit a model, we are fitting it on the training dataset(D_{Train}). Without cross-validation, we only have the information on how our model performs on the seen data.

Ideally we would like to see how does the model perform when we have new data(ie., unseen data), in terms of accuracy of its predictions.

Q) Which value of 'K' should be chosen, if multiple 'K' values give the highest cross-validation accuracy?

Ans) If we get the same highest accuracy for multiple values of 'K', then if we need the test time to obtain the nearest neighbors to be low, we should go for the least value of 'K' among those which give the highest cross-validation accuracy.

If we have lots of data, and test time is of no concern, then picking the largest value of 'K' is preferable, as we can have more trust in the decision, as there are more points supporting our decision/class label.

29.14 k'-Fold Cross-Validation

So far we have seen the dataset 'D' being split into 3 parts. They are D_{Train} (60%), D_{cv} (20%) and D_{Test} (20%). We have been using ' D_{Train} ' to find out the nearest neighbors and ' D_{cv} ' to find out the optimal 'K' value, but because of this, after obtaining the optimal 'K', we are training the final optimal model on ' D_{Train} ' and making predictions on ' D_{Test} '. The ' D_{cv} ' is not at all used anywhere after getting the optimal 'K'. So it goes to waste.

In order to make the maximum usage of our data and not letting any subset of data go to waste, we have come up with a strategy called K-fold Cross-Validation. For this, we have to split the dataset 'D' only into two parts. They are ' D_{Train} '(80%) and ' D_{Test} '(20%). The ' D_{cv} ' gets created internally during the cross-validation. More the data used for the training, the more the model's predictive power would be. But we cannot skip the test data(D_{Test}). So the ' D_{cv} ' would be created from D_{Train} .

Procedure for k'-Fold Cross-Validation

- 1) The dataset 'D' is split into ' D_{Train} ' and ' D_{Test} '.

$D_{Train} \rightarrow$ Finding out the Nearest Neighbors and obtaining the optimal 'K'(hyperparameter of K-NN) value

$D_{Test} \rightarrow$ Unseen data on which the final model has to run and compute the accuracy.

- 2) The training data (D_{Train}) is divided into k' parts(let us assume k'=4 for now).

So the 4 parts would be ' D_1 ', ' D_2 ', ' D_3 ' and ' D_4 '.

D_{Train}

D_1	D_2	D_3	D_4
-------	-------	-------	-------

In k'-fold CV, the number of parts into which ' D_{Train} ' is divided is equal to k'.

3)

	D_{Train}	D_{cv}	Accuracy on D_{cv}
K=1	$D_1 D_2 D_3$	D_4	$a_4^{(1)}$
K=1	$D_1 D_2 D_4$	D_3	$a_3^{(1)}$
K=1	$D_1 D_3 D_4$	D_2	$a_2^{(1)}$
K=1	$D_2 D_3 D_4$	D_1	$a_1^{(1)}$
K=2	$D_1 D_2 D_3$	D_4	$a_4^{(2)}$
K=2	$D_1 D_2 D_4$	D_3	$a_3^{(2)}$
K=2	$D_1 D_3 D_4$	D_2	$a_2^{(2)}$
K=2	$D_2 D_3 D_4$	D_1	$a_1^{(2)}$
.	.	.	.

Avg. Accuracy Score for K=1 → $(\mathbf{a}_1^{(1)} + \mathbf{a}_2^{(1)} + \mathbf{a}_3^{(1)} + \mathbf{a}_4^{(1)})/4$. Let us denote this as ' $\mathbf{a}_{k=1}$ '.

Avg. Accuracy Score for K=2 → $(\mathbf{a}_1^{(2)} + \mathbf{a}_2^{(2)} + \mathbf{a}_3^{(2)} + \mathbf{a}_4^{(2)})/4$. Let us denote this as ' $\mathbf{a}_{k=2}$ '. Like this, we have to compute the average CV scores for all the values of 'K'. (Here 'K' is the hyperparameter in K-NN)

- 4) We should now plot a 2D line plot, with the 'K'(hyperparameter in K-NN) values on the 'X' axis, and their corresponding average CV accuracies on the 'Y' axis.

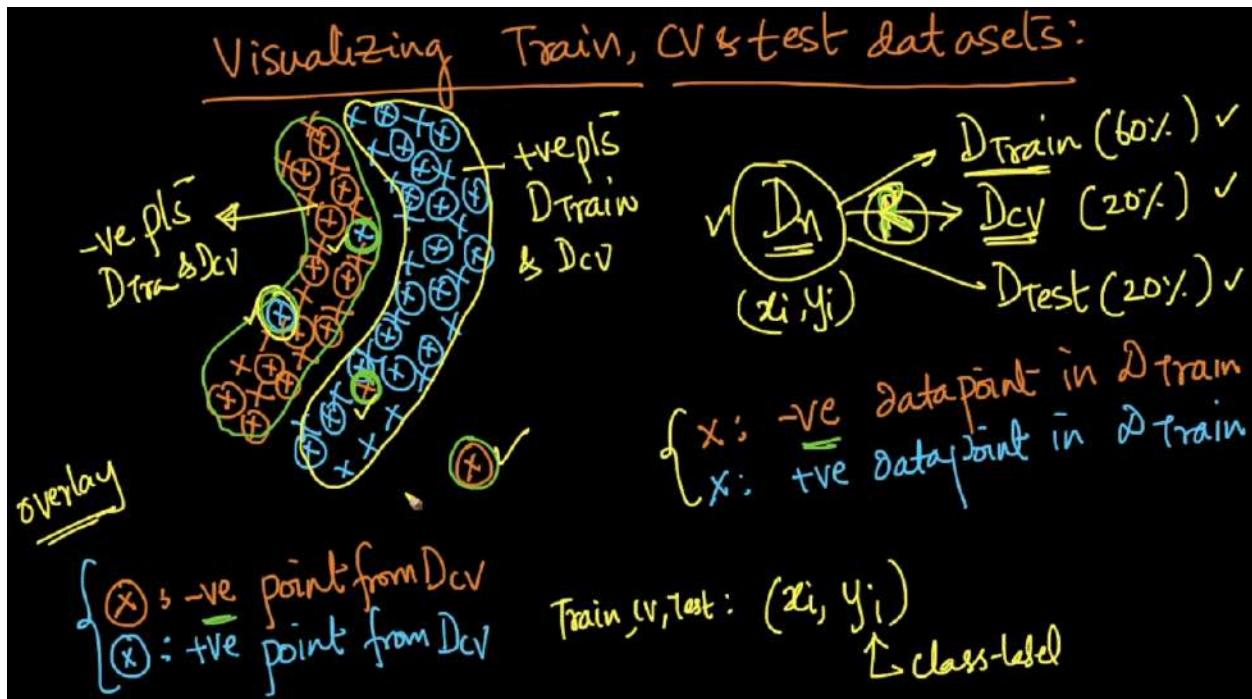
Whichever value of 'K' gives the highest average CV accuracy score, that would be considered as the optimal 'K' value.

- 5) Fit the final K-NN models using the obtained optimal 'K' value on the whole ' D_{Train} ', and make predictions on ' D_{Test} '(unseen data), and compute the final test accuracy score.

Note: In k'-fold cross-validation, the training data ' D_{Train} ' is divided into k' folds and each fold is used in both the training and the cross-validation phases. The most typically used values for k' are 3 (or) 5.

K'-fold cross-validation gives the optimal value of the hyperparameter such that the results obtained on the unseen data are more generalized.

29.15 Visualizing train, validation and test datasets



For now we shall split the dataset ' D_n ' into 3 parts. They are ' D_{Train} '(60%), ' D_{cv} '(20%) and ' D_{Test} (20%)'. Every data point (irrespective of whether it is present in ' D_{Train} ', ' D_{cv} ' and ' D_{Test} ') is represented as (x_i, y_i) .

Observations:

- 1) ' D_{Train} ' and ' D_{cv} ' do not overlap perfectly.
- 2) If there are many +ve/-ve points from ' D_{Train} ' in a region, then it is highly likely to find many +ve/-ve points from ' D_{cv} ' in that region.
- 3) If there are a few +ve/-ve points in a region from ' D_{Train} ', then it is very unlikely to find +ve/-ve points from ' D_{cv} ' in that region. Such points are noise/outliers.

All the above 3 observations are True, as long as ' D_{Train} ' and ' D_{cv} ' are randomly sampled. We also could see the above observations between ' D_{Train} ' and ' D_{Test} ', if they both are randomly sampled.

29.16 How to determine overfitting and underfitting?

Either by using Simple cross-validation (or) k'-fold cross-validation, we get the best value of 'K'(here hyperparameter in K-NN) for the model, which neither leads the model to overfit nor to underfit.

Accuracy = (Number of points correctly classified)/(Total number of points)

Error = 1 - Accuracy

Our main aim is always to maximize the accuracy and minimize the error.

For now, we shall consider the case of simple cross-validation, for that we have to split the dataset ' D_n ' into ' D_{Train} ', ' D_{cv} ' and ' D_{Test} '.

In simple cross-validation, we use the ' D_{Train} ' to find the nearest neighbors, ' D_{cv} ' to find the optimal 'K' value and ' D_{Test} ' to find out the model performance at prediction on the unseen data.

Training Error

We have to fit the model on ' D_{Train} ' and make predictions on the same ' D_{Train} '. Here we come across a few misclassifications while predicting the class labels of ' D_{Train} '. This error obtained is called the **Training Error**.

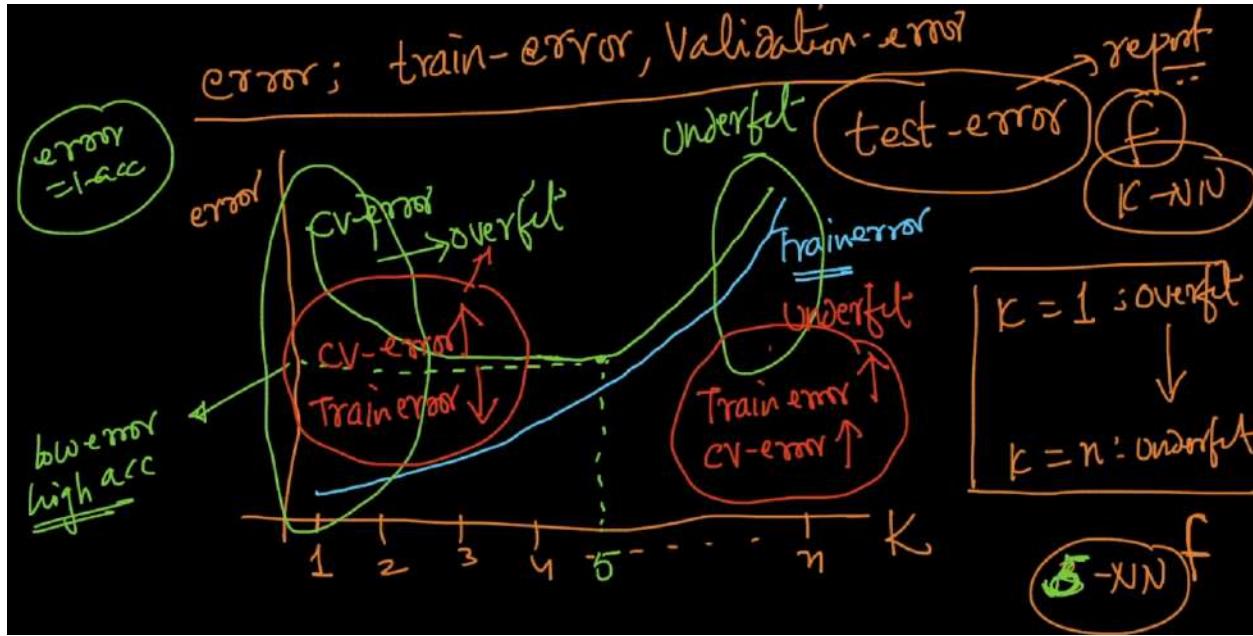
Cross-Validation Error

We have to fit the model on ' D_{Train} ' and make predictions on ' D_{cv} '. Here we come across a few misclassifications while predicting the class labels of ' D_{cv} '. This error obtained is called the **Cross-Validation Error**.

Test Error

We have to fit the model on ' D_{Train} ' and make predictions on the same ' D_{Test} '. Here we come across a few misclassifications while predicting the class labels of ' D_{Test} '. This error obtained is called the **Test Error**.

Overfitting (vs) Underfitting



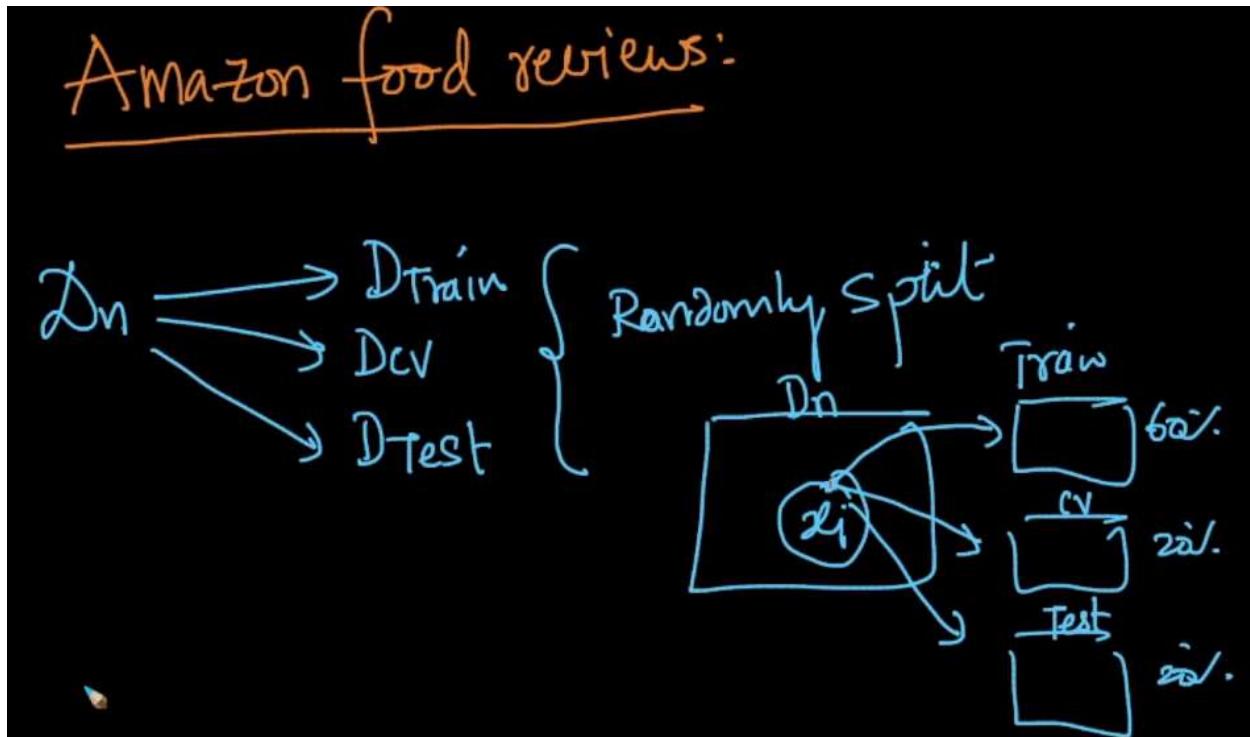
If the **Training Error** is high and the **Cross-Validation Error** is high, then we call it **Underfitting**.

If the **Training Error** is low, but the **Cross-Validation Error** is high, then we call it **Overfitting**.

If the **Training Error** is moderate and the **Cross-Validation Error** is moderate, then we call it the **Best Fit**. (Here both the errors are close enough)

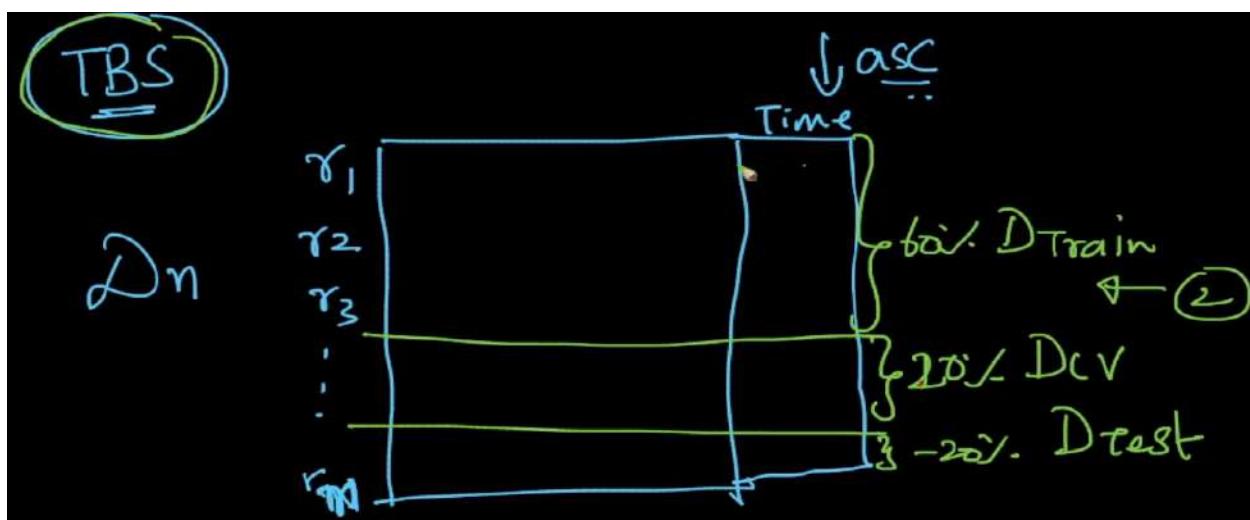
29.17 Time Based Splitting

So far we have split the dataset into D_{Train} , D_{cv} and D_{Test} using random splitting.



We also do have another strategy of splitting the dataset which is known as Time Based Splitting. There are certain problems where Random Splitting is better than Time Based Splitting, and in some problems, Time Based Splitting is better than Random Splitting.

In order to perform Time Based Splitting, we need to have the timestamp column in the dataset.



Procedure to perform Time Based Splitting

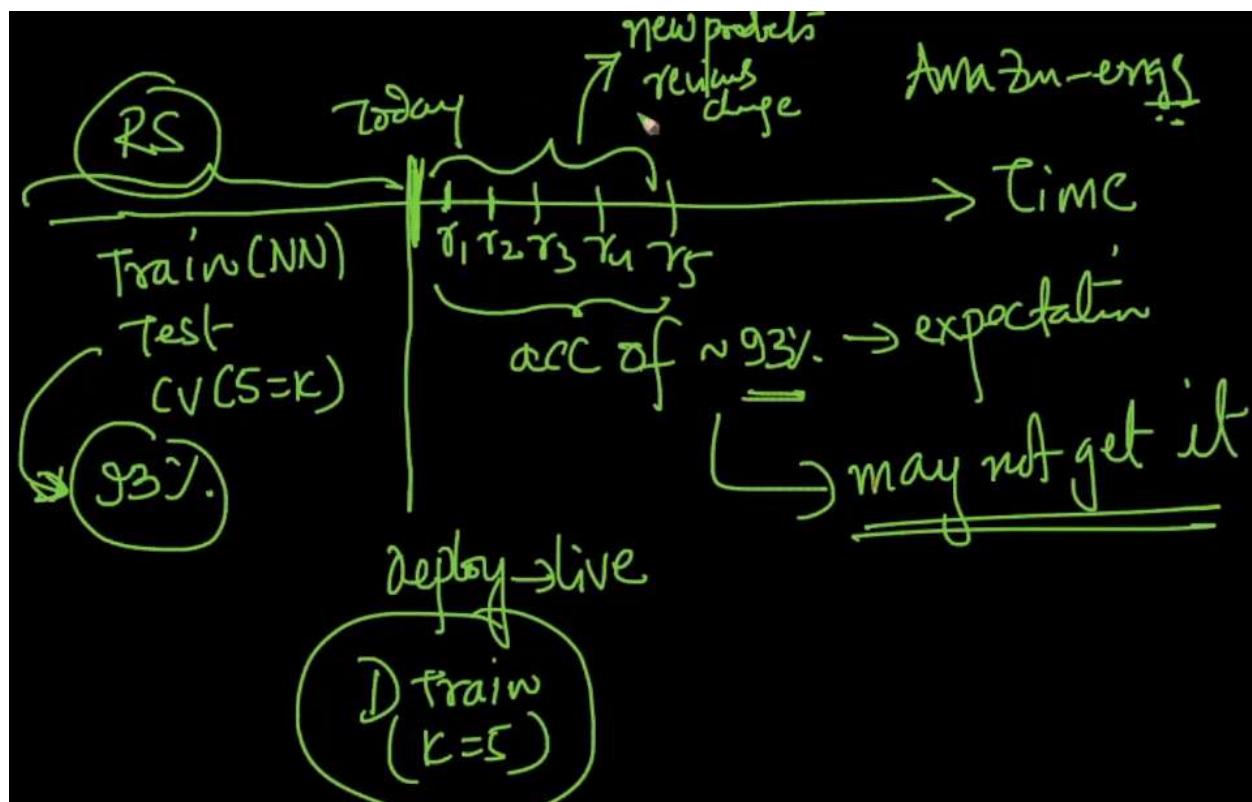
- 1) Sort the dataset ' D_n ' in the ascending order of the time column values.
- 2) Now we split the dataset with the first 60% of the points into ' D_{Train} ', the next 20% of the points into ' D_{cv} ', and the last 20% of the points into ' D_{Test} '.

In case of Random Splitting, we had

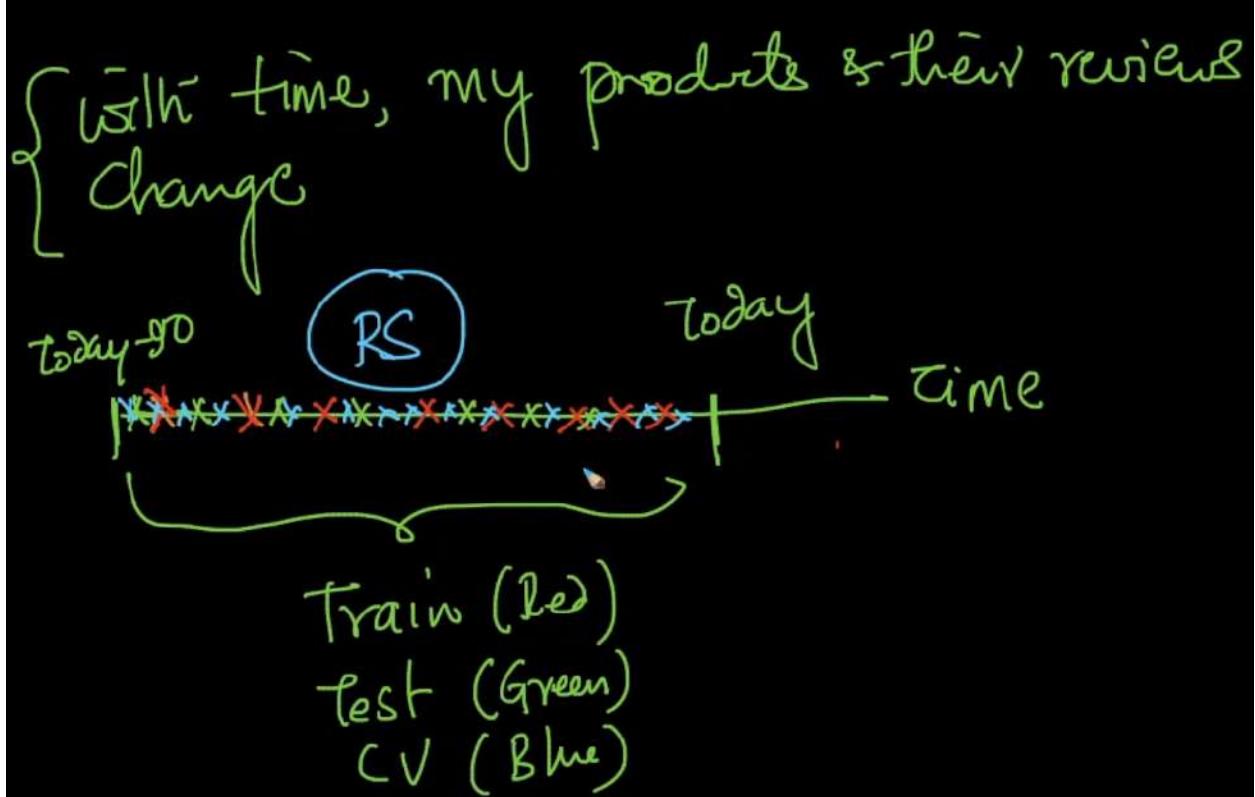
D_{Train} → For finding the nearest neighbors

D_{cv} → For choosing the optimal 'K' value

D_{Test} → For measuring the accuracy



The above figure shows applying random splitting on a real-time dataset. Here for the products, the reviews would change as the time keeps progressing. It doesn't mean the already existing reviews would change, but as the time keeps progressing, we do get many more reviews on the similar products purchased, and the reviews/feedback would change. In such a case, if we perform a random splitting, there are chances for a few of the latest reviews to go into the D_{Train} and D_{cv} , and a few of the older reviews to go into D_{Test} . Due to this, the model once trained in the past and gave a good score on D_{Test} , if it is deployed, might not give a similar score on the latest data. It is because as the patterns in the data keep changing, the model performance also keeps changing.

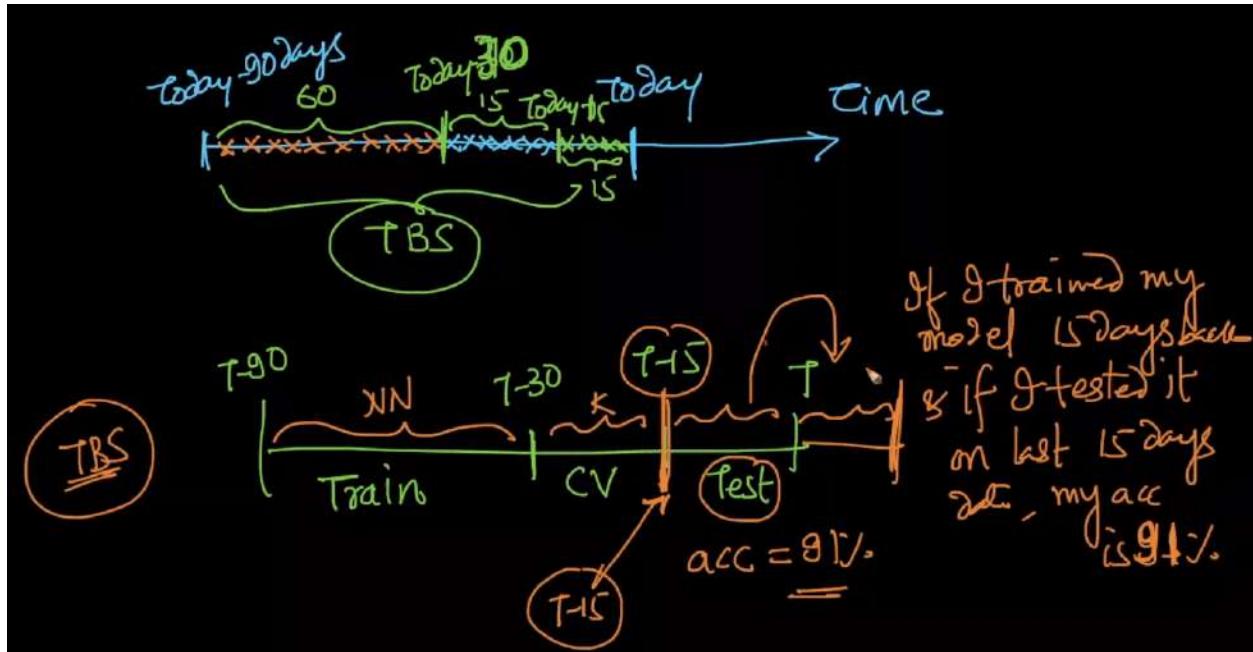


Also there are chances for new products to get added, and also the older products to disappear. In case, if a new product gets added today on the website, and if there is no data about similar products previously, then we couldn't make a valid prediction. Similarly, if we have a product that is no more available for sale, then it is always better to keep it in D_{Train} because, if it comes in D_{Test} , then making predictions again on it, is just a waste of time and meaningless, as that product is never going to be added again for sale. Hence in order to get rid of all these kinds of problems, we go for Time Based Splitting.

In case of the time based splitting, we split the older data into D_{Train} and D_{cv} , and then the future data would be D_{Test} .

Whenever we apply Random Splitting, if we train, test and deploy the model, and if it gives an accuracy of say $x\%$, then in future the same model might not give similar accuracy for the future data, whereas whenever we apply time-based splitting, if we train a model with a particular set of data belonging to a particular interval, cross-validate on a particular interval and test the model on a particular interval of data and finally deploy it, then the accuracy obtained on the test data will be the almost the same for the future data as well.

Note: People wrongly understand that the consistency in the accuracy as mentioned above is due to the presence of the 'Time' column in our dataset. It is not because of the presence of the 'Time' column, but due to the way in which we split the dataset.



Note: Whenever the 'Time' column is available in the dataset, and if things/behavior of the data changes over the time, then time-based splitting is preferred over random splitting.

In case, if the 'Time' column is not present in the dataset, then we have to go for Random Splitting, as we have no other option.

29.18 K-NN for Regression

The dataset for a binary classification task is represented as

$$D = \{(x_i, y_i)_{i=1}^n \mid x_i \in \mathbb{R}^d, y_i \in \{0,1\}\}$$

The dataset for a regression task is represented as

$$D = \{(x_i, y_i)_{i=1}^n \mid x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}$$

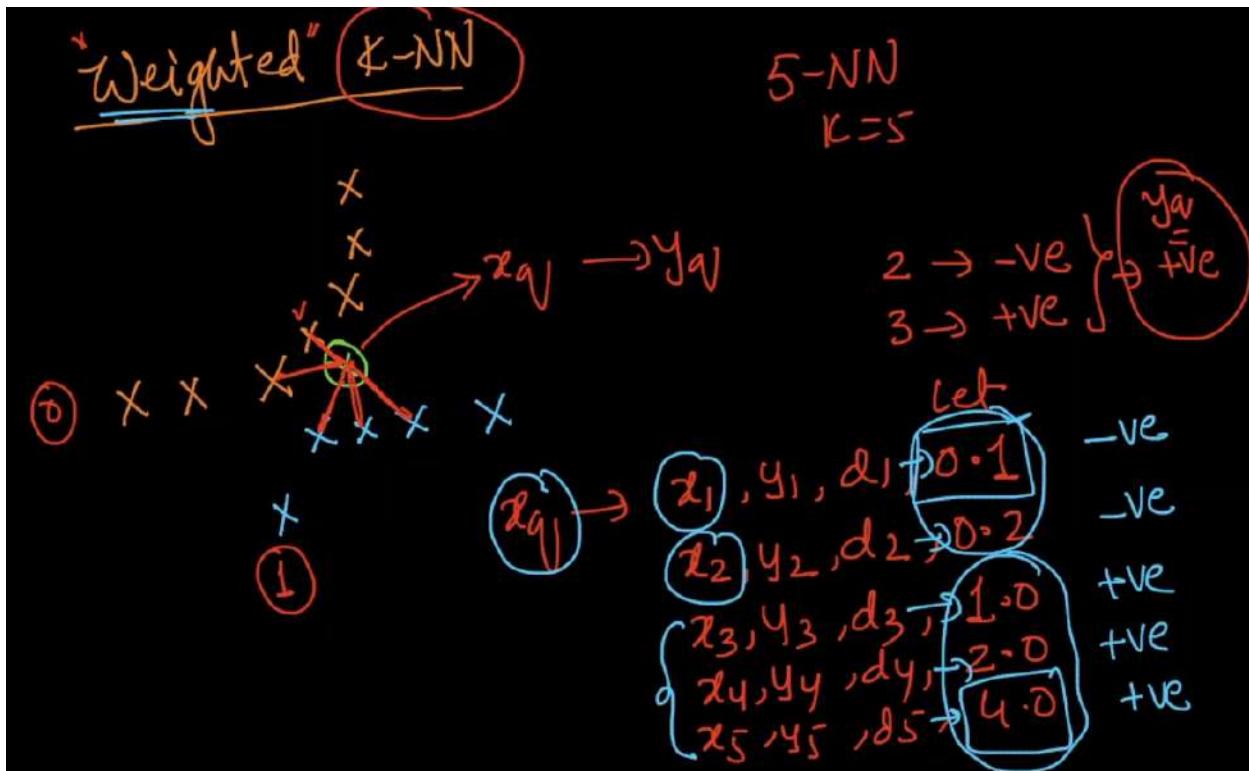
Procedure of K-NN for Regression

- 1) Given ' x_q ', find the 'K' nearest neighbors. Let them be
 $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_k, y_k)$
- 2) Find y_q' from $y_1, y_2, y_3, \dots, y_k$ using the formula
 $y_q' = \text{mean}(y_i)_{i=1}^k$ (or) $y_q' = \text{median}(y_i)_{i=1}^k$
(The median is less prone to the outliers when compared to the mean)

Note: In classification using K-NN, we go with the majority vote, whereas in regression using K-NN, we go with either the mean or the median of the output values of the 'K' nearest neighbors.

It is recommended to go with the median, because median is less prone to the outliers/noise when compared to the mean.

29.19 Weighted K-NN



So far we have seen simple K-NN. Let us now look at the weighted K-NN. Let us assure we are working on a 5-NN problem. So let us assume we have to predict the class label for ' x_q '.

Let ' d_1 ', ' d_2 ', ' d_3 ', ' d_4 ' and ' d_5 ' be the distances of ' x_q ' from the 5 nearest neighbors ' x_1 ', ' x_2 ', ' x_3 ', ' x_4 ' and ' x_5 ' respectively. Let ' x_1 ' and ' x_2 ' be the negative points and ' x_3 ', ' x_4 ' and ' x_5 ' be the positive points. Let us assume

Distance between ' x_1 ' and ' x_q ' = $d_1 = 0.1$

Distance between ' x_2 ' and ' x_q ' = $d_2 = 0.2$

Distance between ' x_3 ' and ' x_q ' = $d_3 = 1$

Distance between ' x_4 ' and ' x_q ' = $d_4 = 2$

Distance between ' x_5 ' and ' x_q ' = $d_5 = 3$

So now we need to calculate the weights. Let us consider the weighted function for the a point ' x_i ' as $w_i = 1/d_i$

As the points ' x_1 ' and ' x_2 ' are the negative points, $w_1 + w_2 = (1/0.1) + (1/0.2) = 10 + 5 = 15$

As the points ' x_3 ', ' x_4 ' and ' x_5 ' are the positive points, $w_1 + w_2 + w_3 = (1/1) + (1/2) + (1/3) = 1 + 0.5 + 0.33 = 1.83$

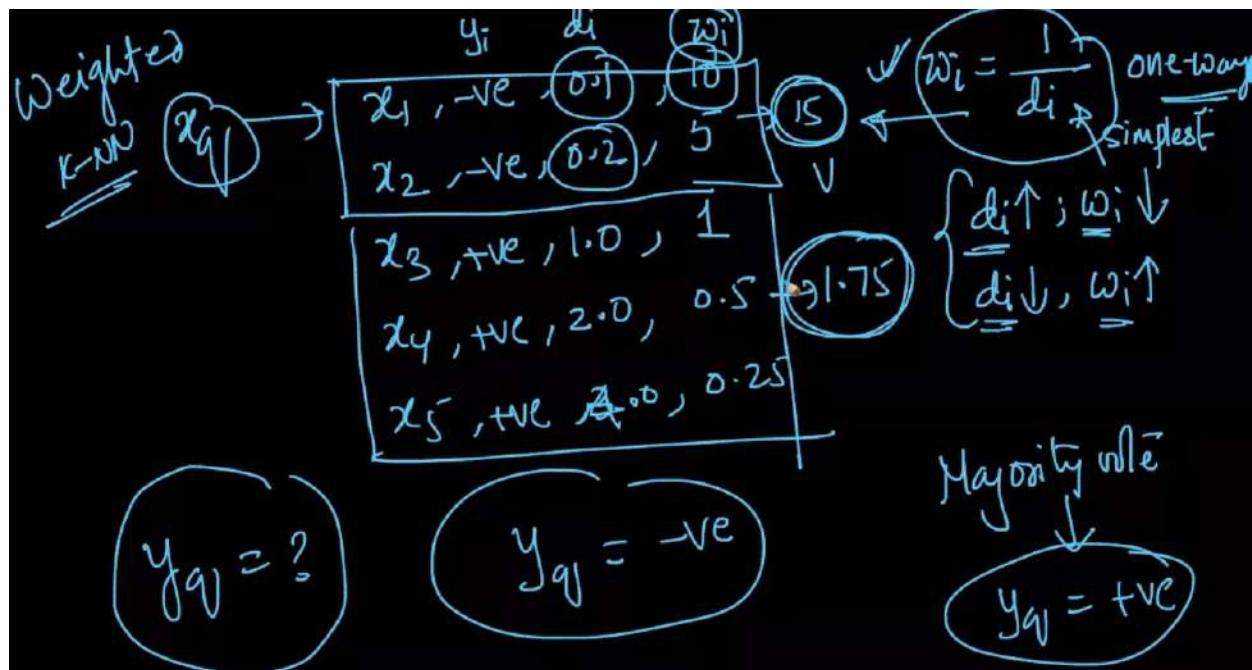
So in simple K-NN, we just go blindly with majority voting. So ' x_q ' will be classified as 'positive' (as the majority count is 'positive')

In the weights K-NN, we go with the weights of the classes. So

Total Weight of the 'negative' class = $w_1 + w_2 = 15$

Total Weight of the 'positive' class = $w_3 + w_4 + w_5 = 1.83$

So in the neighborhood of ' x_q ', the total weight of the 'negative' class points is more than the total weight of the 'positive' class points. Hence the point ' x_q ' is classified as 'negative' by weighted K-NN.



Q) When should we choose Weighted K-NN over Simple K-NN?

Ans) Weighted K-NN is used when we want to prefer closer points over farther points among the 'K' neighbors, in our decision making.

Whenever there is a requirement such that the nearest points among the nearest neighbors should contribute more in predicting the class label, then we have to go for Weighted K-NN.

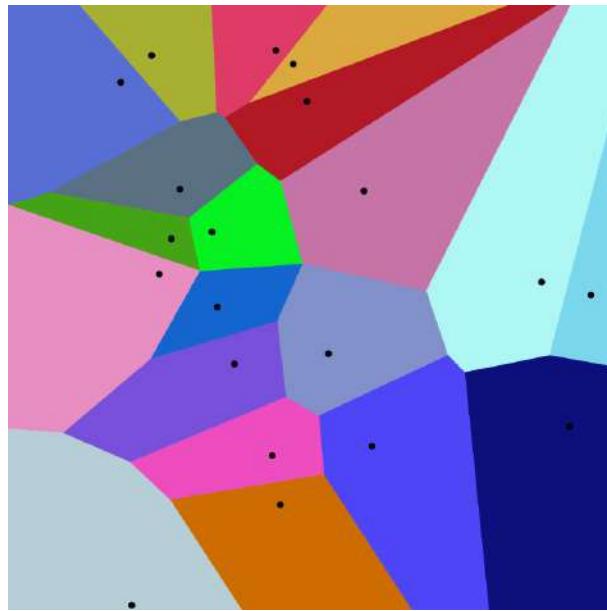
The main disadvantage with Weighted K-NN is we give too much importance to the nearby points, which could result in errors in some instances. Weighted K-NN is problem specific.

Effect of Outliers on Weighted K-NN

Even the weighted K-NN gets easily affected by the presence of the outliers. Hence it is recommended to remove the outliers before applying any model.

Along with the outliers, the scale of the features also easily affects the Weighted KNN model. So we should also make sure all the features are scaled properly, before applying any model.

20.20 Voronoi Diagram



Voronoi diagram is a partitioning of a plane into the regions based on distance to the points in a specific subset of the plane. For each seed(point), there is a corresponding region and throughout that region, if any new point falls, that seed will be the nearest point to the newly arrived point.

In 1-NN, each point has only one region, These regions are called Voronoi cells.

Note: Mostly Voronoi diagrams are not used in practice as they become very complex in high dimensional space. Voronoi diagrams are mostly used in theoretical evaluation of K-NN and in a sub-area of computer science called Computational Geometry.

The partitions dividing two regions are equidistant from the points. All the vertices are equidistant from the points. The Voronoi diagram is related to K-NN with K=1. Just like Logistic Regression has a hyperplane as the decision surface, the decision surface for 1-NN is a voronoi diagram. But in ML, voronoi diagrams aren't much used in practice.

29.21 Binary Search Tree

In simple K-NN implementation,

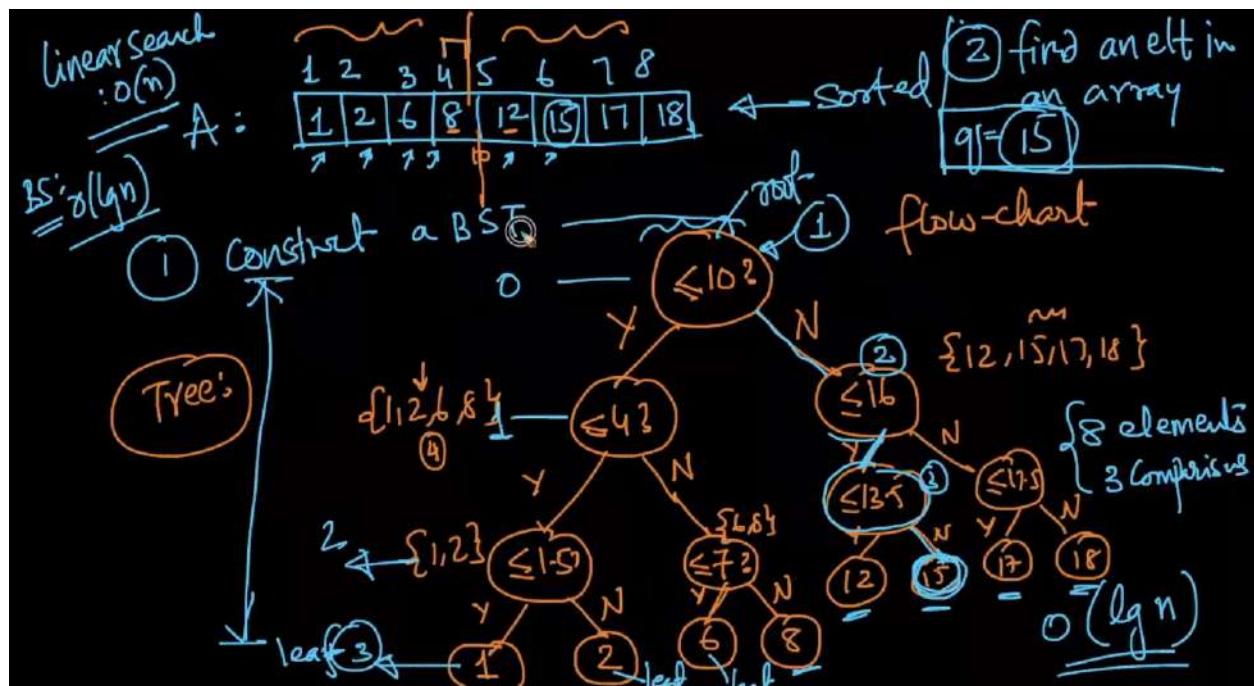
Time Complexity = $O(nd) \sim O(n)$ (if both 'K' and 'd' are small)

Space Complexity = $O(nd) \sim O(n)$ (if 'd' is small)

We could not reduce the space complexity of KNN, but can reduce the time complexity using a technique called KD-Tree. The time complexity reduces from $O(n)$ to $O(\log(n))$. For example, if there are 1024 computations to be performed, then the time complexity of simple KNN would be $O(1024)$ whereas in case of a KD-Tree, the time complexity reduces to $O(\log(n)) = O(\log(1024)) = O(10)$

KD-Tree works similar to the way how a Binary Search Tree (BST) Works. Let us now look at how a Binary Search Tree works.

Construction and Working of a Binary Search Tree



Steps of Construction

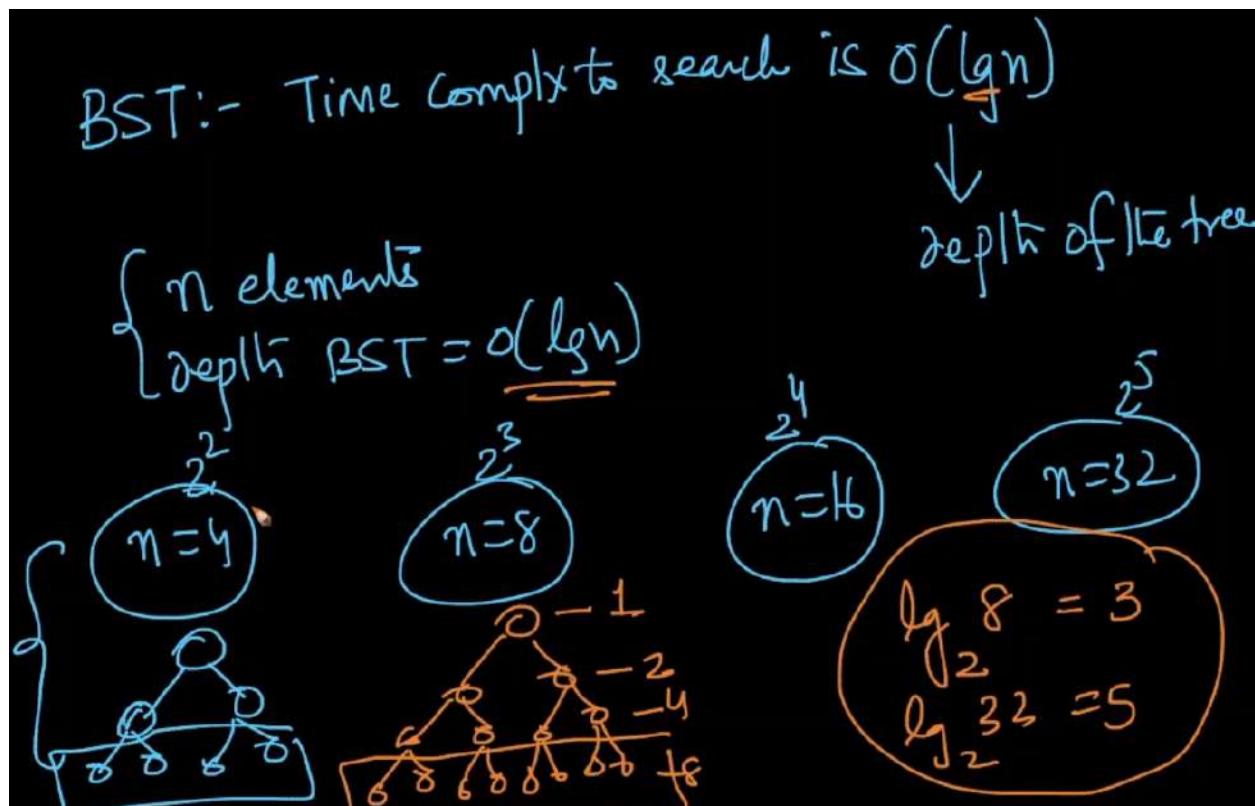
- 1) We have to first sort the given array/list of values in ascending order.
- 2) Find out the median of the values and make it a root.
- 3) All the values that are less than the median should go to the left side(i.e., left subtree) of the root, and all the values that are greater than the median should go to the right side(i.e., right subtree) of the root.

- 4) Repeat steps 2 and 3, until we are left with a single value in each node. All these nodes present in the bottom layer without any child nodes, are called the leaf nodes.

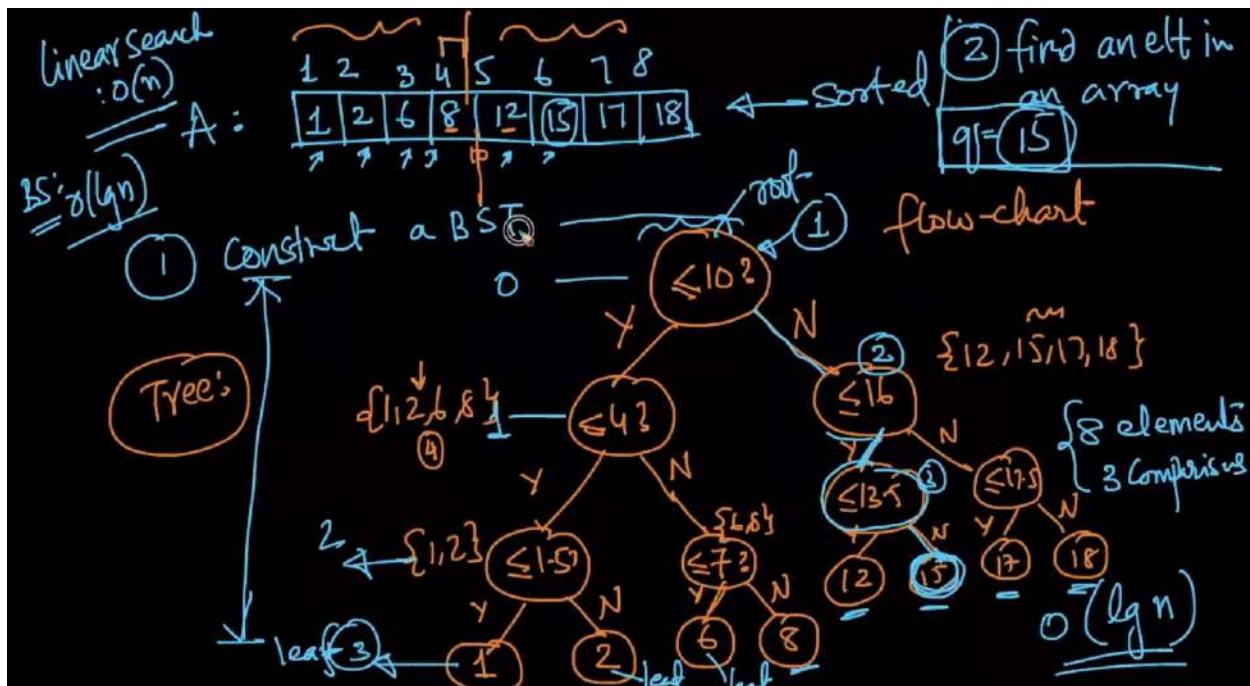
Steps for Searching an element

- 1) If an element is given, we have to compare it with the value in the root node first.
- 2) If the element is the same as the value in the root node, return True. If the element is less than the root node value, then move into the left subtree. Otherwise move into the right subtree.
- 3) Repeat steps 1 and 2, either until the given element is found or all the leaf nodes in the traversed path are also checked. If the given element is not found, return False.

Here as we are first sorting the values and then checking only a portion of the tree, rather than the entire tree. As we are just checking only a half portion in every subtree, the time complexity reduces from $O(n)$ to $O(\log n)$.



Let us check if the numbers 2, 5, 6, 16, 19 are present in the Binary Search Tree.



Checking for Number 2

- The root value is 10. Comparing '2' with '10'. As $2 < 10$, we have to check the left subtree, with '4' as the root.
- The root value is now 4. Comparing '2' with '4'. As $2 < 4$, we have to check the left subtree, with '1.5' as the root.
- Now the root value is 1.5. Comparing '2' with '1.5'. As $2 > 1.5$, we have to check the right subtree, with '2' as the root.
- Now the root value is 2. As $2 == 2$ returns **True**, the whole result turns out to be **True**. It means the specified element '2' is present in the Binary Search Tree.

Checking for Number 5

- The root value is 10. Comparing '5' with '10'. As $5 < 10$, we have to check the left subtree, with '4' as the root.
- The root value is now 4. Comparing '5' with '4'. As $5 > 4$, we have to check the right subtree, with '7' as the root.
- The root value is now 7. Comparing '5' with '7'. As $5 < 7$, we have to check the left subtree, with '6' as the root.
- The root value is now 6. Comparing '5' with '6'. As $5 < 6$, we have to check the left subtree. But here '6' is a leaf node and it has no more subtrees/branches. So as we couldn't find the element '5', it returns **False**.

Checking for Number 6

- a) The root value is 10. Comparing '6' with '10'. As $6 < 10$, we have to check the left subtree, with '4' as the root.
- b) The root value is now 4. Comparing '6' with '4'. As $6 > 4$, we have to check the right subtree, with '7' as the root.
- c) The root value is now 7. Comparing '6' with '7'. As $6 < 7$, we have to check the left subtree, with '6' as the root.
- d) Now the root value is 6. As $6 == 6$ returns **True**, the whole result turns out to be **True**. It means the specified element '6' is present in the Binary Search Tree.

Checking for Number 16

- a) The root value is 10. Comparing '16' with '10'. As $16 > 10$, we have to check the right subtree, with '16' as the root.
- b) Now the root value is 16. As $16 == 16$ then we move to the left subtree of node 16. As the root node value of the left subtree is 13.5 which is less than 16. So we return **False**.

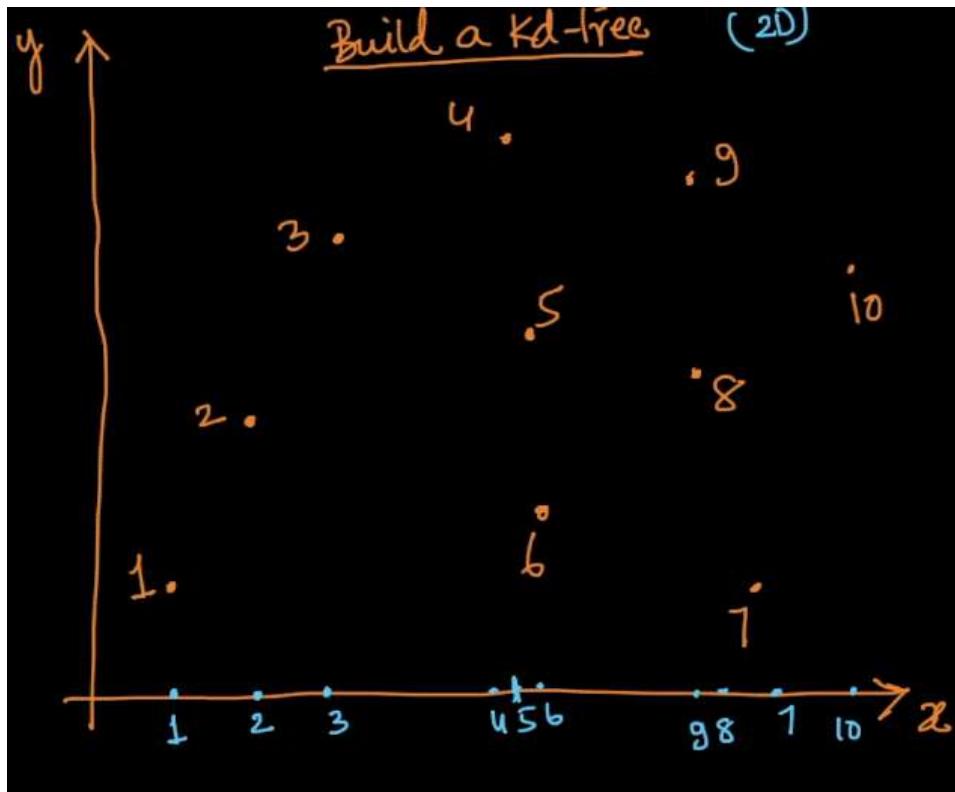
Checking for Number 19

- a) The root value is 10. Comparing '19' with '10'. As $19 > 10$, we have to check the right subtree, with '16' as the root.
- b) The root value is now 16. Comparing '19' with '16'. As $19 > 16$, we have to check the right subtree, with '17.5' as the root.
- c) The root value is now 17.5. Comparing '19' with '17.5'. As $19 > 17.5$, we have to check the right subtree, with '18' as the root. But here '18' is a leaf node and it has no more subtrees/branches. So as we couldn't find the element '19', it returns **False**.

29.22 How to build a KD-Tree

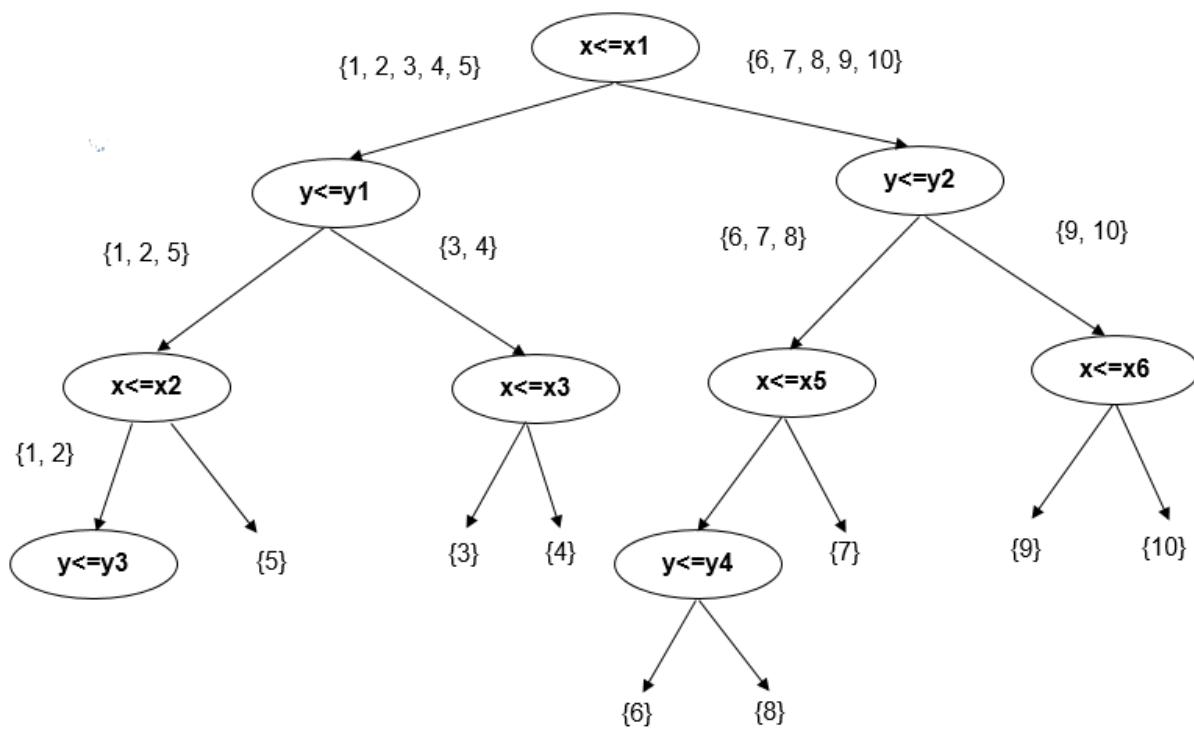
Procedure to construct a KD-Tree

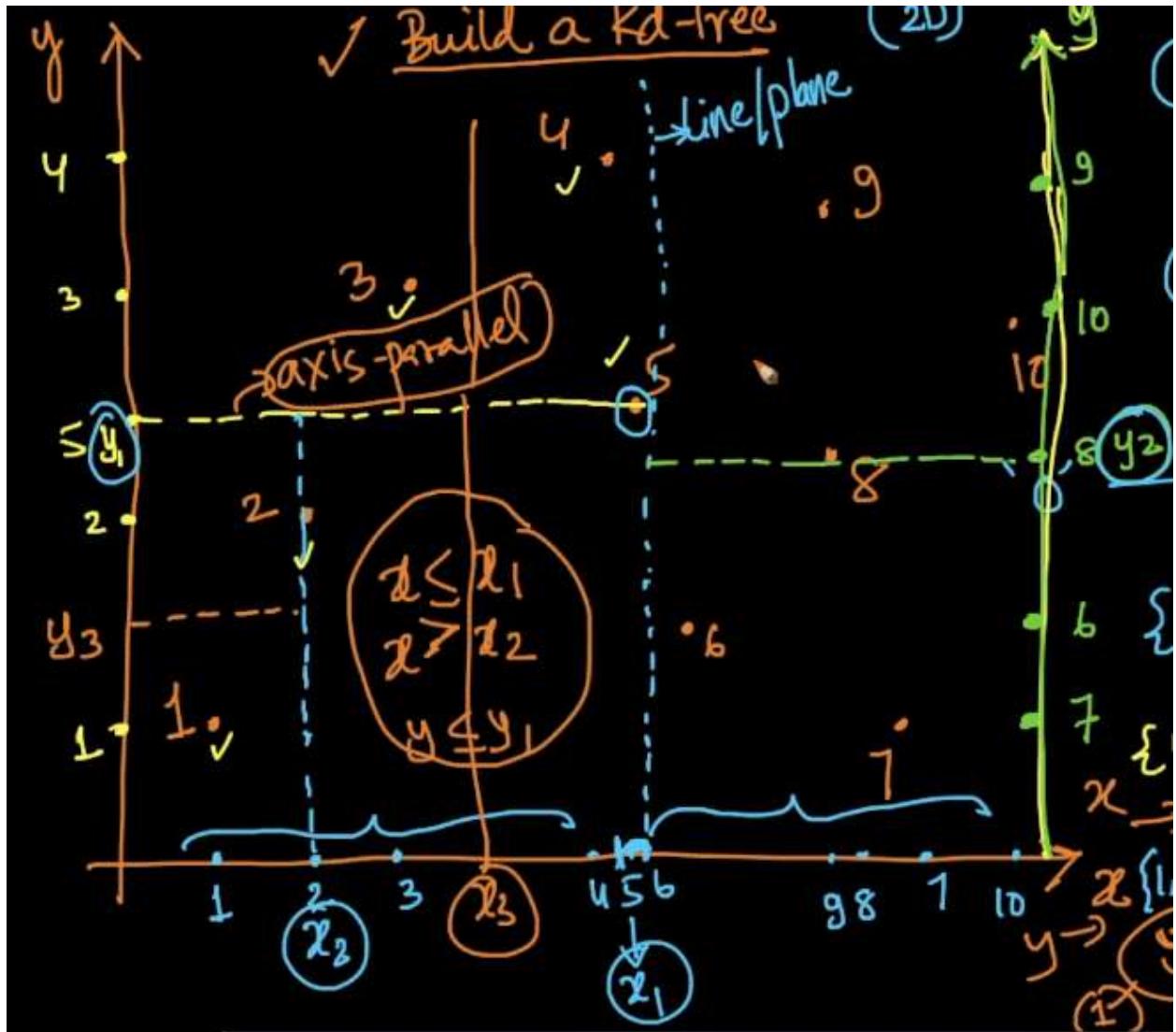
Let the given points be projected in a 2D space as shown below



- 1) Pick the 'X' axis, project all the points on the 'X' axis. Compute the median. Split the data on the basis of the median ' x_1 '.
All the values that are less than or equal to ' x_1 ' should go into the left subtree(ie., 1,2,3,4,5), and those values which are greater than ' x_1 '(ie., 6,7,8,9,10) should go into the right subtree. ' x_1 ' would be the root of the tree.
- 2) Pick the 'Y' axis, project all these points on the 'Y' axis. Compute the median. Split the data on the basis of the median ' y_1 '. Now ' y_1 ' is the root node of the left subtree.
All the values that are less than or equal to ' y_1 '(ie., 1,2,5) should go into the left subtree, and those values which are greater than ' y_1 '(ie., 3,4) should go into the right subtree.
Compute the median of the values {6,7,8,9,10} and it should be the root node of the right subtree. Let it be ' y_2 '. All the values that are less than or equal to ' y_2 ' will now go into the left subtree and the remaining values should now go into the right subtree.

This way we have to keep alternating between the axes. The final KD-Tree looks like below



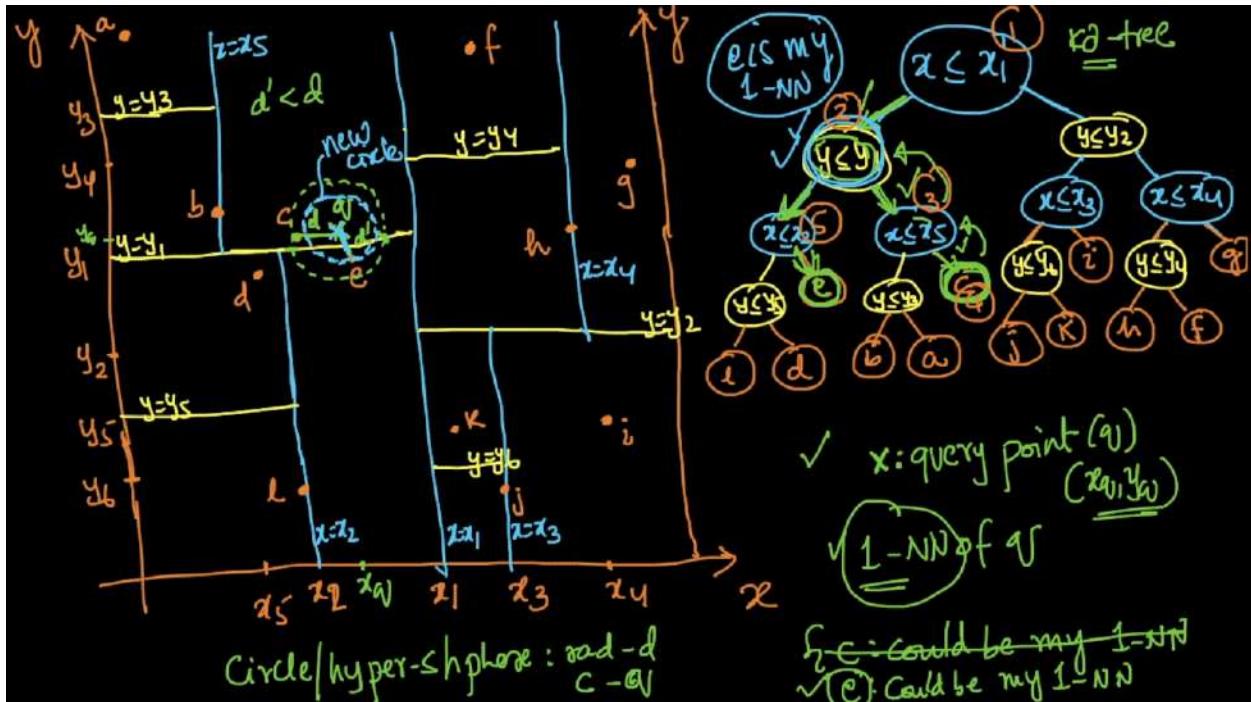


Note:

In KD-Tree, we are breaking the space using axis-parallel lines/planes into rectangles(in 2-D)/ cuboids(in 3-D)/ hyper cuboids (in n-D).

We have to switch the axes alternatively till we reach the leaf nodes. We can build the KD-Tree not only with the 'median' statistic, but also with the 'mean' statistic.' But if we go with the 'mean' statistic, we should make sure, there are no outliers in our data.

29.23 Finding Nearest Neighbors using KD-Tree



Let us assume we have to find out only the 1 nearest neighbor to a given point (x_q, y_q) . In the same way as we did in the previous topic, here instead of using 'x', we have to use ' x_q ' and instead of using 'y', we have to use ' y_q '.

We have to keep traversing the tree in the direction of the results of the conditions at the nodes, until we reach a leaf node. Here the first lead node value we reach is 'c'. So this tree helps us in finding in which cuboid our point (x_q, y_q) lies in. So we can say that 'c' is the 1st nearest neighbor, but we are still not sure.

We shall calculate the distance between 'c' and the point (x_q, y_q) . Let the distance be 'd'. As 'c' is on the same side as (x_q, y_q) , we try to draw a hyperplane with centre at (x_q, y_q) and radius 'D'.

If we find any point inside this hypersphere, then we can say that point is the 1st nearest neighbor of (x_q, y_q) . We looked at only one side of the line $y=y_1$, as 'c' and (x_q, y_q) are on the same side. We shall now look at the other side of $y=y_1$, to find out the nearest neighbor for (x_q, y_q) . For that we need to backtrack till we reach $y \leq y_1$, and then as we have already traversed in the right subtree, we then have to traverse in the left subtree, according to the results of condition checking and reach the left node 'e'. Now we can say 'e' also could be our nearest neighbor.

Let us now calculate the distance between (x_q, y_q) and 'e' and let this distance be D' .

If $D < D'$, we can ignore 'e' and say that 'c' is the 1st nearest neighbor.

If $D' < D$, we can ignore 'c' and say that 'e' is the 1st nearest neighbor.

From the figure, it is clear that $D' < D$, so we can ignore 'c' and declare 'e' as the 1st nearest neighbor.

But again we have to draw a circle/sphere with the centre at (x_q, y_q) and radius D' . Now we'll check if the new circle intersects any other lines other than $y=y_1$. As we don't find any, we can confirm that 'e' is the 1st nearest neighbor.

Note: Number of Comparisons while searching for an element in KD-Tree is

Best Case: $O(\log(n))$

Worst Case: $O(n)$

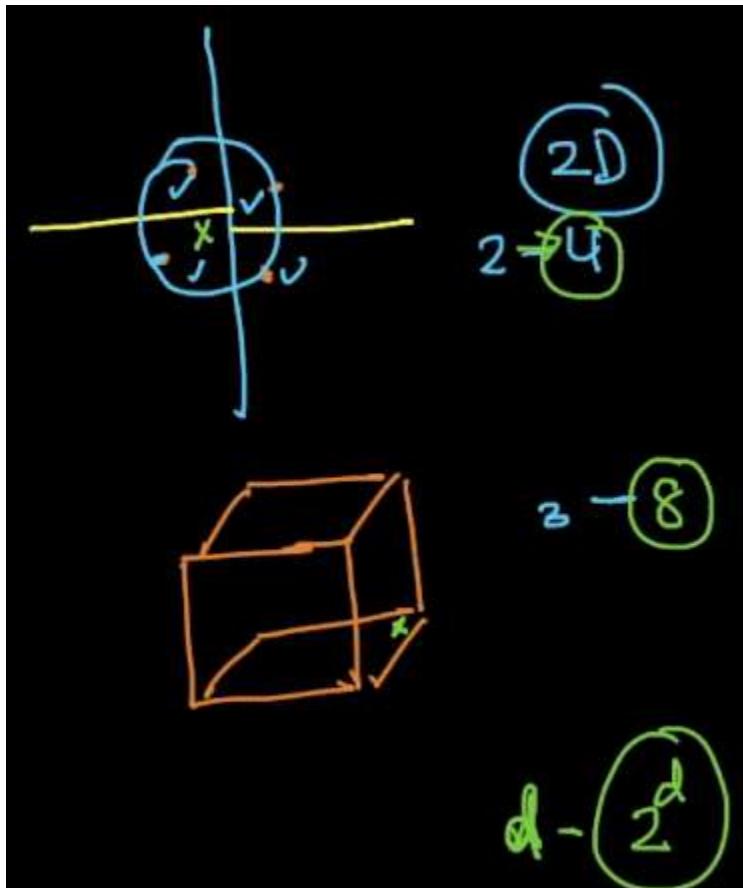
It means if there are no intersections of the planes, then the number of comparisons would be $O(\log(n))$. In the presence of intersections, then the number of comparisons would increase.

29.24 Limitations of KD-Tree

So far we have seen that if 'd' is small,

Best Case Time Complexity for KNN = $O(k * \log(n))$

Worst Case Time Complexity for KNN = $O(k * n)$



So far, we have seen when our data consists of 2-dimensions, we have to look up $2^2 = 4$ adjoining cells.

If the data is in 3-dimensional form, then the number of adjoining cells to look up = $2^3 = 8$.

So if the data is in d -dimensional form, then the number of adjoining cells to look up = 2^d

For example, if the number of dimensions = 20, then

Number of adjoining cells to look up = 2^{20}

Due to this, as the ' d ' value is not small, the time complexity is no more $O(\log(n))$.

The time complexity changes to $O(2^d * \log(n))$.

If $2^d = n$, then the time complexity is $O(n * \log(n))$.

As the number of dimensions increases, the number of adjoining cells also increases, thereby increasing the time complexity drastically. In such a case, KD-Tree is useless.

The Space Complexity of KD-Tree = $O(n)$

Even when the dimensionality is small, $O(\log(n))$ time complexity holds good only if our data is uniformly distributed in space. If the data is not uniformly distributed, the time complexity changes to $O(n)$. (same as that of brute force KNN).

Note: When compared to the other models, KNN is often used as a baseline model, when we are operating with small data. KD-Tree is not developed to find the nearest neighbor in Machine Learning, but was developed to find the nearest neighbors in the computer graphics.

29.25 Extensions

Refer to the wikipedia article link given below, as this video lecture is purely theory based, and also the content in it was taught from the below mentioned link only.

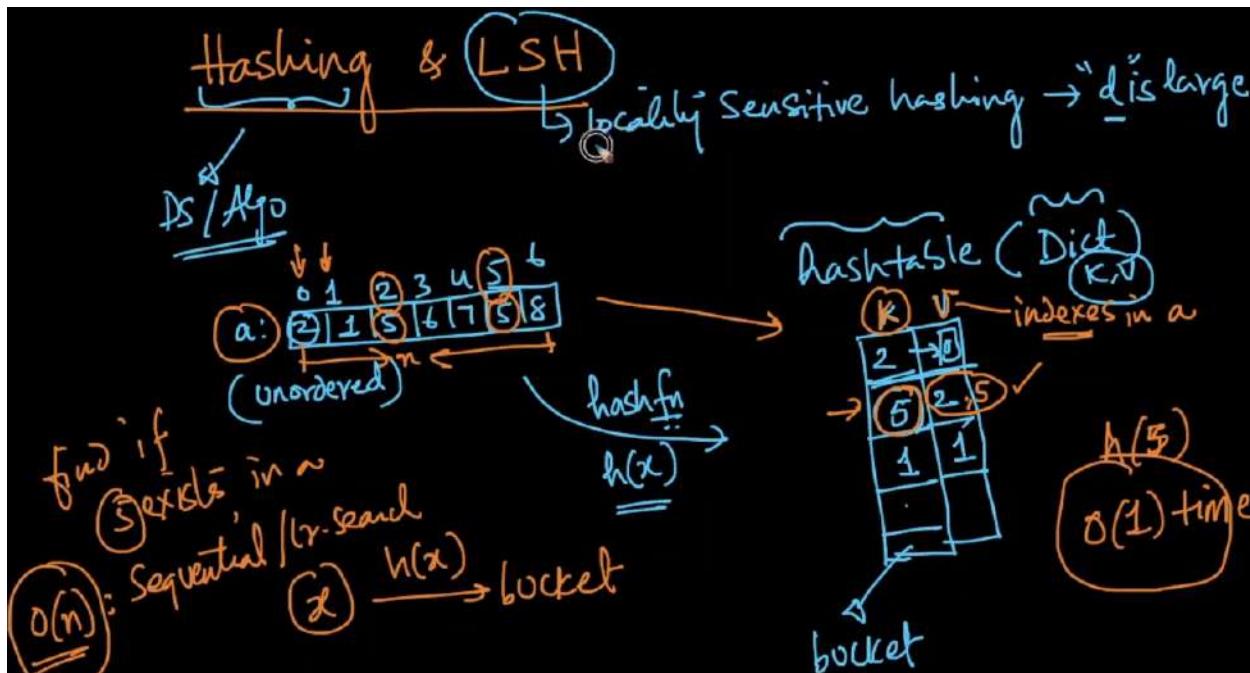
Wikipedia Link: https://en.wikipedia.org/wiki/K-d_tree#See_also

For any queries regarding this topic, please feel free to post them in the comments section below the video lecture.

29.26 Hashing vs LSH(Locality Sensitive Hashing)

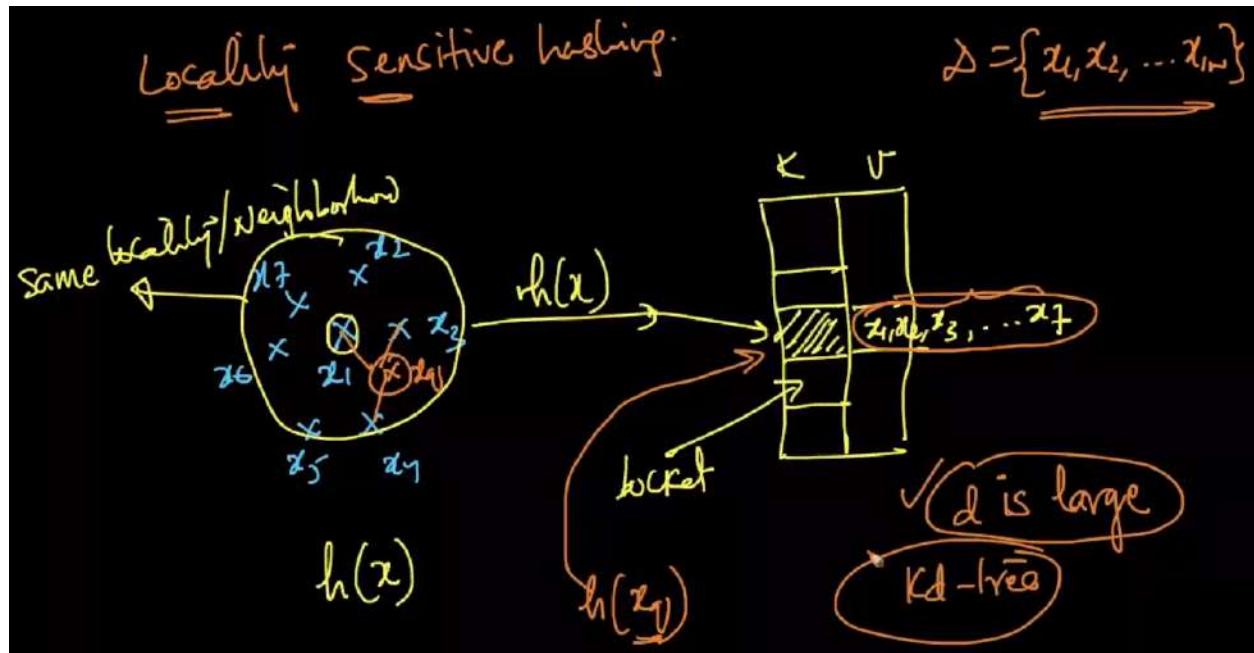
Locality Sensitive Hashing is a powerful technique when compared to KD-Tree and also works well when the dimensionality 'd' is large.

Before we look into LSH, we shall learn about Hashing. In general data structures, we have a data structure called Hashtable. The Hash function is the core concept on which a Hashtable works. One Hashtable will have only one hash function. There are no chances for one hash table to have multiple hash functions at a time. In python, Dictionary is the same as a hashtable.



The data in a hash table is stored in key-value pair format. If a value is given along with a key, then we can directly store that key-value pair in the hashtable. If a value is given without a key, then that value is passed through the hash function. The output of the hash function for a given point is the key, and each key is associated with a bucket in the hashtable. After we get the 'key' for a given value from the output of the hash function, then that particular value is stored in the bucket associated with the obtained key. So whenever we want to retrieve the value from the bucket, if we already know the 'key', then we can directly access it. Otherwise, we again have to pass the same value through the same hash function, we get the same key and then we can access it with that key. All the keys in a hashtable are unique, there is no duplication of keys.

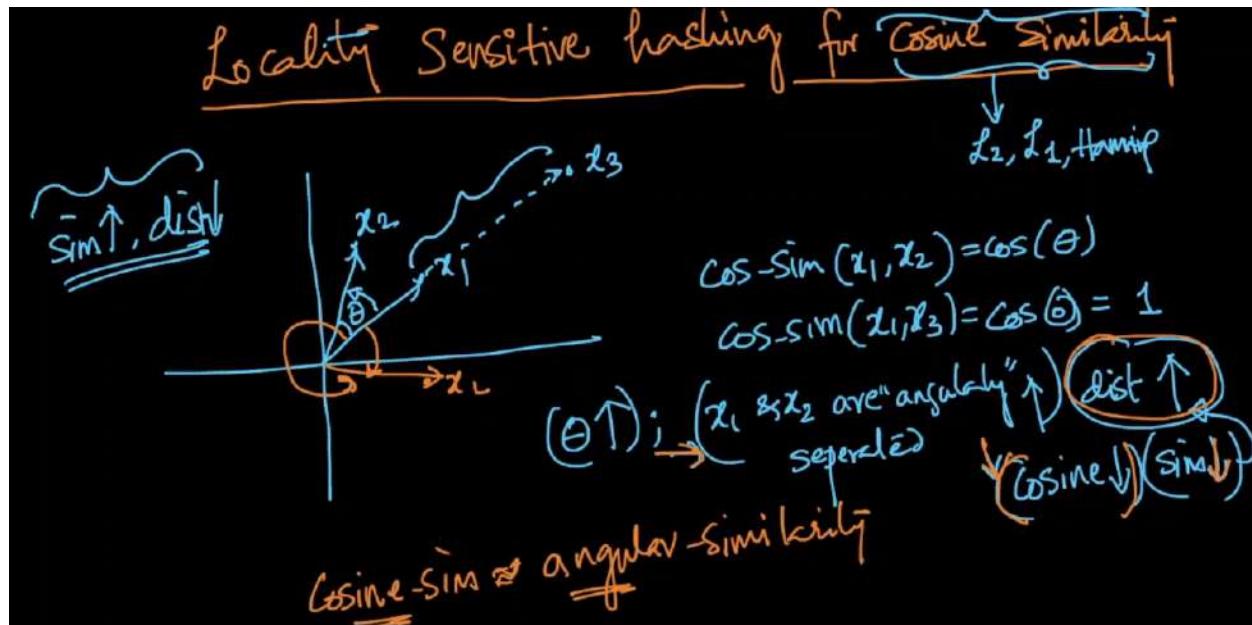
Let us now look at the functioning of Locality Sensitive Hashing. Let $h(x)$ be the hash function that is generated.



All the points here are in the neighborhood. For any given point ' x_i ', in such a way that all the points that are nearby go to the same bucket in the hashtable. For the point ' x_i ', first it has to go through the hash function, and a hash value is generated. This hash value will be the 'key' and ' x_i ' will be the value. These two are going to be stored in the key-value pair format.

So here, all the points that lie in a neighborhood are stored in the cell associated with the same bucket. So we can easily pull out the 'n' nearest neighbors using LSH, as all these values are stored in the same cell.

29.27 LSH for Cosine Similarity



We have 3 vectors ' x_1 ', ' x_2 ' and ' x_3 '. The vectors ' x_1 ' and ' x_3 ' are in the same direction and the angle between them is 0° . Whereas the vectors ' x_1 ' and ' x_2 ' are separated by an angle ' θ '. So we can say the vectors ' x_1 ' and ' x_2 ' are angularly separated. If the angle ' θ ' increases, the cosine similarity decreases, and the cosine distance increases.

$$\text{cosine_similarity}(x_1, x_2) = \cos\theta$$

$$\text{cosine_similarity}(x_1, x_3) = \cos(0^\circ) = 1$$

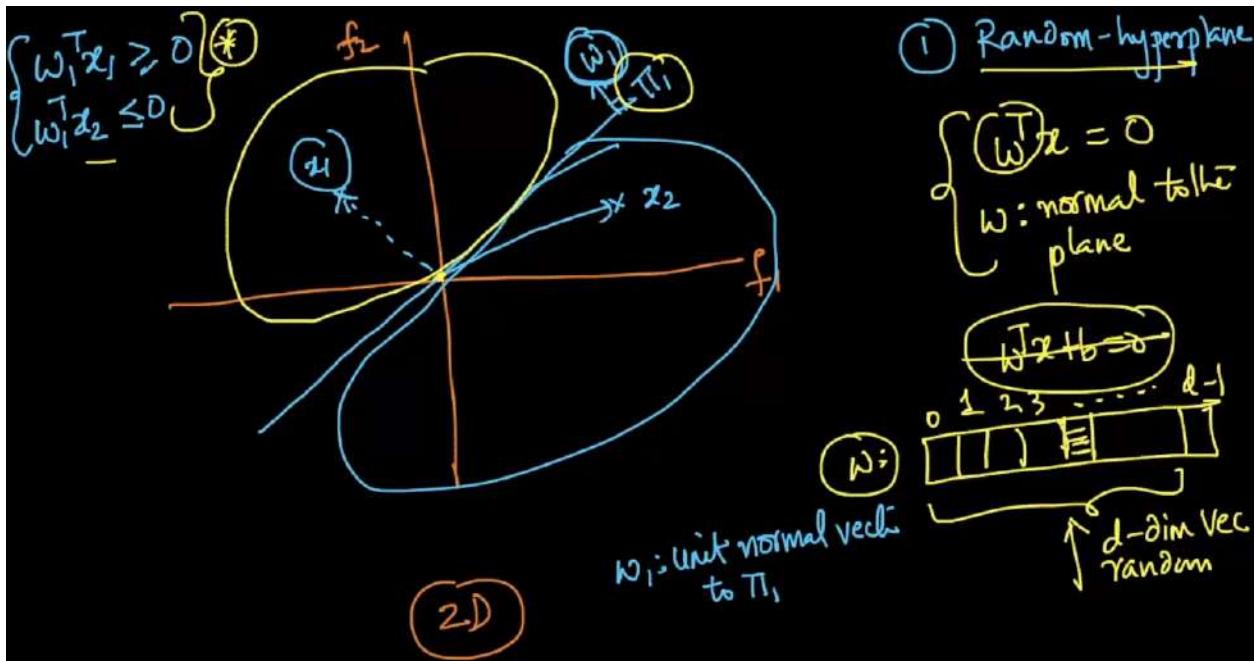
As ' θ ' increases, ' x_1 ' and ' x_2 ' are more angularly separated. Cosine Similarity is all about angular similarity, but not the geometrical distance. So in LSH, all the points that are more similar/close should go to the same bucket.

So if ' x_1 ' and ' x_2 ' are similar, then the hash function associated with these two points will become the key.

$$\text{Key} = h(x_1) = h(x_2)$$

LSH is a randomized algorithm. LSH says it always doesn't give the correct answer. But it always says it will give the correct answer with high probability. Every time LSH might not yield the nearest neighbors accurately, but everytime whatever results are obtained, are given with high probabilities.

Let us consider a 2-D plane and divide it into two parts using a line/plane/hyperplane(here it is a random plane), as shown below.

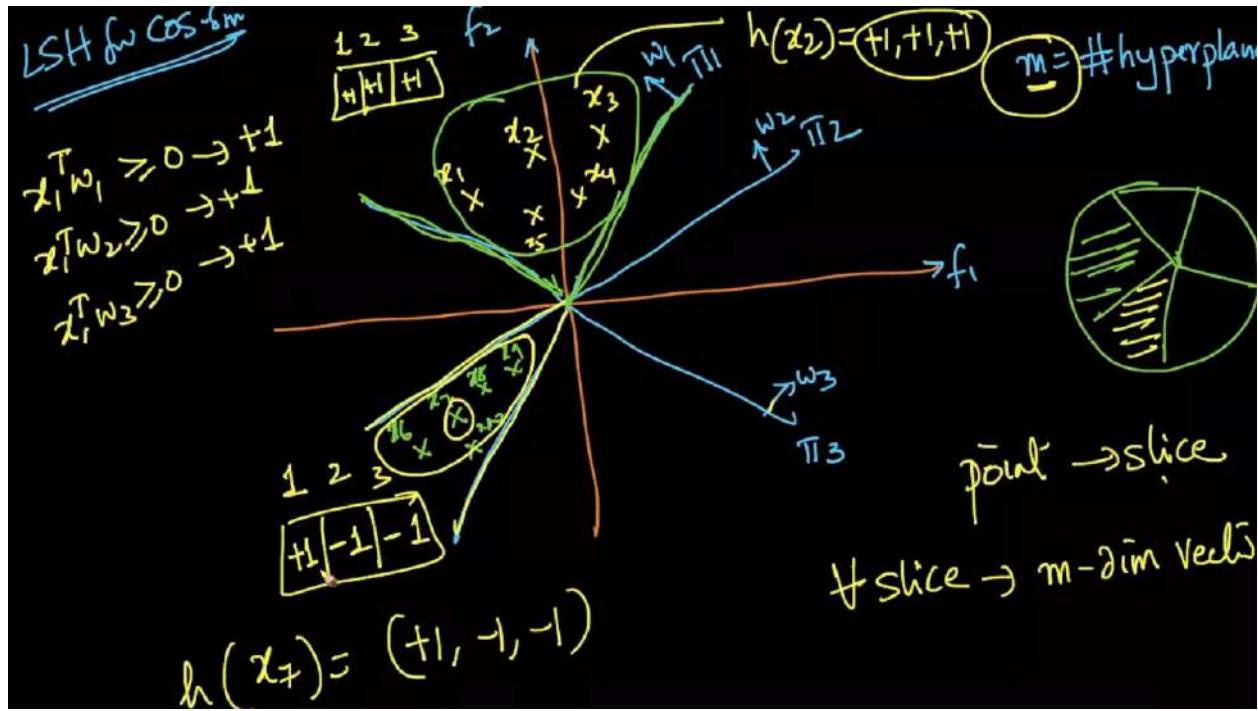


This plane is obtained by randomly generating the values for the unit normal vector ' w_1 '. Since ' x_1 ' is present on the same side as ' w_1 ', $w_1^T x_1 \geq 0$.

But the point ' x_2 ' is present on the opposite side of the vector ' w_1 '. So $w_1^T x_2 \leq 0$. Here ' w_1 ' is a d-dimensional random vector. If we are able to determine ' w_1 ', then we can easily find out the equation of the hyperplane ' π_1 '.

If a hyperplane passes through the origin, then the equation of the hyperplane is given as $w^T x = 0$. If the hyperplane doesn't pass through the origin, then the equation of the hyperplane is $w^T x + b = 0$. (Here ' w ' is a unit vector normal to the hyperplane and ' b ' is the intercept)

In ' w_1 ', every component is a random number sampled from Normal distribution. With the help of ' w_1 ', we can generate a random number hyperplane. Let us take a random hyperplane and understand how it works.



Let us consider 3 random hyperplanes and perform Locality Sensitive Hashing using Cosine Similarity.

If we multiply any point ' x_i ' in slice 1 with ' w_1 ', then $w_1^T \cdot x_i > 0$.

So we shall compute the hash key for the points in slice 1.

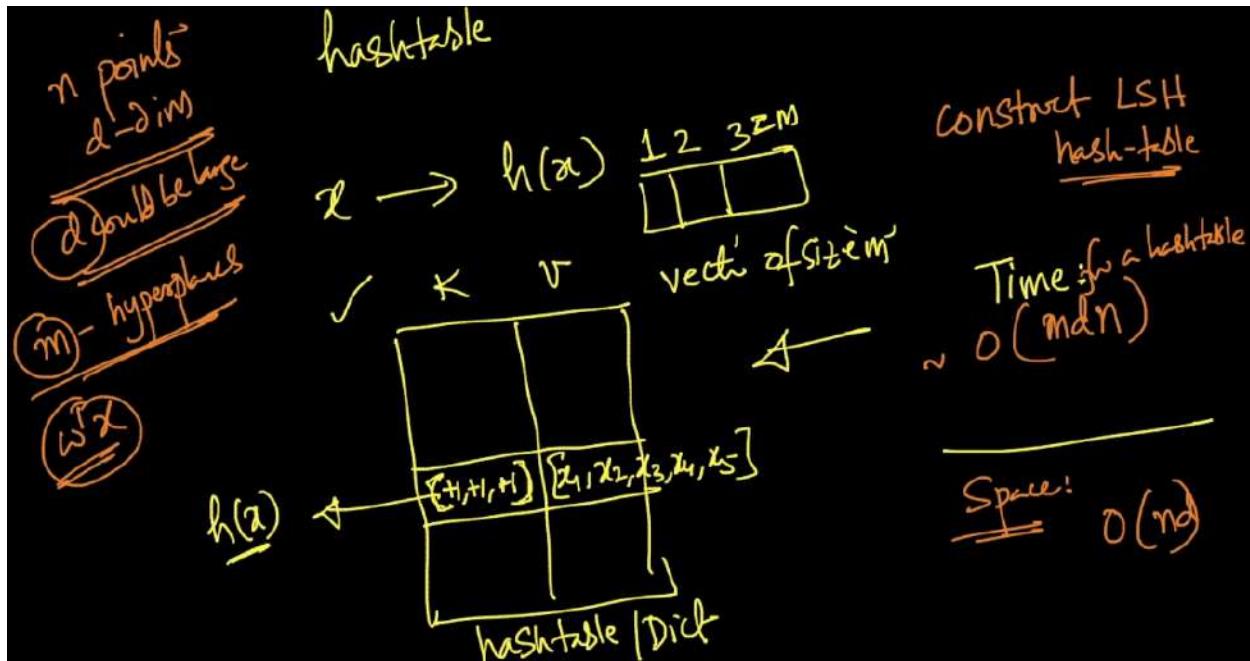
$\text{hash}(x_1) = \{+1, +1, +1\}$ (These 3 values denote the signs of $w_1^T \cdot x_1$, $w_2^T \cdot x_1$ and $w_3^T \cdot x_1$ respectively)

$\text{hash}(x_2) = \text{hash}(x_3) = \text{hash}(x_4) = \text{hash}(x_5) = \text{hash}(x_6) = \{+1, +1, +1\}$

As we got the keys using the hash function, we now have to construct the hash table.

Key (Hash Function Value)	Value
.	.
.	.
{+1, +1, +1}	[$x_1, x_2, x_3, x_4, x_5, x_6$]
.	.
.	.

Here we are taking the vectors as the key, and the points with these vectors, as the values. We have ' n ' data points and each data point is ' d ' dimensional.



So space complexity $\rightarrow O(nd)$ (To compute a hashtable)

Let us assume we have 'm' hyperplanes.

Time Complexity to compute one key-value pair = $O(m*d)$

Time Complexity to compute all the key-value pairs = $O(m*d*n)$

So in order to find out the nearest neighbors for the point ' x_q ', we need to compute the hash function for the point ' x_q ', and pull out the value associated with this hash function. In case if we want 'K' nearest neighbors, then after pulling out the list of the values from the hashtable, we have to compute the cosine similarity scores of ' x_q ' with all the points extracted from the hashtable. Those 'K' points with the top Cosine Similarity scores are chosen.

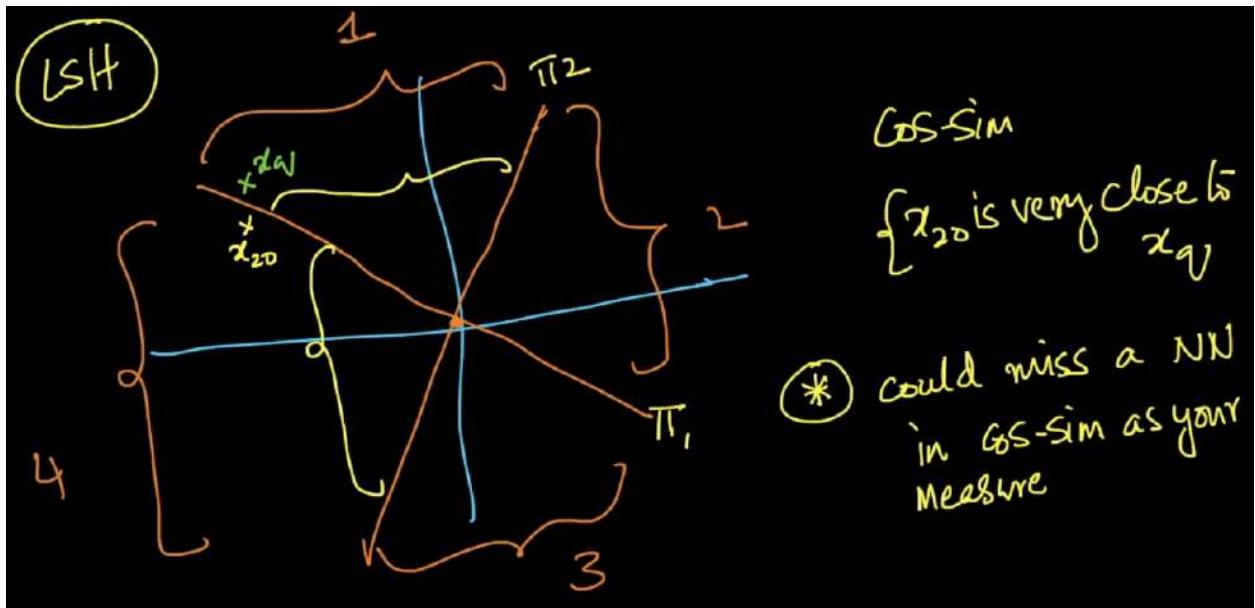
Time Complexity for querying a point = $O(m*d)$

In case, if there are n' elements in that bucket, then the total time complexity = $O(md + n'd)$

The ideal value for 'm' = $\log(n)$

So the time complexity = $O(d * \log(n))$

But here we come across a special case. So far, we have seen the points lying in the same bucket. But let us look at the figure below.



We have the point ' x_{20} ' in a different bucket/slice when compared to ' x_q '. So here even if ' x_{20} ' is a nearest neighbor of the query point ' x_q ', it will not be considered in the same bucket. So we would definitely miss this point, if we use cosine distance as the metric.

In order to get such points under the same bucket, we need to

- Take 'm' hyperplanes and create a hash function. Let it be $h_1(x_q)$.
- Take another 'm' hyperplanes and create another hash function. Let it be $h_2(x_q)$.

In this way, we have to keep doing it. At one hash function value, we get both these points in the same bucket.

So now, for every data point ' x_i ', we have to find out the hash value by passing it through all those hash functions, obtaining the nearest neighbors from every hash function, and performing a set union operation. Out of all the nearest neighbors, we have to pick our 'K' nearest neighbors, using the cosine distance.

So the total time complexity to query 'L' hash tables = $O(m*d*L)$

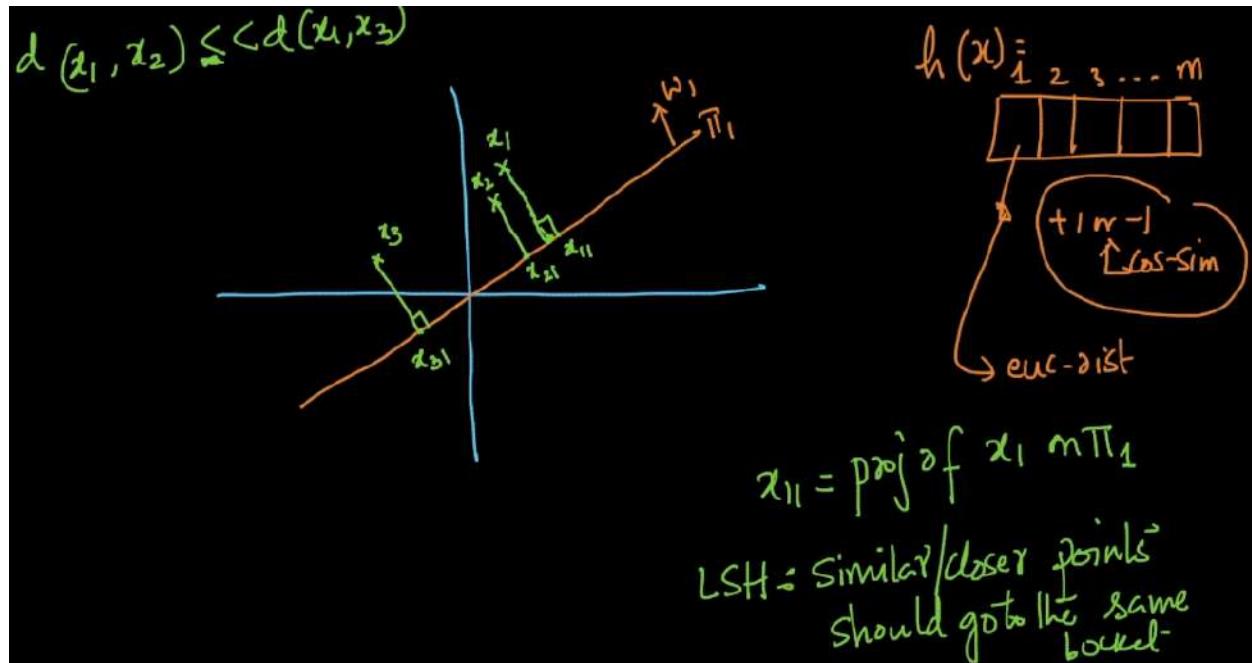
As the number of hyperplanes increases, the number of slices/buckets increase, and the number of points in each slice decreases.

As the number of hyperplanes decreases, the number of slices/buckets decreases, and the number of points in each slice increases, thereby the time complexity becomes worse.

29.28 LSH for Euclidean Distance

So far we have seen that if a point ' x_i ' belongs to the region same as ' w_1 ', then we use +1, and if it is present on the opposite side, we use -1 in the hash key.

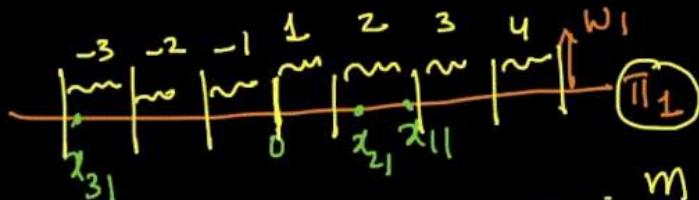
Now we shall draw the perpendiculars of all the points onto the hyperplane ' π ', as shown below.



We now have to divide this hyperplane into the regions on the basis of the projections of the points on it. We shall give labels to each region as shown below.

$$h(x_{31}) = \boxed{1 \ 2 \ \dots \ m}$$

breaking plane into
pieces/regions
↑ 'a' regions



$$a=8$$

$$h(x) = \boxed{1 \ 2 \ \dots \ m}$$

$$h(x_{21}) = \boxed{2 \ 1 \ \dots \ m}$$

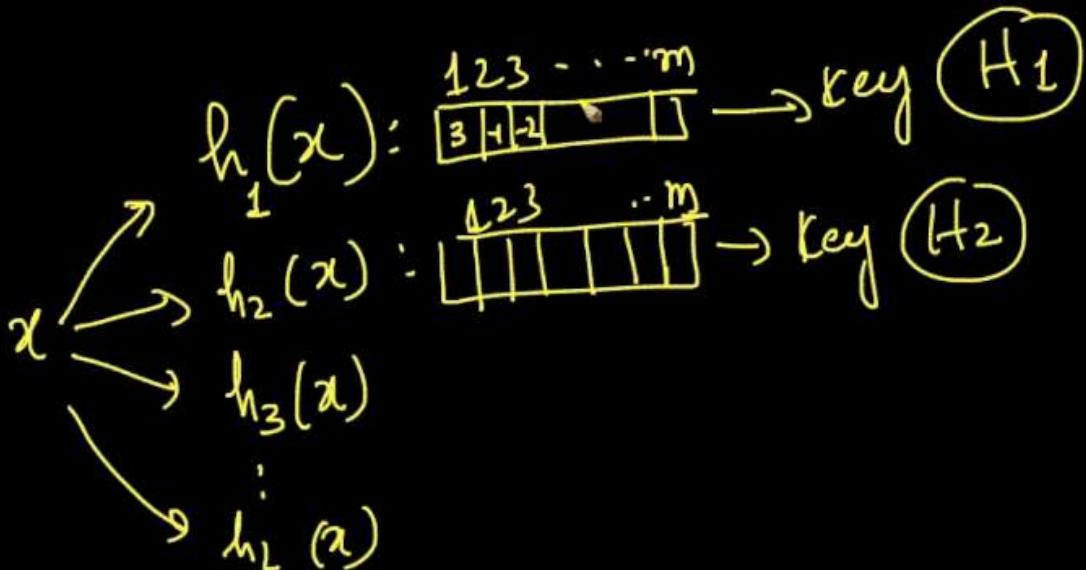
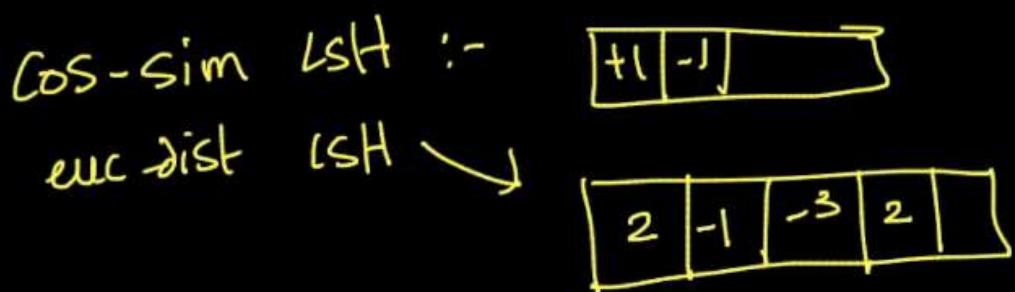
Now we have to create the hash key. If there are 'm' hyperplanes, then the hash key would be a 'm' dimensional vector.

The hash key for the point 'x₁₁' consists of '2' for the component '1'. (It is because this point is having a projection onto the hyperplane 'π₁'. So the 1st component of the 'm' dimensional vector would be filled with the value. This value here would be the region number in which this point is present). We can see the point 'x₁₁' lies in region '2' of the hyperplane 'π₁', so the first component of the hash key would be '2'.

When it comes to the hash key of 'x₂₁', even here this point lies in the 2nd region of the hyperplane 'π₁'. So the first component of the hash key should be '2'.

If we look at the hash key of the point 'x₃₁', as this point is present in the region '-3', we fill the first component of the hash key with '-3'.

This is the difference between LSH for Cosine Similarity and LSH for Cosine Distance. The hash values in LSH for Cosine Similarity contain +1 or -1, whereas in LSH for euclidean distance, the value can be anything(the region number associated with the hyperplane).



Let us assume, here is LSH for euclidean distance, we have 'L' hashtables, then for every point 'x', we have to generate 'L' hash keys, as shown above. Each of these hash keys is 'm' dimensional.

$h_1(x) \rightarrow$ Hash key of the point 'x' in Hashtable 1

$h_2(x) \rightarrow$ Hash key of the point 'x' in Hashtable 2

\vdots

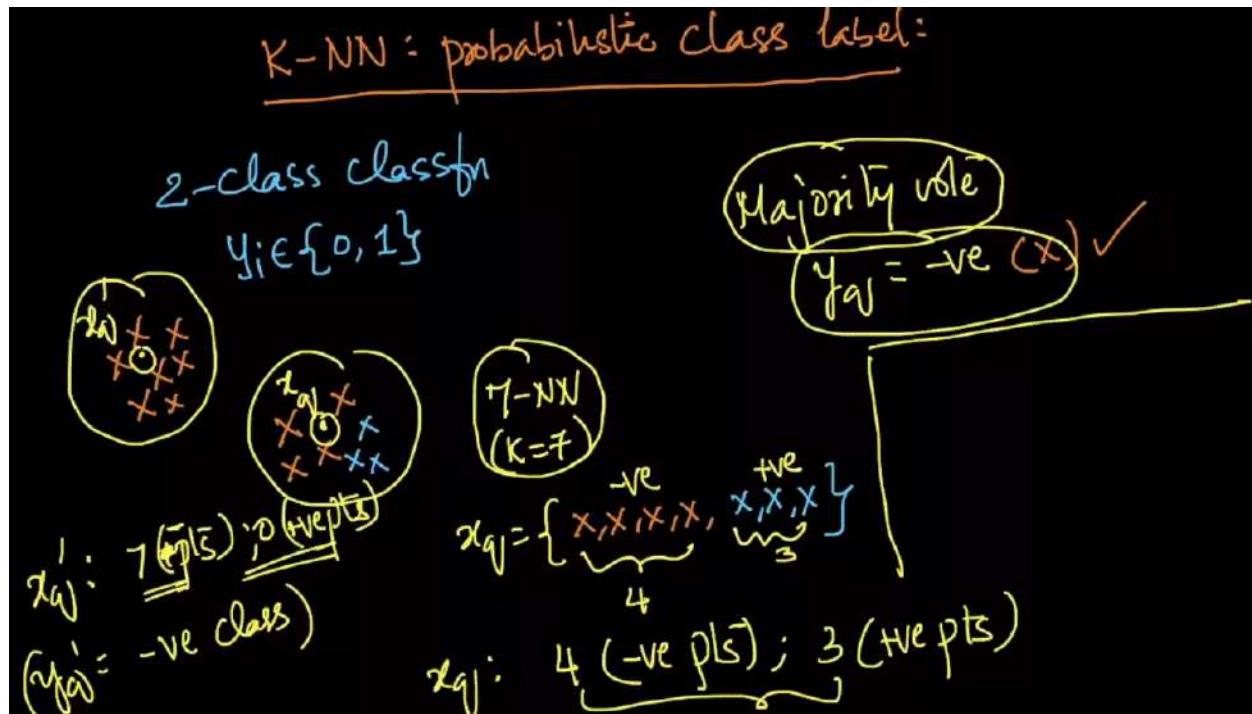
$h_L(x) \rightarrow$ Hash key of the point 'x' in Hashtable L

The points that have the same region number tend to be closer. If 2 points have the same region value of a hyperplane ' π_1 ', then the distance between them will be closer.

But again as these hyperplanes are formed randomly, they can be formed in any direction, and due to this sometimes the points may fall in different regions.

So LSH is not perfect all the time. It is a probabilistic/randomized algorithm and is still used extensively.

29.29 Probabilistic Class Label



Let us consider a binary classification problem. (ie., $y_i \in \{0, 1\}$). Let us assume we have the points as shown above, and are working on a 7-NN problem. ($K=7$)

So the 7 nearest neighbors of ' x_q ' are

$$x_q = \{-ve, -ve, -ve, -ve, +ve, +ve, +ve\}$$

Here if we go with the majority vote count, we get $y_q = -ve$.

Let us assume we have another point ' x_q' ' and its 7 nearest neighbors are

$$x_q' = \{-ve, -ve, -ve, -ve, -ve, -ve, -ve\}$$

Here if we go with the majority vote count, we get $y_{q'} = -ve$.

$$P(y_q = -ve) = (\text{Number of '}-ve'\text{ neighbors of } x_q) / (\text{Total number of points in the neighborhood of } x_q) = 4/7$$

$$P(y_q = +ve) = (\text{Number of '}+ve'\text{ neighbors of } x_q) / (\text{Total number of points in the neighborhood of } x_q) = 3/7$$

$$P(y_{q'} = -ve) = (\text{Number of '}-ve'\text{ neighbors of } x_{q'}) / (\text{Total number of points in the neighborhood of } x_{q'}) = 7/7 = 1$$

$$P(y_{q'} = +ve) = (\text{Number of '}+ve'\text{ neighbors of } x_{q'}) / (\text{Total number of points in the neighborhood of } x_{q'}) = 0/7 = 0$$

$\boxed{1\text{-NN}}$ $x_{qj} : \begin{cases} 4 \text{-ve pts} ; 3 \text{+ve pts} \\ 2_{qj}^1 : -7 \text{ (+ve pts)} \end{cases}$; $\begin{cases} \underline{y_{qj}} = -ve \\ \underline{y'_{qj}} = -ve \end{cases} \left\{ \begin{array}{l} \text{majority rule} \\ \text{more certain} \end{array} \right.$

Quantify this uncertainty

$$\begin{cases} P(\underline{y_{qj}} = -ve) = \frac{4}{7} = \frac{\# \text{-ve pts}}{\text{Total \# pts}} & \left\{ \begin{array}{l} P(y_{qj} = +ve) = \frac{3}{7} \\ P(y'_{qj} = +ve) = \frac{0}{7} \end{array} \right. \\ P(\underline{y'_{qj}} = -ve) = \frac{7}{7} = 100\% \end{cases}$$

\checkmark Probabilistic class label
 $\underline{x_{qj}} \rightarrow y_{qj} \left\{ \begin{array}{l} P(y_{qj} = +ve) \\ P(y_{qj} = -ve) \end{array} \right\} \rightarrow \text{certain}$
 $\underline{x_{qj}^1} \rightarrow P(y_{qj}^1 = -ve) = 1$

Probabilistic class labels are mostly used in classification tasks. Instead of giving deterministic class labels, we are giving probabilistic class labels. Sometimes giving probabilistic class labels is better than giving deterministic class labels.

Probabilistic result is a mathematical way to show how certain we are in predicting the class labels.

Note: We are not giving any notes for the video lectures 29.30 and 29.31 as they both are of only the code discussions.

You can download the ipython notebooks from the link given below.

<https://drive.google.com/drive/folders/1tMYRWzbrSMxQ7aQ5mc8Qf4190gPSt5f>

For any queries, please feel free to post them in the comments section below the video lecture.

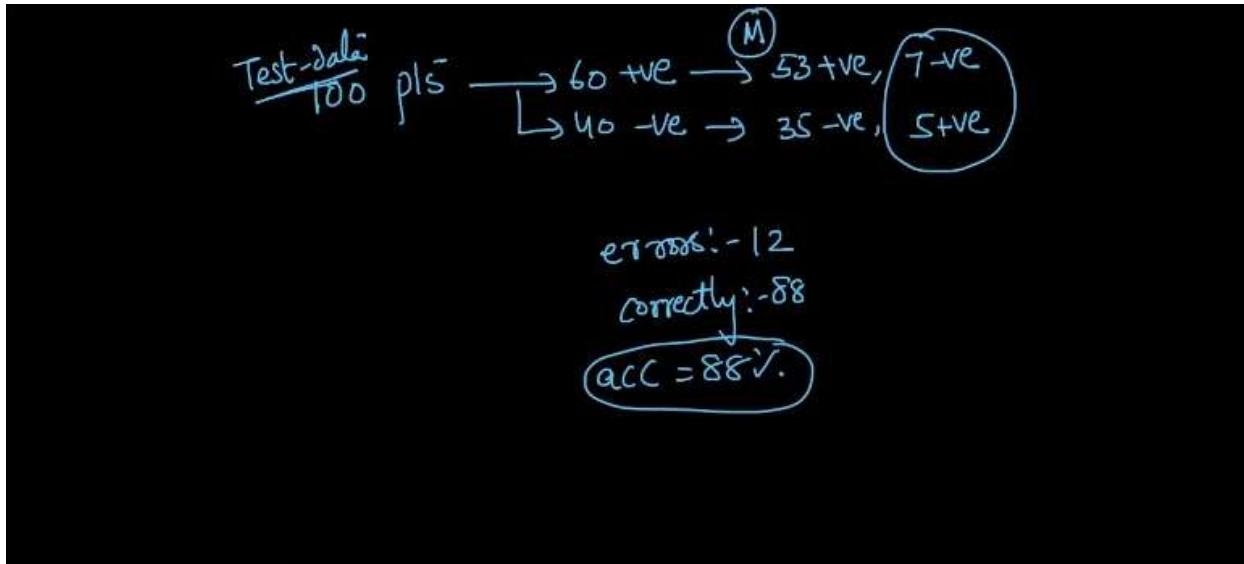
In this chapter we will look at various metrics by which we can assess the performance of our models. We will also look at the advantages and disadvantages of them. We will mainly look at metrics for classification and regression models.

30.1 Accuracy

The image shows handwritten notes about accuracy as a performance metric for classification and regression models. At the top, 'Performance prediction of models' is written above 'Accuracy metric'. Below this, 'Classification; regression (KNN)' is mentioned. A formula for Accuracy is given: $\checkmark \text{ Accuracy} = \frac{\# \text{ correctly classified points}}{\text{Total } \# \text{ points in Test}}$. To the right, a scale from 0 to 1 is shown, where 0 is labeled '(bad)' and 1 is labeled 'better'. A bracket under the formula points to the text 'easy to understand'.

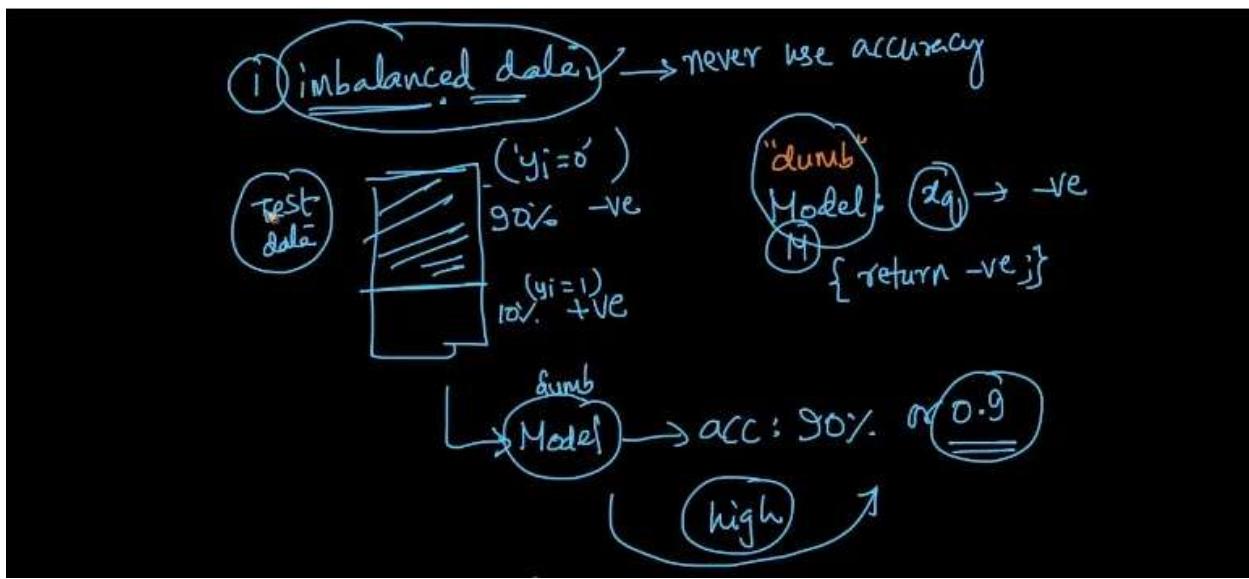
Timestamp 2:07

Accuracy can be defined as the ratio of the correctly classified and the total number of points, as shown in the image. It is generally used for classification. It ranges from 0 to 1, where 0 means that the model hasn't predicted any of the data points correctly and 1 means all the data points were classified correctly. It is very easily interpreted.



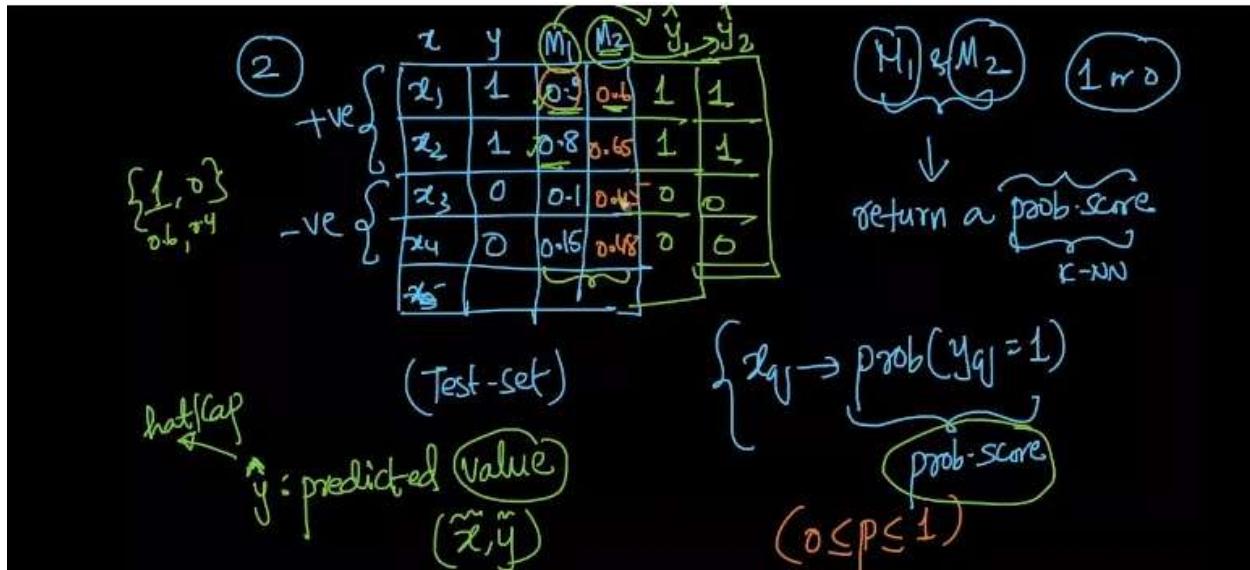
Timestamp 3:07

In the above image we can see how accuracy is calculated. For our final prediction we mainly compute our metrics on test data.



Timestamp 6:00

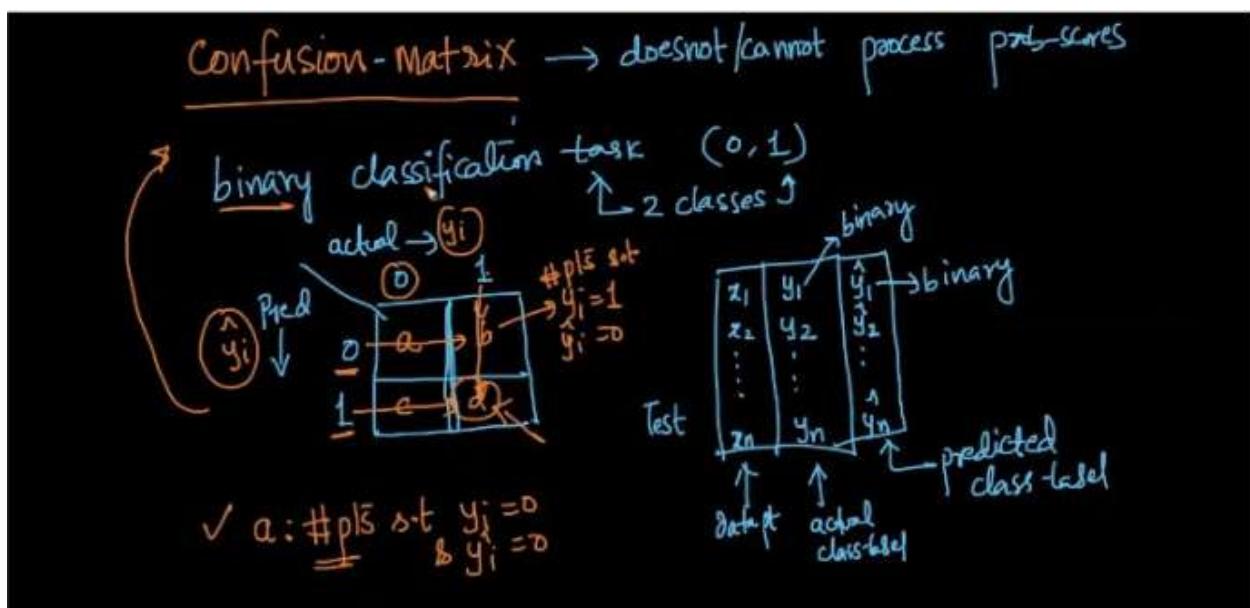
One of the disadvantages of using accuracy is that it performs very poorly for an imbalanced dataset. As you can see from the image, if my dataset has one majority class, say negative(say 90%) and we make a model such that we predict the majority class(negative) for every data point. This model will result in having a very high accuracy giving us a wrong conclusion that my model has performed very well. Even though my model hasn't learnt anything.



Timestamp 12:08

Another disadvantage of using accuracy is that it doesn't deal with probability scores directly. Consider KNN which can output probability scores. Let's say we have two models M1, M2. We have trained these models on x 's and our true classes are y 's. As shown in the image. Now we can see the probabilities predicted by these models. Let's say these values represent the probability that the predictions are 1. So according to that we have filled the predictions(\hat{y}) for both the models. Now we can calculate the accuracy for both these models using our \hat{y} 's. We will see that both have equal accuracy even though M1 gave much better predictions.

30.2 Confusion Matrix, TPR, FPR, FNR, TNR

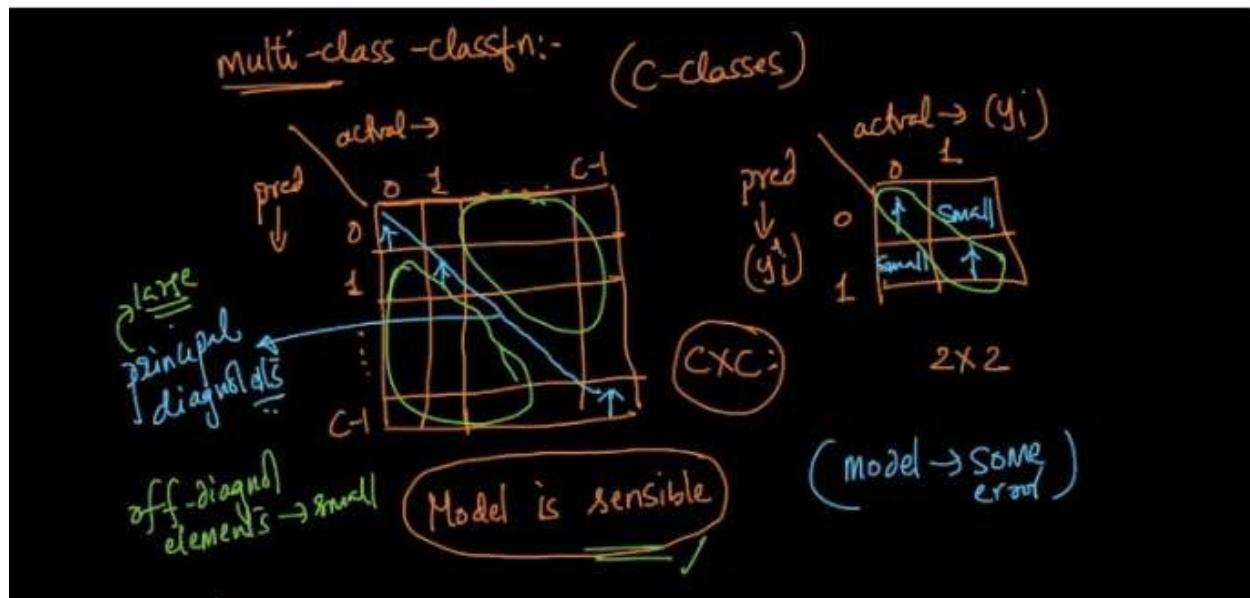


Timestamp 4:29

Here we will look at the confusion matrix, which is a very useful way of summarizing the predictions of the model. It works on actual class predictions and not probabilities. Consider a binary classification problem where y_i 's(ground truth) $\in \{0, 1\}$. Now for a binary classification problem, the combinations can have 4 possibilities. These 4 possibilities are represented using the cells of the 2-D matrix as shown in the image.

1. a: #pts such that $y(\text{actual}) = 0$ and $\hat{y}(\text{predicted}) = 0$.
2. b: #pts such that $y(\text{actual}) = 1$ and $\hat{y}(\text{predicted}) = 0$.
3. c: #pts such that $y(\text{actual}) = 0$ and $\hat{y}(\text{predicted}) = 1$.
4. d: #pts such that $y(\text{actual}) = 1$ and $\hat{y}(\text{predicted}) = 1$.

Also keep in mind that the headers of the cell are important like in which direction you have actual values and in which direction you have predicted values. In some textbooks these can change.



Timestamp 7:41

Similarly the confusion matrix can be extended for c classes. These can be represented using a square 2-D matrix of c dimensions. Note in the matrix the diagonal elements represent the correct predictions whereas off diagonal elements represent the wrong predictions. A good or sensible model will have large diagonal elements and lower off diagonal elements.

	pred	0	1
0	TN	FN	
1	FP	TP	
	N	P	

✓

TP, FP, FN, TN

What is the predicted label
are u correct

Q N: # neg
P: # pos
 $n = N + P$

Timestamp 11:05

We can name our cells as shown below:

True Negative(TN) - Ground truth: 0 and Predicted: 0

False Negative(FN) - Ground truth: 1 and Predicted: 0

False Positive(FP) - Ground truth: 0 and Predicted: 1

True Positive(TP) - Ground truth: 1 and Predicted: 1

One easy to trick to remember this is to break it into two parts, where the first part says whether we are correct and the second part says our prediction

For e.g. - In TP, T - represents whether our prediction was correct, in this case it was correct i.e. True and P represents what is our prediction, in this case positive.

If we add FN + TP , we get our total positive points, similarly TN + FP gives us the total negative points that were present in the dataset.

If we add both positive and negative data points we get our total no of data points in the dataset.

$$\left. \begin{array}{l} \checkmark TPR = \frac{TP}{P} \\ TNR = \frac{TN}{N} \\ FPR = \frac{FP}{N} \\ FNR = \frac{FN}{P} \end{array} \right\}$$

Hand \rightarrow

0	1
0	TN
1	FP

(TP)

N P

$\underline{N+P=n}$

Timestamp 13:03

Here we will define a few more terms.

True Positive Rate (TPR) = TP/P

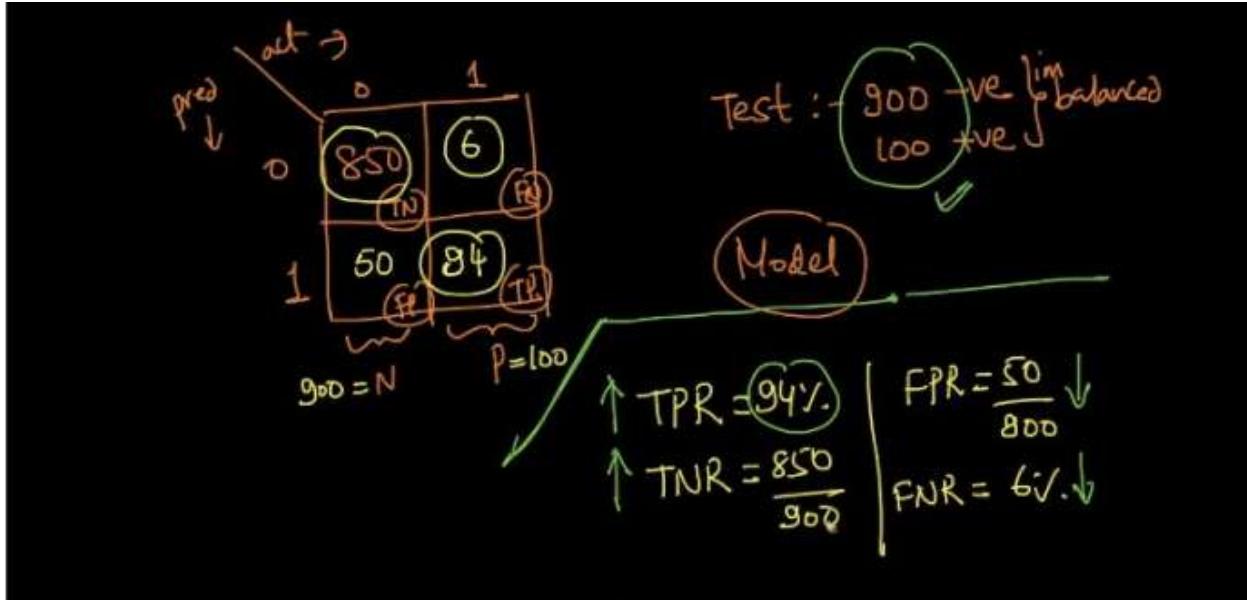
True Negative Rate(TNR) = TN/N

False Positive Rate(FPR) = FP/N

False Negative Rate(FNR) = FN/P

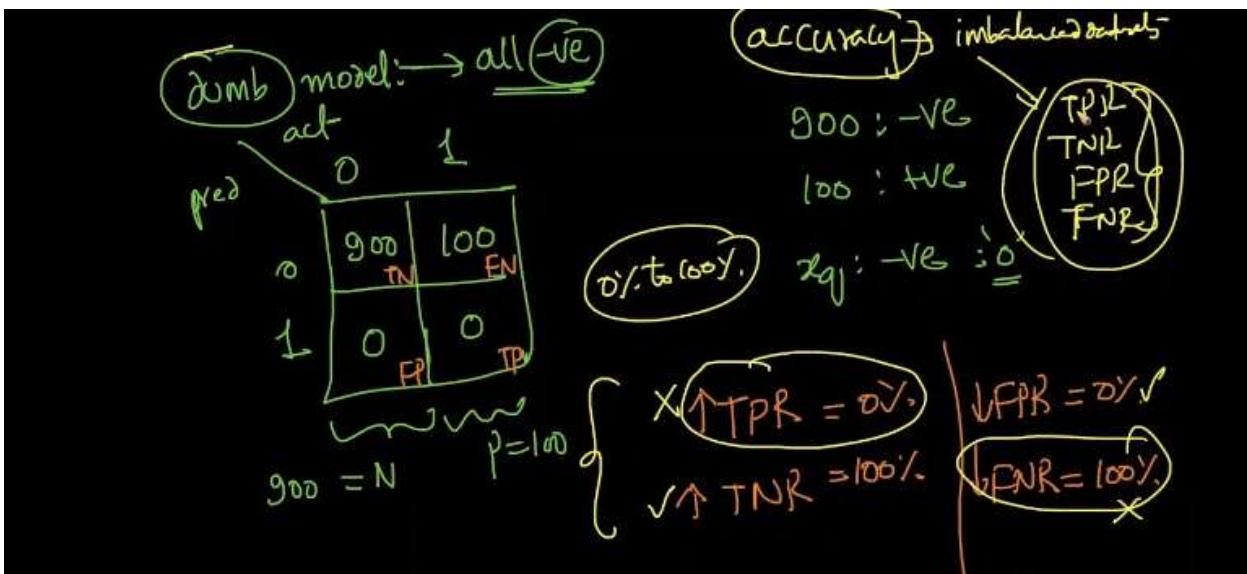
Here P = #positive points, N = #negative points.

We can relate it to the diagram in the image above to help remember these formulas better.



Timestamp 16:19

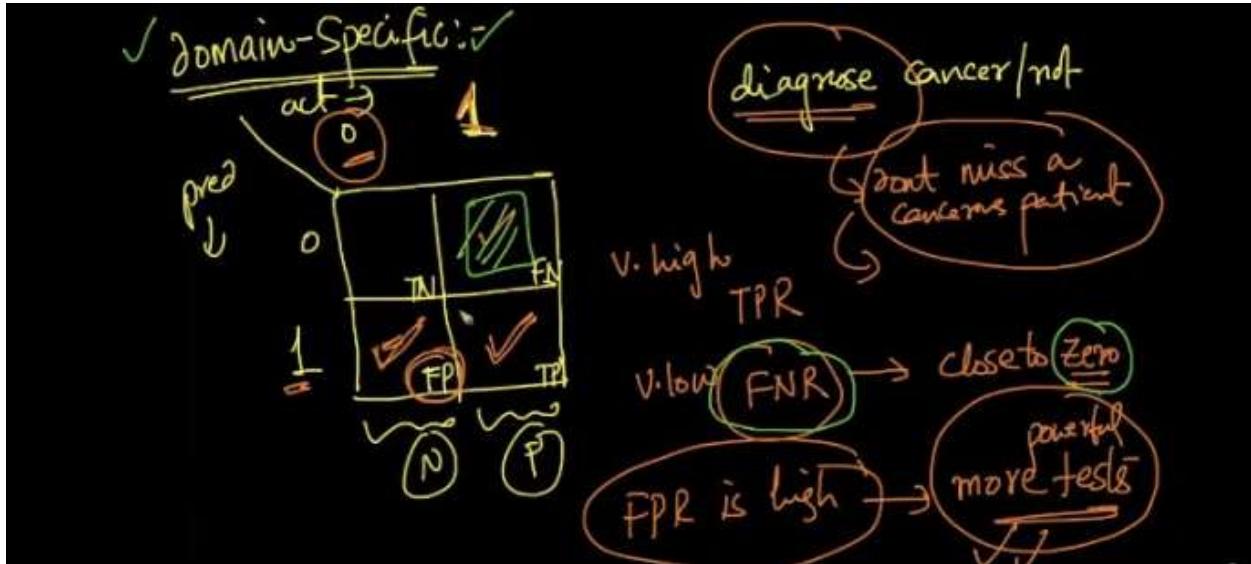
In the above image it can be seen that we have taken an imbalanced dataset. Suppose our model has made predictions for all the data and based on that we have filled the cells of confusion matrix and calculated various metrics. It can be clearly observed that the diagonal values are high whereas off diagonal values are low. This means that our model is good. Good models have high TPR, TNR whereas FPR, FNR should be low.



Timestamp 20:02

Here we have again taken an imbalanced dataset and used dumb model. Dumb model predicts the majority class without learning anything. Now, we have filled the confusion matrix according

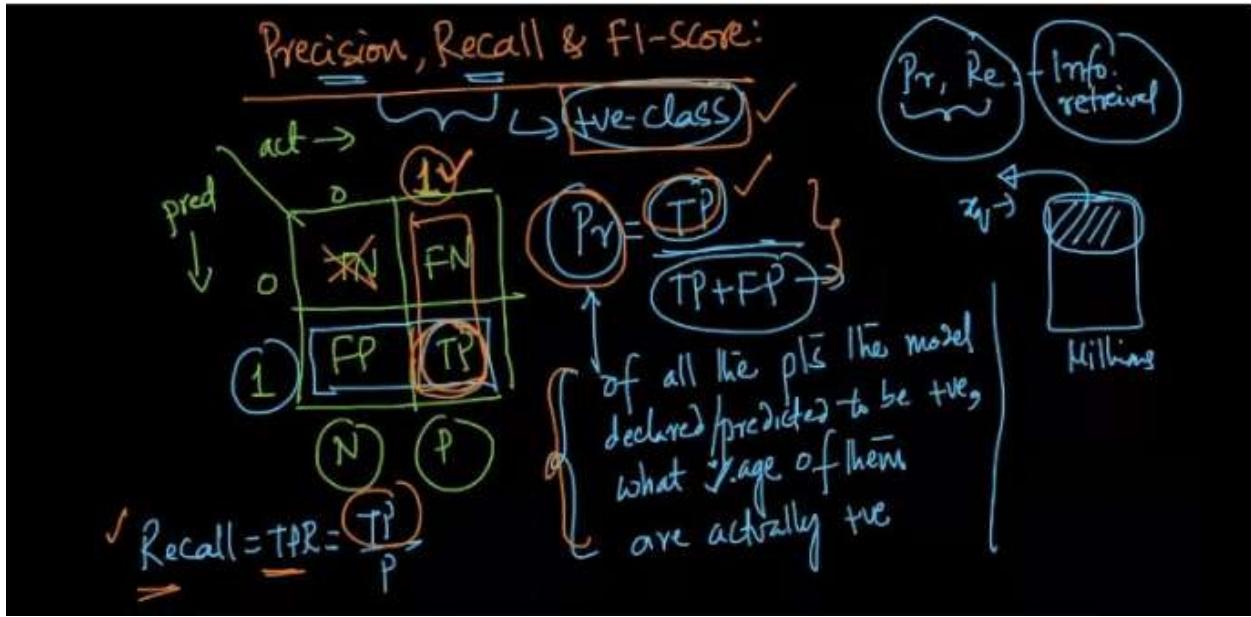
to this. We can now observe that TNR is high, but TPR is low. Also FNR is high, indicating that my model is not good and we need to improve it. This type of observation was not possible in case of accuracy. But on the contrary now we have to deal with 4 numbers instead of one like in accuracy.



Timestamp 24:50

Now the question comes whether we should use a confusion matrix and deal with 4 numbers or use just accuracy and deal with just one number. The answer is it is domain specific. It depends on the problem at hand. Consider a case of cancer diagnosis, where we have built a model that predicts whether a person has cancer. Now in this case we want to make sure that if a patient has cancer, our model detects it i.e. high TPR and also low FNR so that we don't miss a person that has cancer. It is very important to have low FNR. This wouldn't have been possible if we would have used accuracy.

30.3 Precision, Recall & F1-Score



Timestamp 5:17

Here we will talk about precision and recall. Both these metrics have come from information retrieval, which focuses on retrieving certain documents from millions of documents based on certain conditions. Both these metrics focus on positive class. So for any problem we can use these metrics by changing our requirement to positive class.

$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$. Intuitively it means that out of all the points our model predicted to be positive, what percentage of them are actually positive. It's a number between [0,1], with large values meaning high precision.

$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$. Intuitively it means that out of all positive points that were present in the dataset, how many of them were predicted successfully. It's a number between [0,1], with large values meaning high recall.

$$\begin{aligned}
 & \text{Pr, Re} \rightarrow \text{one measure} \\
 & \begin{array}{cc} \text{Pr} \uparrow & \text{Re} \uparrow \\ (0-1) & (0-1) \end{array} \\
 \rightarrow & \text{F1-Score} = \left(2 * \frac{\text{Pr} * \text{Re}}{\text{Pr} + \text{Re}} \right)
 \end{aligned}$$

Timestamp 6:23

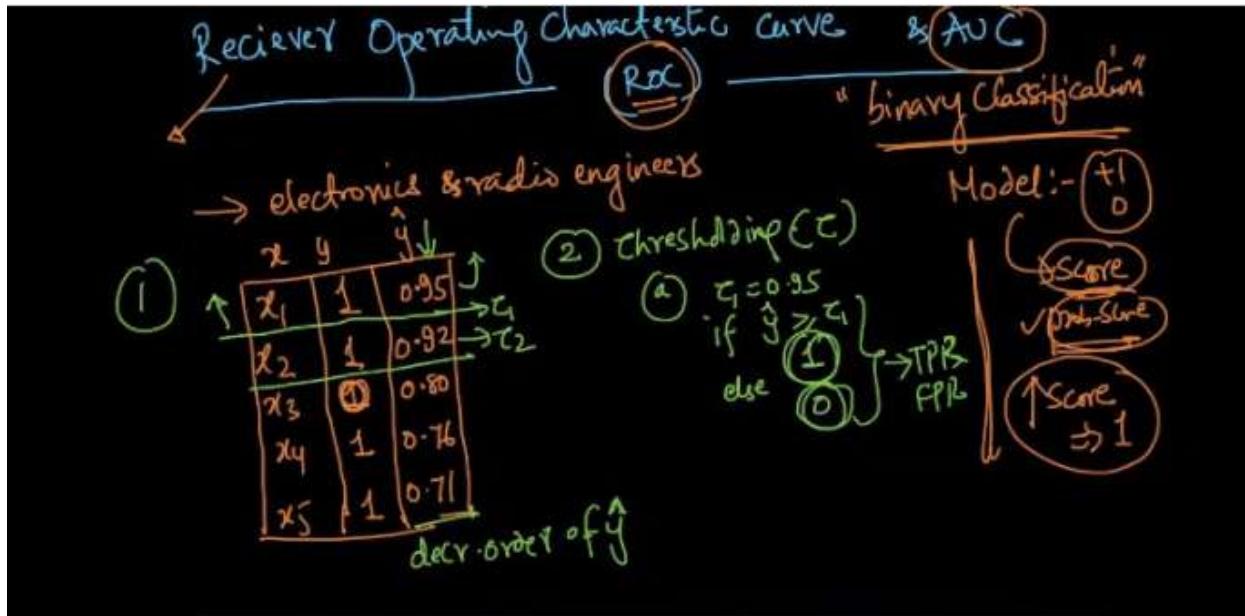
Both precision and recall can be combined to give us one score. Generally we want both precision and recall to be high. We combine them using something called F1-Score which is the harmonic mean of precision and recall.

$$\begin{aligned}
 & \text{Pr, Re} \rightarrow \text{one measure} \rightarrow \text{interpretable} \\
 & \begin{array}{cc} \text{Pr} \uparrow & \text{Re} \uparrow \\ (0-1) & (0-1) \end{array} \\
 \rightarrow & \text{F1-Score} = \left(2 * \frac{\text{Pr} * \text{Re}}{\text{Pr} + \text{Re}} \right) \\
 & \text{Simple English}
 \end{aligned}$$

Timestamp 9:07

F1-Score's are less interpretable than precision and accuracy. It is widely used in kaggle competitions. It ranges between 0 & 1, where a higher F1 score means a better model.

30.4 Receiver Operating Characteristic Curve (ROC) curve and AUC



Timestamp 6:15

Receiver Operating Characteristic Curve or ROC is a metric which is mainly used for binary classification problems. It was developed during the world war mainly by electronic and radio engineers.

Suppose we have a model that outputs probability scores as output, remember ROC can work on any scores. Now in order to calculate the ROC we need to follow the below steps.

1. In this step we sort the data table based on the decreasing value of predictions (\hat{y}). As it can be seen in the above image marked by 1.
2. Thresholding - In this step we first select the topmost value say τ_1 , in our example we have $\tau_1 = 0.95$, and use this as the threshold i.e. any value greater than or equal to this value will be assigned a class label of 1 and the other ones will be labeled as 0.

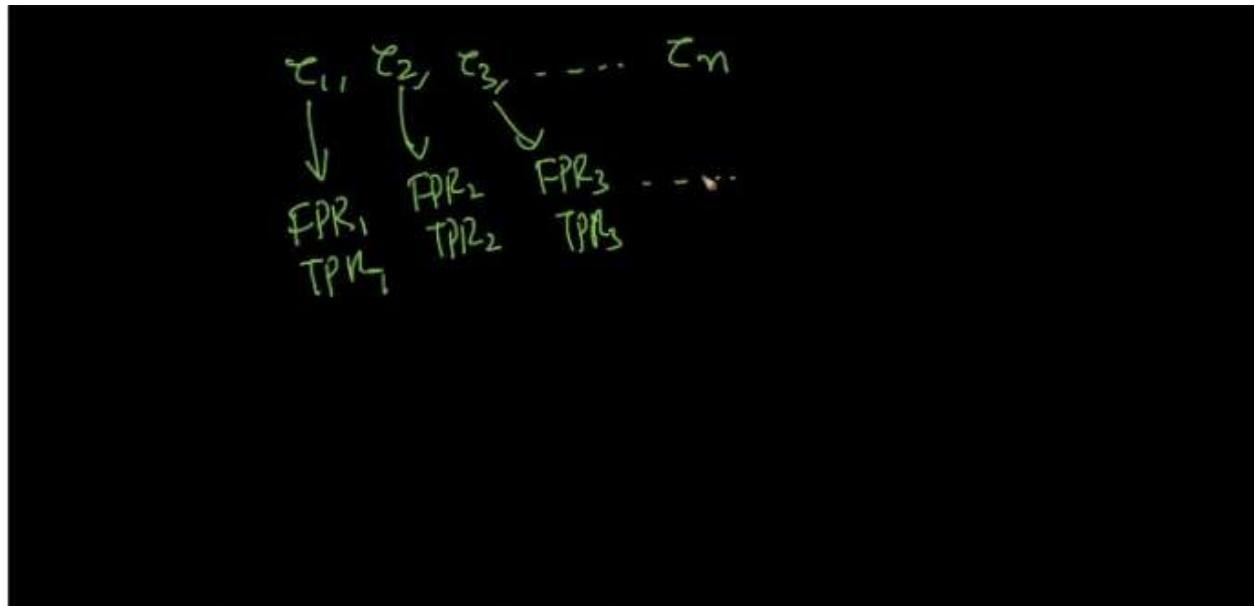
x	y	\hat{y}	$\hat{y}_{\tau_1=0.95}$	$\hat{y}_{\tau_2=0.92}$
x_1	1	0.95	1	1
x_2	1	0.92	0	0
x_3	0	0.80	0	0
x_4	1	0.76	0	0
x_5	1	0.71	0	0

$\hat{y}_{\tau_1=0.95} \rightarrow \text{TPR}, \text{FPR} (\tau_1)$

$\hat{y}_{\tau_2=0.92} \rightarrow \text{TPR}, \text{FPR} (\tau_2)$

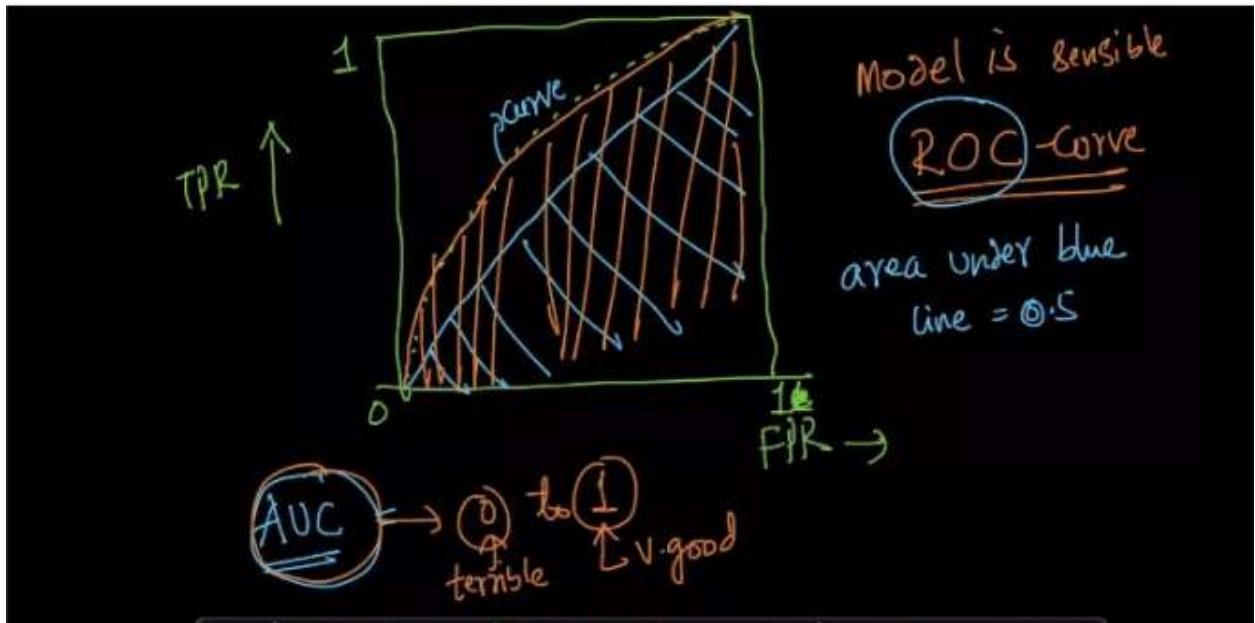
Timestamp 6:40

Similarly these thresholding process should be followed for other \hat{y} 's based on their ordering. For each threshold we will get a different list of predictions as shown in the above image. Now, for each of these predictions w.r.t different thresholds, we calculate their TPR and FPR.



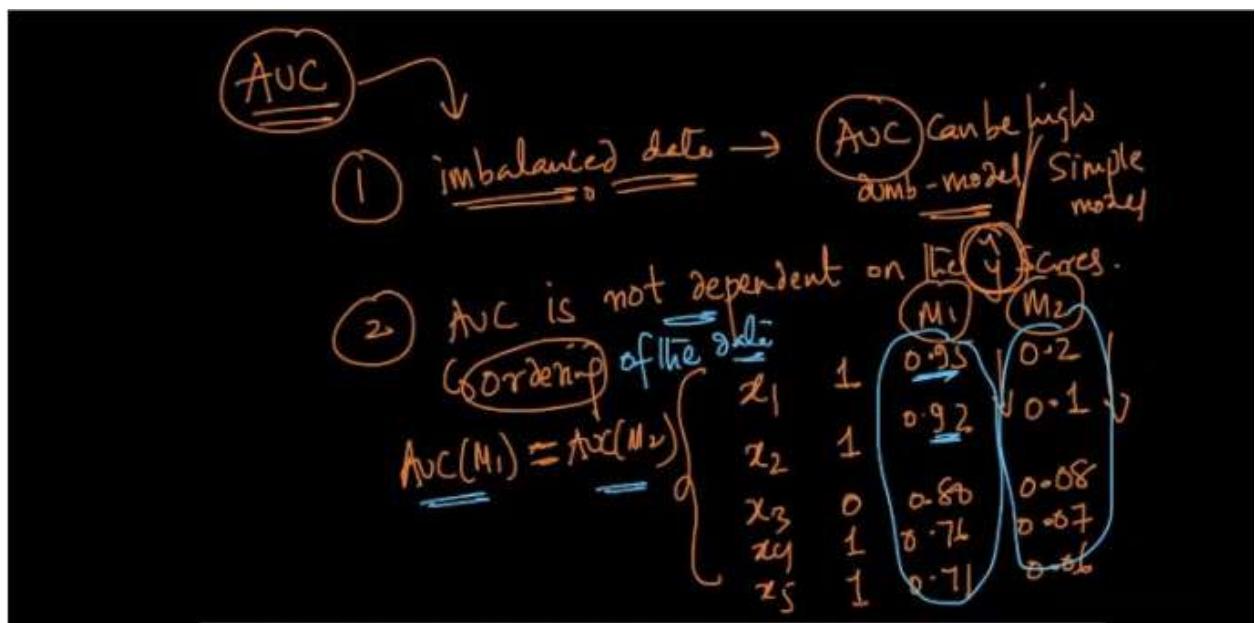
Timestamp 7:27

Now, if our dataset had n rows, then we will have n different thresholds, and for each of these thresholds we will have different TPR and FPR.



Timestamp 10:15

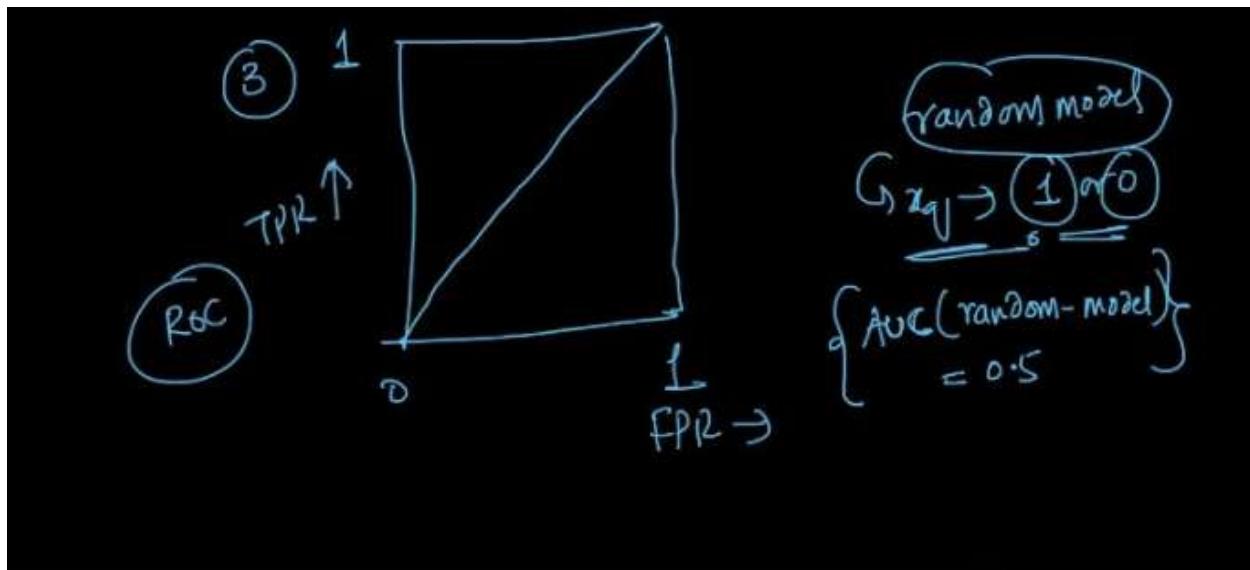
Now we have a list of FPR's and their corresponding TPR's. Also keep in mind that their values lie between 0 & 1. We can now plot these values in a graph with FPR on x-axis and TPR on y-axis. After plotting the points we will get a curve shown in red line, it is called the ROC Curve. The total area under the curve can be 1, as the maximum value is 1 on both axes. The blue line represents the line under which half of the area lies. Now, the area under the ROC Curve is called AUC(Area under the curve). Its value can range from 0 to 1, with higher value meaning better model.



Timestamp 13:30

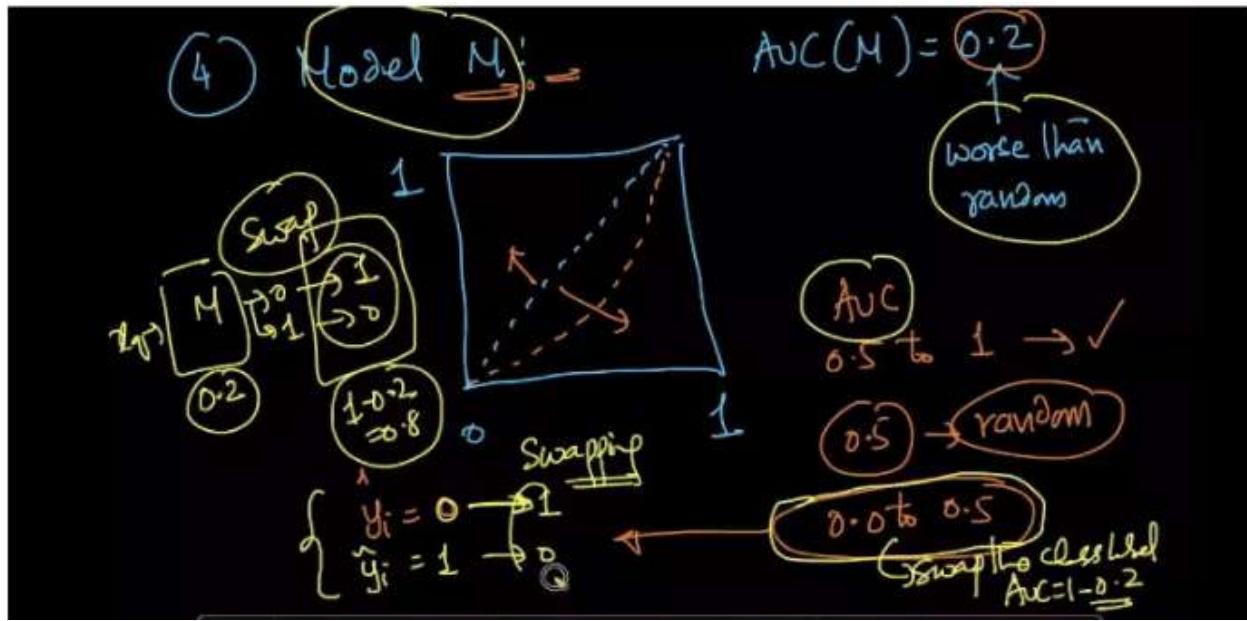
Now let's look at several properties of AUC.

1. AUC is impacted by imbalanced dataset i.e. even a dumb or simple model can have high AUC.
2. AUC is not dependent on the actual predictions but rather on the ordering of the predictions. Consider two models M1 and M2 making different predictions as shown in the image above. If we calculate the AUC for both these models, it will be the same. But it can be clearly observed that M1 did much better at predicting than M2.



Timestamp 14:53

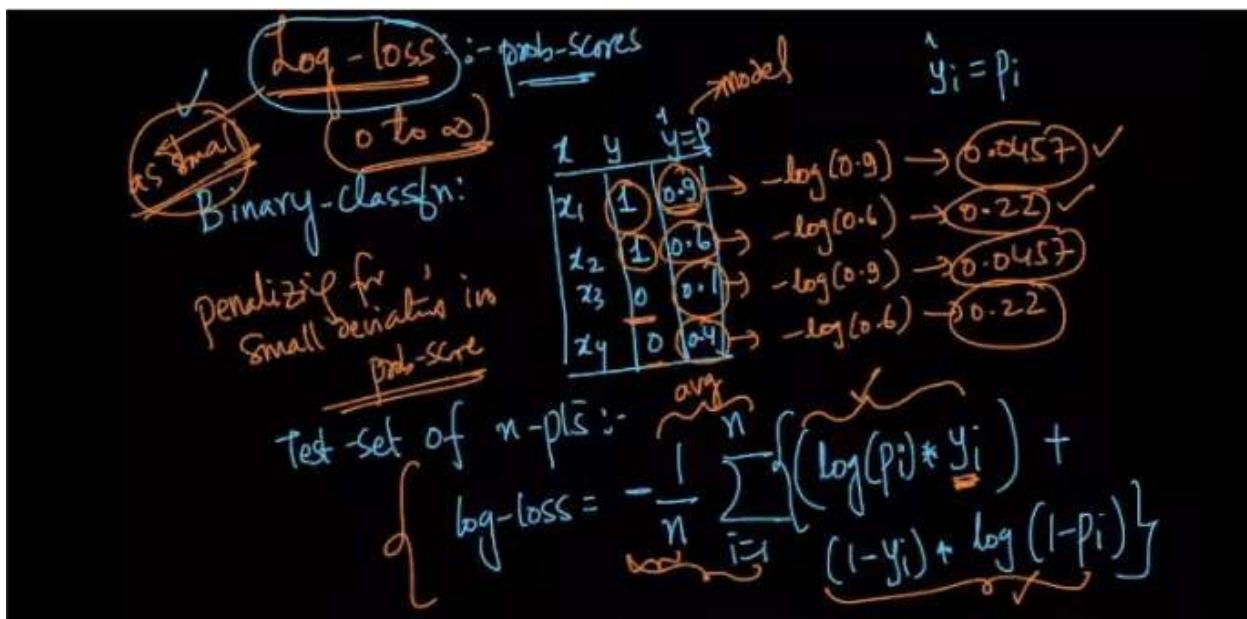
3. AUC for a random model will be 0.5. A random model is such that for each prediction it will select class 0 or class 1 with equal probability, same as tossing a coin.



Timestamp 18:18

4. Suppose we have our model M, that has an AUC of 0.2 as shown in the image by the red line, which is worse than a random model. In that case we can deploy a trick. Suppose my model predicts class 0, then we switch our predictions and say that the prediction is 1 and vice-versa. We can just switch our predictions. Doing this will change our AUC score from 0.2 to $1 - 0.2 = 0.8$, which is a pretty good auc score.

30.5 Log Loss



Timestamp 6:15

Log loss is loss metric which lies between $[0, \infty)$. The lower the loss value the better the model.
 Log loss deals with direct probability scores unlike other metrics that were defined till now.
 Consider a set of n points, log loss for these data set can be defined as:-

$$\text{Log-loss} = -\frac{1}{n} \sum_{i=1}^n \{ (\log(p_i) * y_i) + (\log(1 - p_i) * (1 - y_i)) \}$$

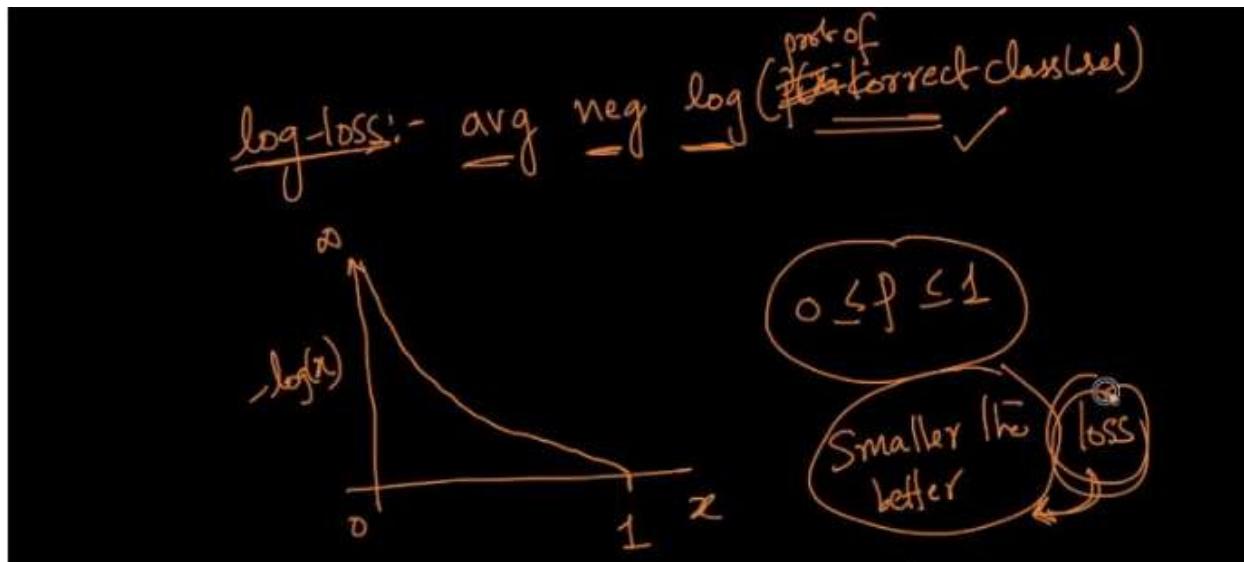
Here p_i = Probability that the class belongs to 1,

y_i = Actual class label, n = total no of datapoints

As it can be seen from the above formula it is a negative average of all points. This formula can be thought of in two parts.

Suppose our class label $y_i = 1$, then the second part vanishes as $1 - y_i = 0$, so the formula that remains is $-\log(p_i) * y_i$. Now here since $y_i = 1$, we have $-\log(p_i)$, which is the probability that y_i belongs to class 1. Now greater the value of p_i smaller will be the negative log value, hence smaller the loss. This can be verified for the class 0 case also.

In the above image we have calculated various log values and shown how it varies w.r.t change in probability values for different classes.



Timestamp 8:16

In the above image we can see a graph which shows the relationship between the probability scores and the negative log of those scores. As the scores rise the negative log value decreases. We always strive for smaller loss values.

Multi-class log-loss: $[0, 1] \times [0, \infty)$

$$\mathcal{L}_q \rightarrow \begin{array}{|c|c|c|} \hline & p_2 & \dots & p_c \\ \hline \end{array}$$

$$\left\{ -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^c y_{ij} \log(p_{ij}) \right\}$$

prob that $x_i \in$ class j

$$= 1 \text{ if } x_i \in \text{class } j$$

$$0 \text{ or } 0$$

best case = 0

log-loss(M) $\leq 1 \cdot D$

$D = 10^D$

Timestamp 11:02

The formula for log loss can be easily extended to a multiclass setting as shown in the above image. Here c represents the number of classes that we have. We can verify this formula by setting $c = 2$, and observing that it reduces to binary log loss.

One of the disadvantages of log loss is that it is very difficult to interpret.

30.6 R-Squared/Coefficient of determination

(R²) or Coefficient of determination

$y_i \in \mathbb{R}$

classfn-measures

regression

test :- x_i, y_i, \hat{y}_i

$i: 1 \text{ to } n$

\hat{y}_i model output

$e_i = \hat{y}_i - y_i$

error

y_i actual value

model output

Timestamp 2:02

Here we will talk about the metrics for regression problems. Our y_i 's $\in \mathbb{R}$ i.e. set of real numbers unlike classification tasks. Now, we can define error as the difference between the ground truth or actual value and predicted value as shown in the above image. We represent it as e_i .

$$\text{total SS} \quad \left\{ \begin{aligned} \text{sum of squares} \\ SS_{\text{total}} &= \sum_{i=1}^n (y_i - \bar{y})^2 \end{aligned} \right.$$

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

$$\left\{ \begin{aligned} \bar{y} &: \text{avg value of } y_i \text{'s} \\ &\text{in train data} \end{aligned} \right.$$

$$\text{regression} \rightarrow \text{simplest model} \rightarrow \text{avg-model}$$

$$x_q \rightarrow \text{mean}(y_i) = \bar{y}$$

$$\text{avg} = 152\text{cm}$$

$$\left\{ \begin{aligned} \text{Train data} & \quad \text{predict height (EIR)} \\ & \quad \text{features: } w, c, hc, \dots \end{aligned} \right.$$

$$x_q \rightarrow \bar{y} \rightarrow 152\text{cm}$$

$$x_q' \rightarrow \bar{y} \rightarrow 152\text{cm}$$

Timestamp 6:32

Here we will define a term SS_{total} , total sum of squares,

$$SS_{\text{total}} = \sum_{i=1}^n (y_i - \bar{y})^2$$

y_i = ground truth

\bar{y} = average value of all the y_i 's in the train dataset.

This term represents the sum of the square of the errors between y_i and \bar{y} . Here for prediction we have predicted \bar{y} for every datapoint in train data, which is the mean of y_i 's. This is a very basic model where we haven't learnt anything useful. This is also called the average-model or simple-mean-model.

$$SS_{res} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n e_i^2$$

residue = $e_i = y_i - \hat{y}_i$

y_i actual \hat{y}_i predicted

Timestamp 8:53

Here we will define a term SS_{res} , residual sum of squares,

$$SS_{res} = \sum_{i=1}^n (y_i - \hat{y})^2$$

y_i = ground truth

\hat{y} = predicted value based on a trained model.

This term represents the residual sum of the square of the errors between y_i and \hat{y} . Here we have trained a model on the dataset and based on that we are giving predictions.

$$R^2 = \left(1 - \frac{SS_{res}}{SS_{tot}} \right)$$

Case 3
Model: SS_{res}
↳ same as a simple-mean-model
best-value

Case 1: $SS_{res} = 0 \Leftrightarrow e_i = 0 \rightarrow R^2 = 1$

Case 2: $SS_{res} < SS_{tot}; R^2 \in 0 \text{ to } 1$

Case 3: $SS_{res} = SS_{tot}; R^2 = 1 - 1 = 0 \rightarrow$ Model is same as S-M-model

Timestamp 11:30

Now we can define our coefficient of determination (R^2),

$$= (1 - SS_{res} / SS_{total})$$

Let's understand this based on cases:

Case 1: $SS_{res} = 0$, If SS_{res} is 0, this means that our predicted model has made no errors and predicted everything correctly.

In that case $R^2 = (1 - 0/SS_{total}) = 1$, this is the best value i.e. $R^2 = 1$

Case 2: $SS_{res} < SS_{total}$; R^2 lies between 0 and 1.

Case 3: $SS_{res} = SS_{total}$; This means that our model is the same as the simple mean model. Here $R^2 = 0$.

Casey: $SS_{res} > SS_{tot}$

$$R^2 = 1 - (q > 1) = \text{(-ve)}$$

{ Model is worse than a simple-
Mean-model

Timestamp 12:07

Case 4: $SS_{res} < SS_{total}$; $R^2 = 1 - (\text{something which is greater than 1}) = \text{-ve}$.

So, if R^2 is negative then our model is worse than a simple mean model.

So, based on the values of R^2 , we can determine how my model is performing with reference to a simple mean model.

Also there are a lot of different interpretations of coefficients of determination, please read the wikipedia article to know more about it.

https://en.wikipedia.org/wiki/Coefficient_of_determination

30.7 Median absolute deviation (MAD)

Median Absolute deviation of errors

$$SS_{res} = \sum_{i=1}^n e_i^2$$

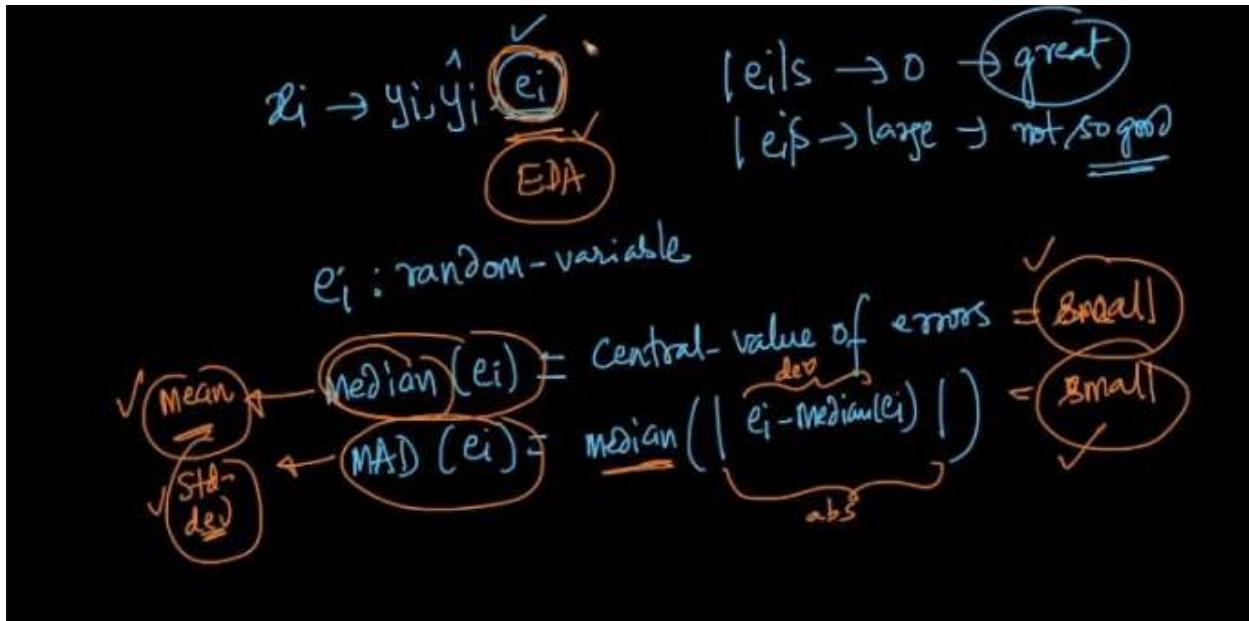
one e_3 is very large

R^2 is not very robust to outliers

MAD

Timestamp 1:21

There is one issue with R^2 , i.e. it is not very robust to outliers. Suppose there are outliers in my dataset and due to which one my error terms gets very large, this will impact our R^2 , giving us a wrong picture.

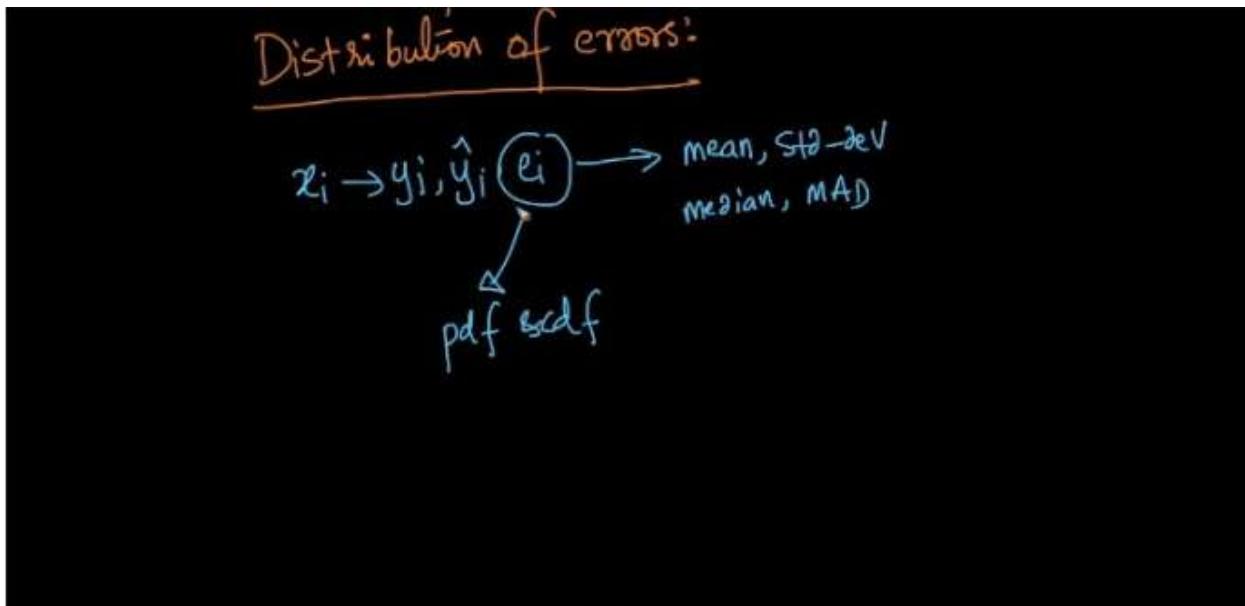


Timestamp 4:15

Now, we saw in earlier chapters that we calculated the mean of the residual errors to get the total error. Mean's are severely affected by outliers. So instead of using means for the central tendency we can use medians. So for total error we calculate the median of total errors. Also if

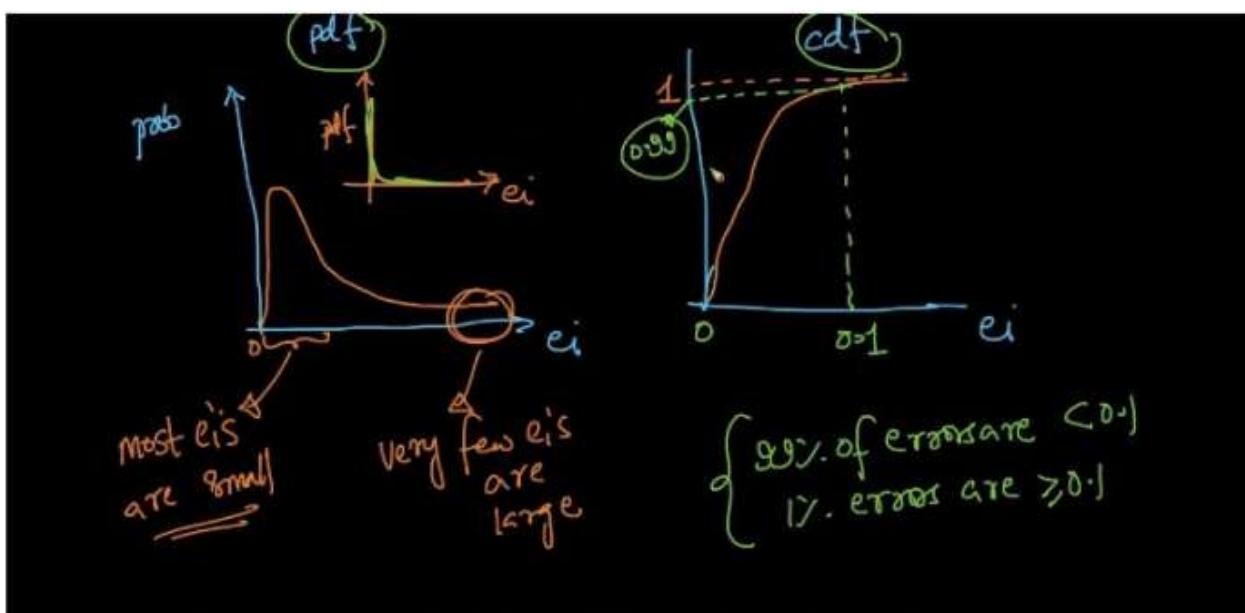
you want to calculate the standard deviation we can calculate the Mean Absolute Deviation, which is defined as the absolute difference between the error terms and their median. Please refer to the image above. If these values are small we can conclude that our model is good.

30.8 Distribution of errors



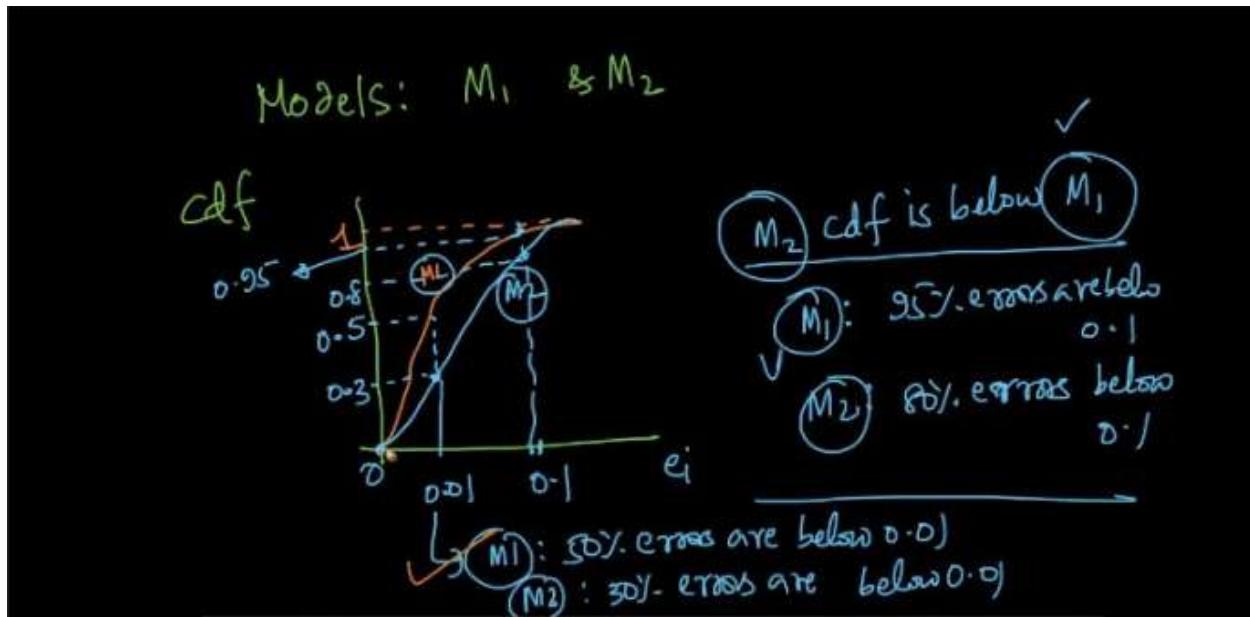
Timestamp 0:33

In addition to using central tendencies to analyze our errors. We can also use PDF's & CDF's.



Timestamp 3:31

Suppose we did a regression and we got our e_i 's. In the above image we have drawn the distribution of e_i 's. Now if we look at the graph of PDF, we can conclude that most of e_i 's are small and very few of them large. Ideally we want each of them to be as close to 0 as possible. The same can be concluded from the graph of CDF's also.



Timestamp 6:30

We can compare two models using the distribution of their errors. Consider a case where we trained two models M_1 & M_2 . Let's say the distribution of CDF observed by model M_1 is shown with the red curve whereas for model M_2 with the blue curve, as shown in the above image. Now we can take any value of e_i and observe the CDF's for these curves. Higher value of CDF will mean that more points have error below this point, meaning that particular model is better. We can see other examples from the image.

31.2 Imbalanced vs Balanced Dataset

Let us assume we have a binary classification problem. Let us consider a dataset ' D_n ' (with ' n_1 ' positive points and ' n_2 ' negative points).

$$n_1 + n_2 = n$$

If $n_1 \approx n_2$ (say $n_1 = 580$, $n_2 = 420$), then we can say that the given dataset is a **Balanced Dataset**.

If $n_1 << n_2$ (or) $n_2 >> n_1$ (say $n_1=100$, $n_2=900$ (or) $n_1=150, n_2=850$), then the given dataset is an **Imbalanced Dataset**.

Q) What happens when we apply K-NN on severely imbalanced data?

Ans) Let us assume we are working on a 2-class classification problem.

$n_1 \rightarrow$ Number of positive points

$n_2 \rightarrow$ Number of negative points

If $n_1 >>> n_2$, then we have a lot of positive points and a very few negative points. In case, if the negative points overlap with the positive points, then it becomes difficult to visualize, and the predictions are going to be the majority class biased for any new query point.

As K-NN is based on majority voting, almost all of the new query points are classified into the majority class.

Q) How to work around the imbalance dataset issue? (irrespective of the algorithm)

Ans)

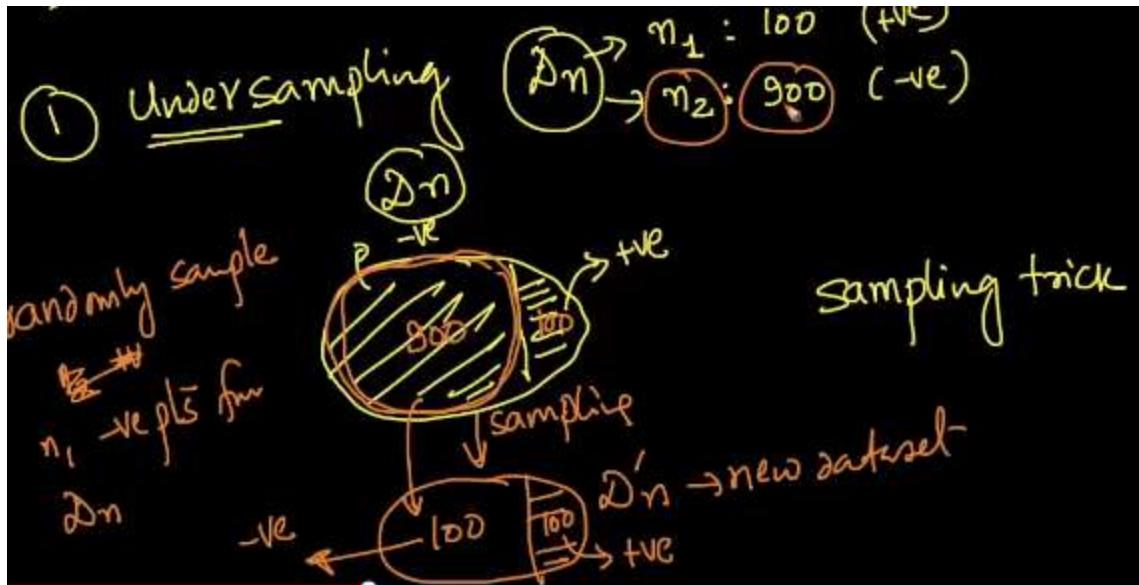
1) Undersampling

Let us consider the dataset ' D_n ' where

Number of positive points (n_1) = 100

Number of negative points (n_2) = 900

Here we have the negative class as the majority class. So as per undersampling, we have to create a new dataset D'_n . All the minority class data points should be copied from D_n to D'_n . From the majority class, data points have to be randomly sampled and those randomly sampled data points should be added to D'_n . The count of the data points in the random sample should be equal to the total number of minority class data points from ' D_n '.



Now we have our dataset D'_n in balanced form, and we should use it in data modelling from here onwards instead of using ' D_n '.

Problem with Undersampling

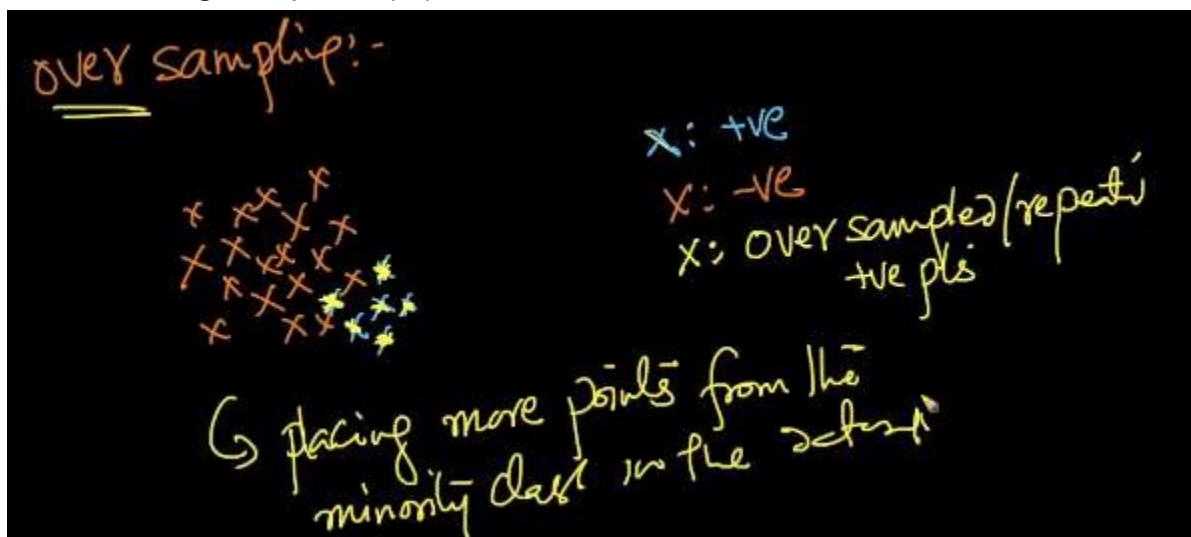
We so far had $D_n=1000$ and $D'_n=200$. Here we are removing nearly 800 data points. So data modelling is performed on a small amount of the data. As we all know that large amount of data leads to better results, small amount of data could not give good results as we are losing lot of information.

2) Oversampling

Our dataset 'Dn' has 1000 data points.

Let Number of positive points (n_1) = 100

Number of negative points (n_2) = 900



In oversampling, we create another dataset D_n' with all the majority class data points, and the same number of minority class data points. The minority class data points are increased by repeating the same data points multiple times. This way, we get our dataset balanced. We now have to use D_n' for data modelling from here onwards.

There are also certain complex ways of oversampling. Repeating the same data points is a simple idea. In the complex way, instead of repeating, we create artificial/synthetic points.

One of the ways of creating the artificial/synthetic points is by taking the region of the minority class points and creating artificial points in that region. This way of creating artificial points is known as **Extrapolation**.

3) Using the Class Weights

Let our dataset ' D_n ' have 1000 data points.

Number of positive points (n_1) = 100

Number of negative points (n_2) = 900

Now we shall take the class weights into consideration. The ratio of the class weights is inversely proportional to the counts.

$$\text{class_weight}_{+ve}:\text{class_weight}_{-ve} = (1/1):(1/9) = 9:1$$

So here we consider every positive data point as 9 positive data points, and then go for data modelling. Here we are not creating new points, but are assuring each point of minority class as more points. (proportional to the majority class)

Drawback of having Imbalanced Dataset

Let us assume we have 100 positive(n_1) and 900 negative(n_2) points in our dataset ' D_n '. Here it leads to a dumb model. A dumb model is the one which blindly gives the same result. It goes blindly with the majority class. So here we can get high accuracy with imbalance data on a dumb model.

So the metric 'accuracy' shouldn't be taken into consideration, when we are working with Imbalanced data. Instead of throwing away the data (in undersampling), we should better go for oversampling, as oversampling can fix much more fundamental problems.

Q) Why do we prefer oversampling over undersampling?

Ans) The whole point of oversampling is to ensure that our model doesn't classify all the points into the majority class. Oversampling ensures that we have roughly equal number of training points from each class thereby avoiding the model from choosing the majority class as the class label for all the test data points.

Q) When is it generally better to go for undersampling?

Ans) Undersampling is typically performed when we have billions of data points and we don't have sufficient memory(RAM) resources to process the data. Undersampling may lead to worse performance as compared to training the data on full data (or) the oversampled data, in some cases. In other cases, we may not have a significant loss in performance due to undersampling.

Undersampling is mainly performed to make the training of models more manageable and feasible when working within limited memory/storage resources.

31.3 Multi-class Classification

So far we have seen binary classifiers where the output variable can contain only 2 classes. In a multi-class classification problem, we can see the output variable can contain more than 2 classes.

Example:

Let us assume we are working on a 7-NN problem, and the distribution of the nearest neighbors of a given query point ' x_q ' are as follows.

0	6	1	0	...	0	number of nearest neighbors
1	2	3	4	...	C	classes

So if we have to classify a given query point ' x_q ',

- a) By majority vote, we get $y_q' = \text{class 2}$
- b) By probabilistic vote, we get

$$P(y_q'=2) = 6/7$$

$$P(y_q'=3) = 1/7$$

$$P(y_q'=1) = P(y_q'=1) = \dots = P(y_q'=1) = 0$$

K-NN can easily be extended to multi-class classification. But there are certain classification algorithms like Logistic Regression, SVM, etc which couldn't be extended to multi-class classification as K-NN. So in case, if we are working on a multi-class classification problem, we have to convert the 'C' class classification($C>2$) into 'C' binary classification problems.

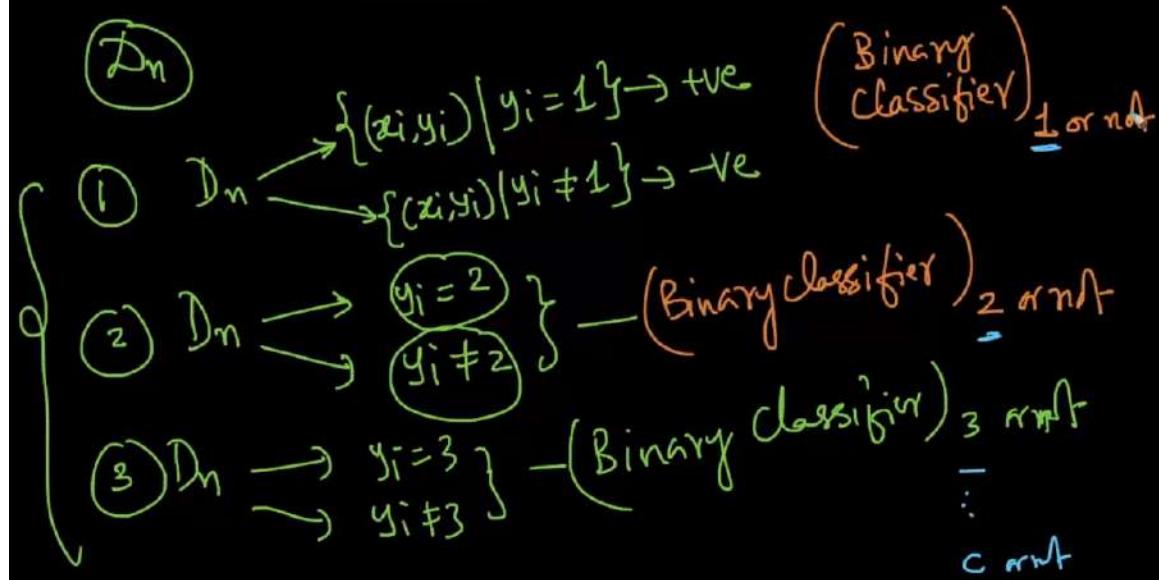
Conversion of a 'C' ($C>2$) class classification into 'C' binary classification problems

The given dataset is represented mathematically in the form of a set as

$$D_n = \{(x_i, y_i) | y_i \in \{1, 2, 3, \dots, C\}\}$$

So now we are splitting it into 'C' binary classification problems as shown below

$y_i \in \{1, 2, 3, \dots, C\} \rightarrow c$ binary classifiers



So a 'C' class classification problem can create 'C' binary classifiers. This solution is called one (vs) rest approach. KNN and Naive Bayes are excluded from this approach.

Here as 'C' binary classifiers are getting created, we get 'C' number of functions the models learn to map the input to the output values.

31.4 K-NN, given a Distance or Similarity Matrix

Sometimes we come across a situation where we could not represent the input ' x_i ' in a vector format (or) conversion of text to vector is not possible.

Let us assume we are working in the Pharmacy domain. Let ' x_i ' be a chemical compound. Conversion of the input of this compound into a vector is more difficult.

A pharmacist/domain expert couldn't convert a chemical compound into a vector format, but can give us the similarity scores between the pairs of compounds. So sometimes we do not get the input data in a vector form, but can get a similarity matrix as an input.

If we have 'n' data points in the training dataset, then the similarity matrix consists of 'n' rows and 'n' columns. The element at position (i,j) in the similarity matrix denotes the similarity score between the i^{th} and j^{th} compounds. The Similarity matrix is represented as shown below

$$\text{Sim}(x_i, x_j) =$$
$$S = \begin{matrix} & x_i \\ x_i & \begin{matrix} & & & \\ & & & \\ & \diagup & \diagdown & \\ & & & \end{matrix} \\ & x_j \end{matrix}$$
$$n \times n$$
$$S_{ij} = \text{Sim}(x_i, x_j)$$
$$\text{Dist : } d_{ij} = \frac{1}{S_{ij}}$$

The Similarity matrix is denoted by 'S' and has a shape $n \times n$. An element ' S_{ij} ' in the similarity matrix 'S' denotes the similarity score between the i^{th} and j^{th} compounds.
 $S_{ij} = \text{Sim}(x_i, x_j)$

Once we have the similarity matrix 'S', we can convert it into a distance matrix.
 $D \rightarrow d_{ij} = 1/S_{ij}$

Given a similarity matrix or a distance matrix, instead of $x_i \in R^d$ explicitly, we can easily apply K-NN, as K-NN cares only about the distances, but not about representation of input.

Techniques like Logistic Regression do not work as easy as K-NN, when the Similarity Matrix (or) Distance matrix is given, instead of vector representation of the input.

Note: For a new query point(in test dataset), the domain expert could generate the similarity scores between a query chemical compound and the existing compounds in the training dataset, using the formulae/rules/processes of Chemistry.

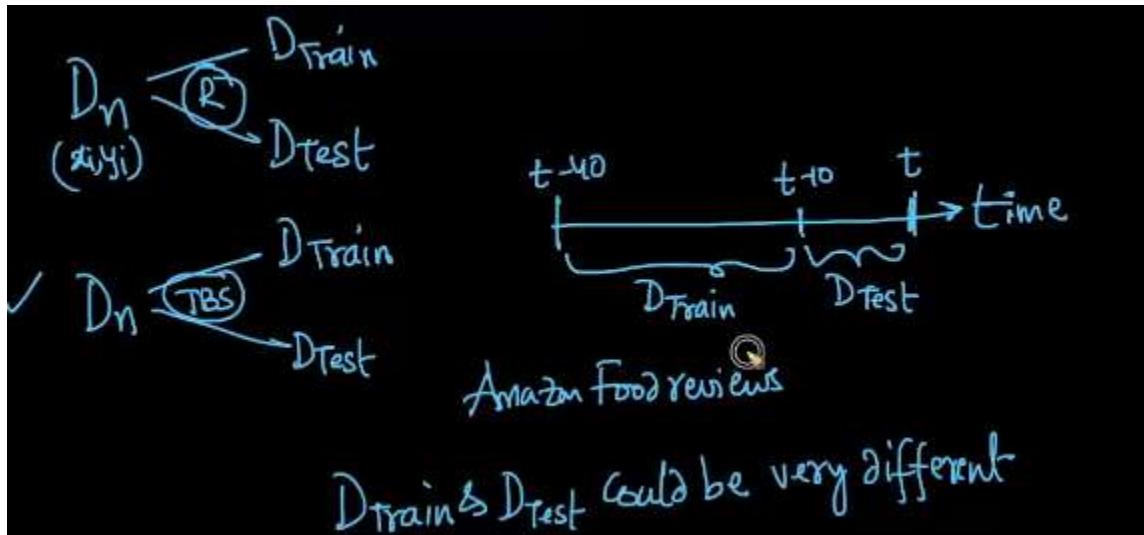
Once the domain expert has the similarity scores, K-NN is one simple strategy that he may use and is very common. Often times, computing the similarity matrix is a hard part.

Q) Is the similarity matrix the same as the correlation matrix?

Ans) Let us assume we have a dataset with 'n' data points and 'd' dimensions. Then the correlation matrix is of the order $d \times d$. Correlation matrix tells us how much correlated each feature is with the other feature.

Whereas in the similarity matrix, each cell ' S_{ij} ' tells us about the similarity between the i^{th} and the j^{th} compounds. The similarity matrix is of the order $n \times n$.

31.5 Train and Test Set Differences



So far we have seen a dataset (D) being split into D_{Train} and D_{Test} using random sampling where each data point has an equal chance of being selected into the sample.

But here a problem arises when we have time based splitting of the data. This problem occurs rarely in random splitting, but frequently in time based splitting.

The problem here is we get the data of new categories added in the test data over time. These categories might not be present in the training data. Also old category products that are not making good business might be removed from the training data. The underlying data changes over time and we have to be very careful, if we are applying time based splitting on the changing data.

Due to addition of the new category data and removal of some old categories data, the K-NN decision surface and the shape of the distribution also changes. The decision surface created using ' D_{Train} ' will work good for ' D_{Train} ', but not for ' D_{Test} '. The cross-validation error will be low, but the test error will be high. The distribution of test data has changed entirely to a new different region. In this case, we have to perform a check (ie., checking whether ' D_{Train} ' and ' D_{Test} ' have the same distribution or not)

How to check if D_{Train} and D_{Test} have the same distribution?

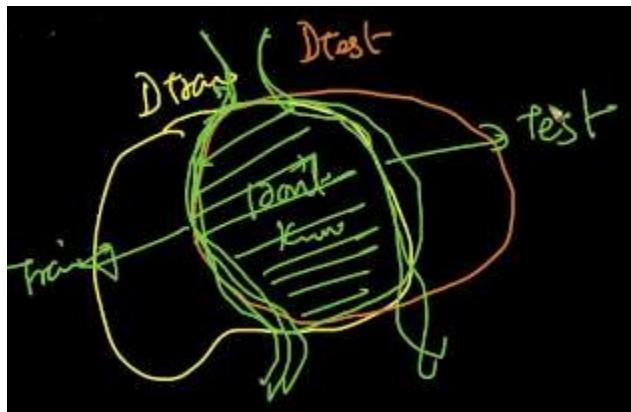
Let us assume we have a dataset ' D_n '. We split it into ' D_{Train} ' and ' D_{Test} ' using Time Based Splitting. Both ' D_{Train} ' and ' D_{Test} ' consist of both positive and negative points.

We shall now create a new dataset D_n' . For every data point (x_i, y_i) in both ' D_{Train} ' and ' D_{Test} ', we should create (x'_i, y'_i) .

$$y'_i = 1 ; x'_i = \text{concat}(x_i, y_i) \text{ in } D_{Train}$$

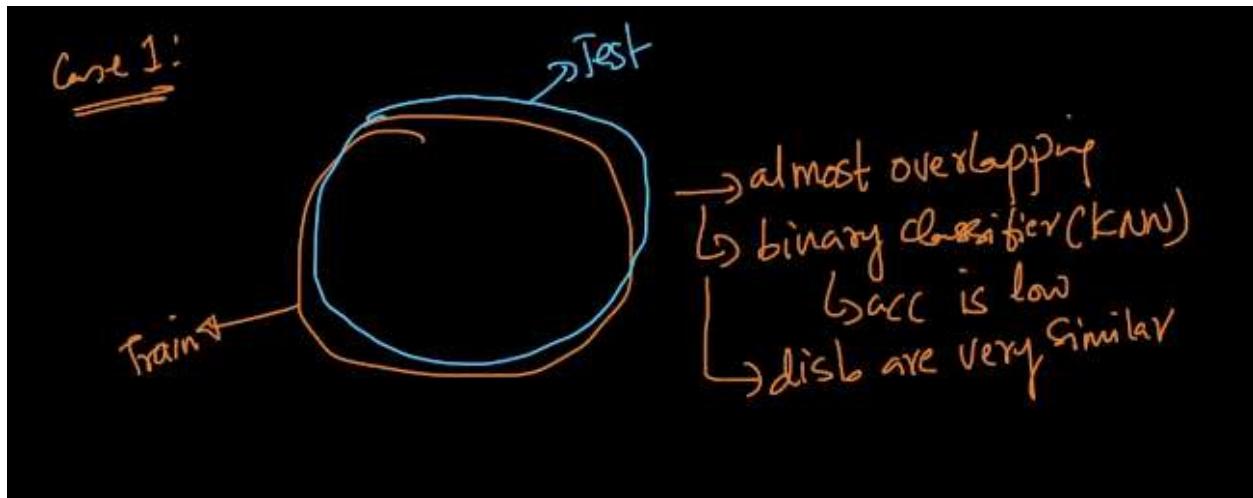
$$y'_i = 0 ; x'_i = \text{concat}(x_i, y_i) \text{ in } D_{Test}$$

We shall now build a binary classifier on D_n . Let $f(x)$ be the function that predicts the label as 0 or 1. Let the distribution of the points might be like this.



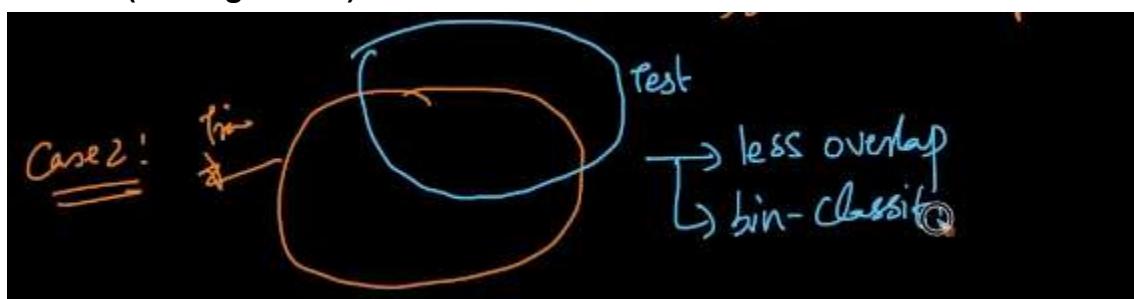
The binary classifier will be able to separate ' D_{Train} ' and ' D_{Test} ' to some extent. If this binary classifier gives an accuracy of 70%(say), it means in 70% of the cases, we are able to separate ' D_{Train} ' and ' D_{Test} '.

Case 1: (Best Case)



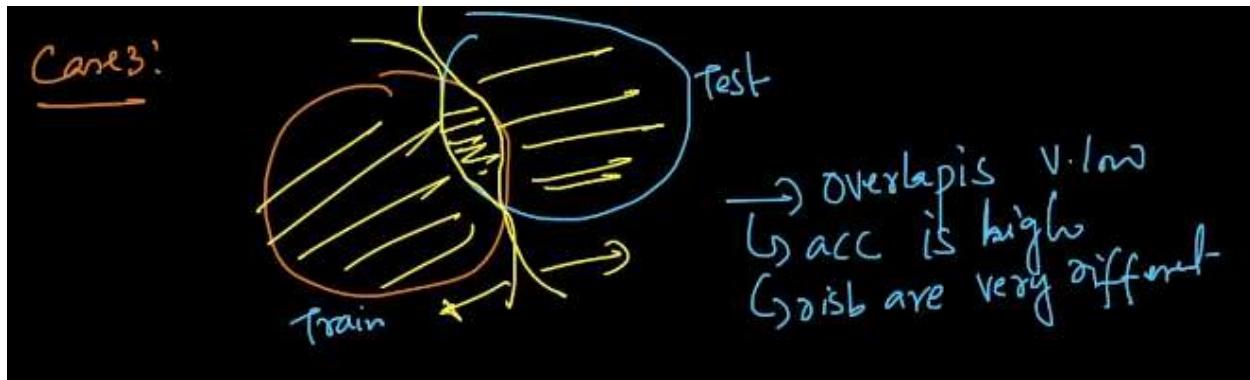
Here we have the distributions almost overlapping. Such cases lead to low accuracy. It means both the ' D_{Train} ' and ' D_{Test} ' distributions are similar.

Case 2: (Average Case)



Here we have less overlap. This case leads to medium accuracy. It means the distributions of ' D_{Train} ' and ' D_{Test} ' are similar.

Case 3: (Worst Case)



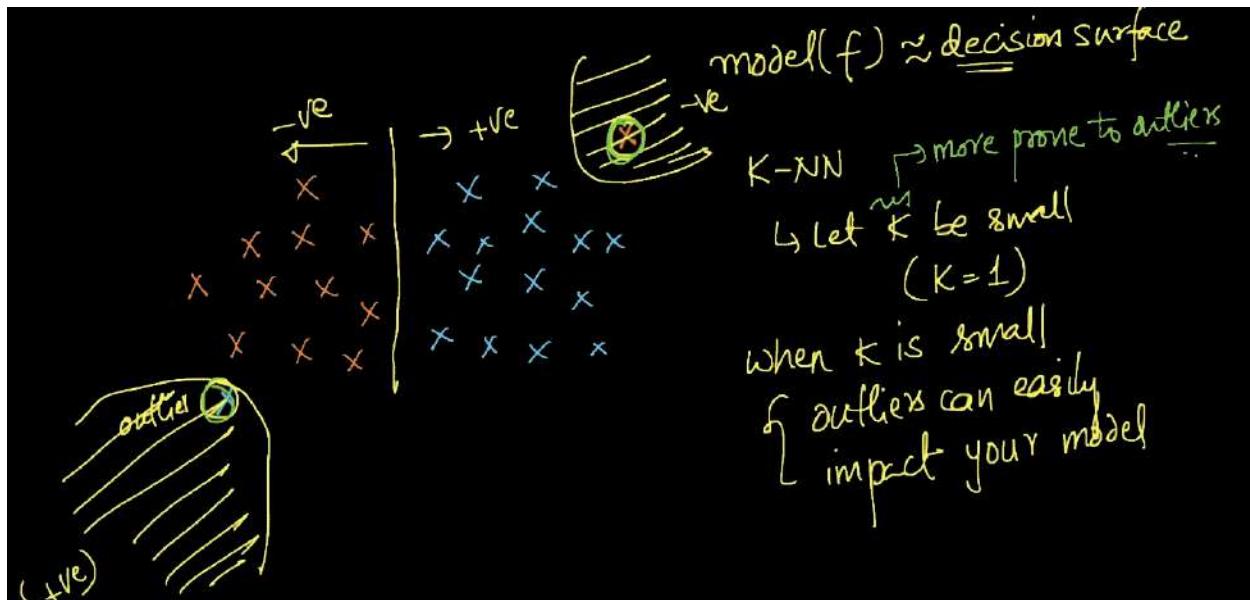
Here we have least overlap. This case leads to high accuracy, and the distributions of the ' D_{Train} ' and ' D_{Test} ' are different, and K-NN can perform the classification task very well.

Note: If ' D_{Train} ' and ' D_{Test} ' are from the same distribution, then the ML algorithms work perfectly well.

In case, if ' D_{Train} ' and ' D_{Test} ' are from different distributions, then no ML algorithm can perform well. So in such a case, it means the features chosen are changing over time (or) features are not temporally stable. In this case, we need to change/design/build new features.

31.6 Impact of Outliers

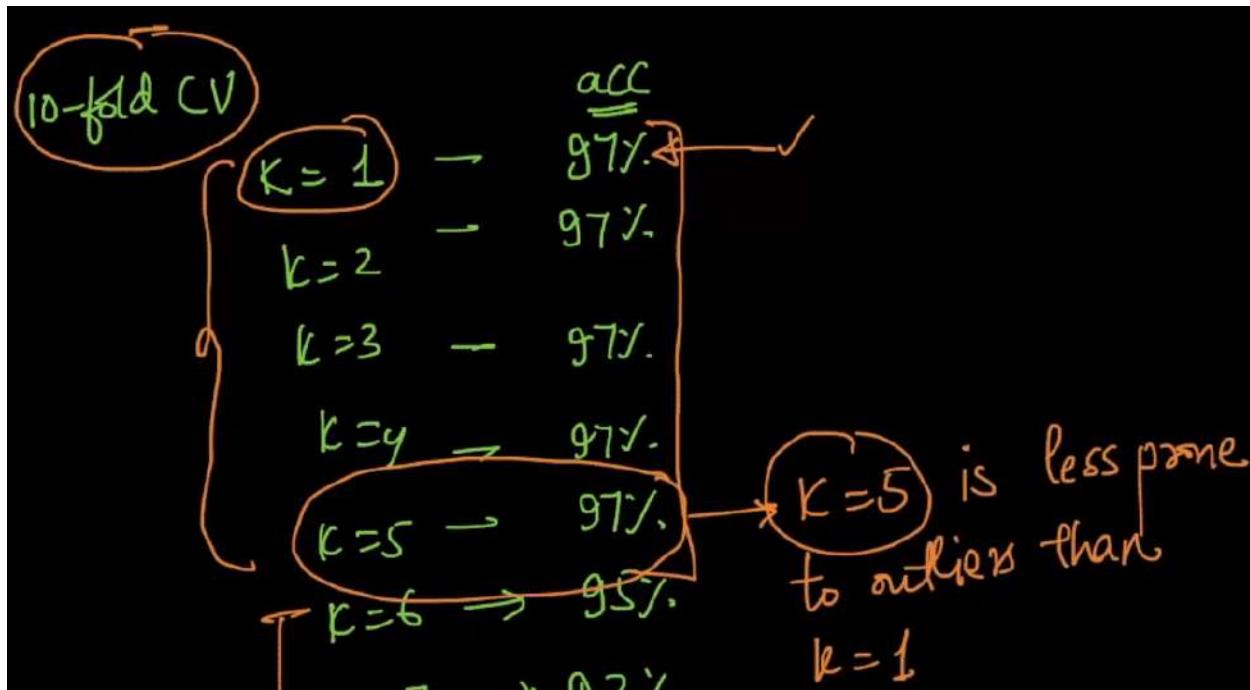
For every classification problem, we need to understand the impact of outliers on the model. A model can easily be understood by its decision surface, because the decision surface is the geometrical surface that separates the data points belonging to different classes.



Let us assume we are applying the K-NN model with $K=1$. In the left side region, we have a +ve point present in the region with -ve points. This point is an outlier. So if any query point falls nearer to this outlier, then by considering this 1 nearest neighbor(ie., the outlier), the query point will be classified as +ve, even though it is present in the region with the -ve class points in majority.

Similarly on the top right, we could see a -ve point falling in the region with the +ve class points in majority. Even here a new query point falls nearer to this negative point, by taking the 1-NN into consideration, that query point will get classified as -ve, even though it is present in the region with the +ve class points in majority.

So when the 'K' value is small, the outliers can easily impact our model. When the 'K' value is small, the model tends to be more prone(easily affected) by the outliers.



Let us assume we are performing 10-fold cross-validation for different values of 'K' (here it is 'K' in K-NN), then let us assume that we got the best accuracy score of 97% for $K=1,2,3,4,5,..$. In such a case, we have to choose $K=5$, as the optimal hyperparameter value because the model with $K=1$ is more prone to the outliers, whereas the K-NN model with $K=5$, is less prone to the outliers. It means the model with $K=5$ is less affected by the outliers.

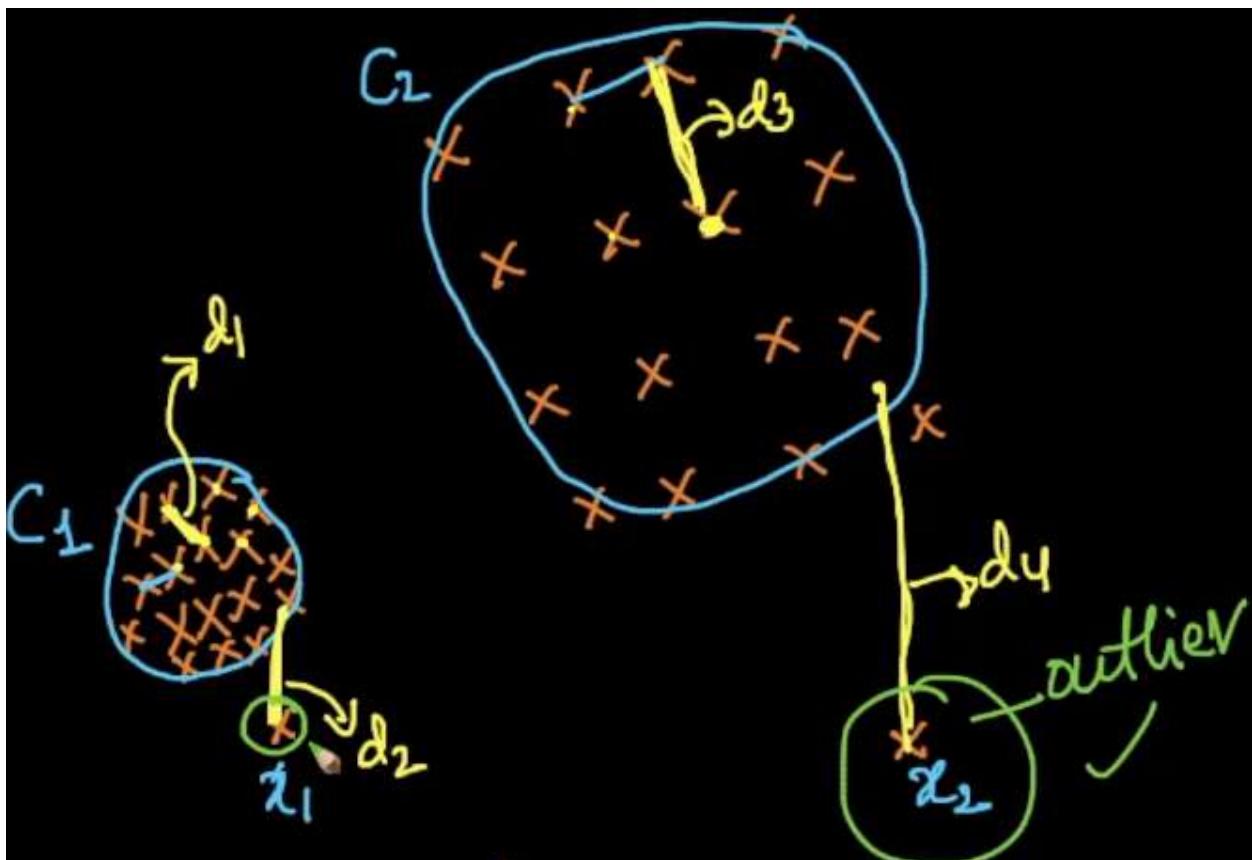
In the next set of lectures, we are going to discuss a topic called 'Local Outlier Factor' which is used to remove the outliers from our dataset.

31.7 Local Outlier Factor (Simple Solution: Mean distance to KNN)

Local Outlier Factor(LOF) is a technique used to remove the outliers from the dataset. It is inspired by K-NN.

Example

Let us consider a binary classification setting with +ve and -ve classes. We shall focus only on the -ve class points. These points are divided into two clusters ' C_1 ' and ' C_2 ' (' C_1 ' being a dense cluster and ' C_2 ' being a sparse cluster). ' C_1 ' and ' C_2 ' clusters contain only -ve points as of now. We have to first process all the -ve points and find out the outliers among them, and remove them from both the classes. Let us assume $K=5$, in K-NN.



Simple Solution

- 1) For every point ' x_i ', compute its ' K '(here $K=5$) nearest neighbors.
- 2) Compute the average of the distances of ' x_i ' to all its 5 nearest neighbors. (ie., $\text{avg}(d_1, d_2, d_3, d_4, d_5)$ where ' d_p ' is the distance of the point ' x_p ' to ' x_i ').

3) Sort all the points x_i 's by their average distances in ascending order.

If the average distance of a point ' x_1 ' is very high when compared to the other average distances, then we declare the point ' x_1 ' as an outlier.

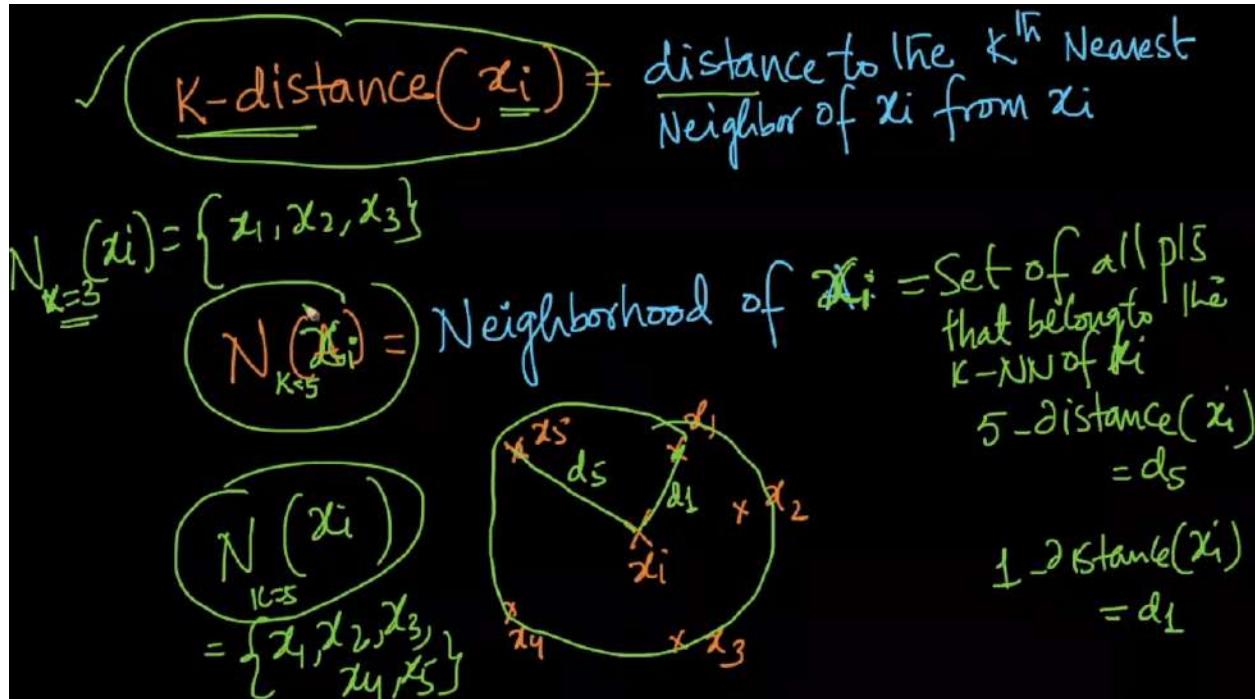
But this approach fails if we consider ' x_1 ' as an outlier because the distance of the point ' x_1 ' to its neighbors in the cluster ' C_1 ' is almost equal to the distance between the points in the cluster ' C_2 ', as ' C_2 ' is a sparse cluster, and the distances between the points in the cluster ' C_2 ' is high when compared to the distances between the points in the cluster ' C_1 '. Hence this approach fails in detecting the outliers, as it doesn't take the density of the points into consideration.

If we had all the points in denser form, and there are no sparse clusters, then this approach works well in detecting the outliers. But in the real world, it is not guaranteed that the data we get always is present in denser form. The technique used to detect the outliers should be able to do its job in both the simple and complex cases perfectly.

Hence while declaring a point as an outlier, we also should take the local density into consideration. We have discussed another approach for LOF in the next set of lectures, which also takes the local density into consideration at the time of detecting the outliers.

31.8 K-Distance

K-Distance of a point ' x_i ' is the distance of the point ' x_i ' to its K^{th} nearest neighbor.



In the above example,

- If $K=5$, then the distance of the point ' x_i ' to ' x_5 ' is the K-distance of ' x_i '.
- If $K=3$, then the distance of the point ' x_i ' to ' x_3 ' is the K-distance of ' x_i '.

Notation:

$\text{K-distance}(x_i) \rightarrow$ Distance of K^{th} nearest neighbor of ' x_i ' from ' x_i '.

$N(x_i) \rightarrow$ Neighborhood of ' x_i '

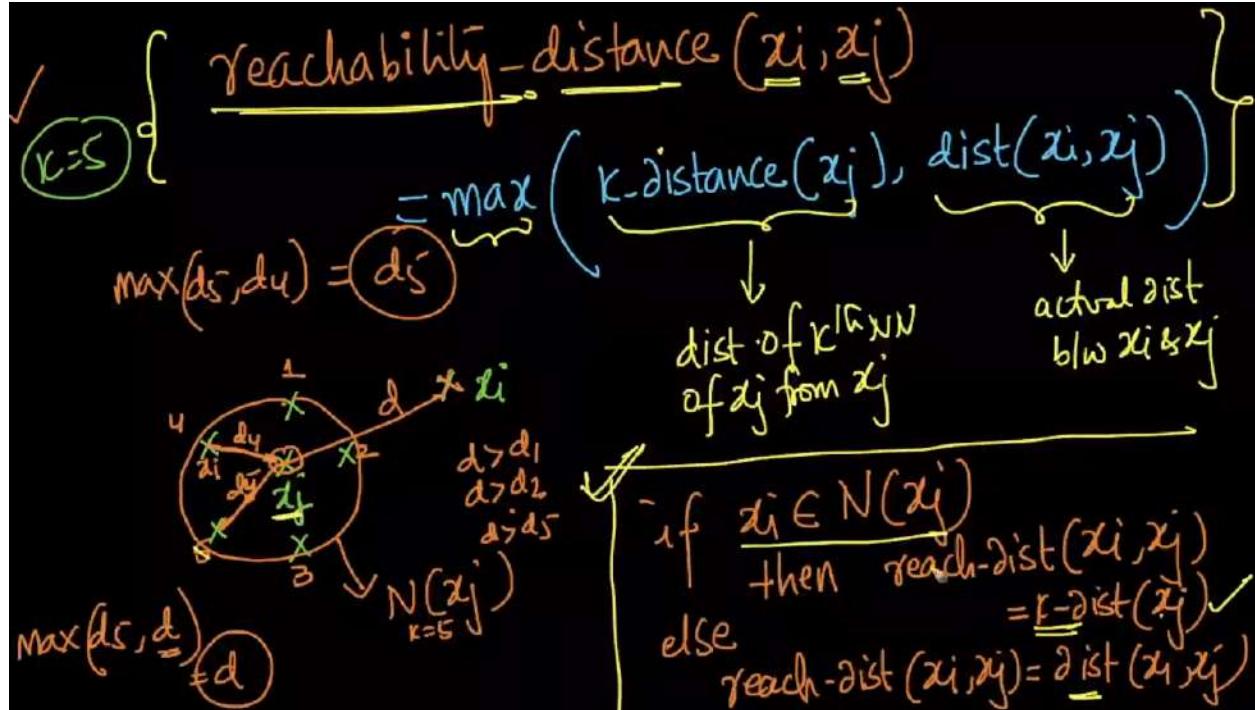
For example,

$N_{K=5}(x_i) \rightarrow$ Set of all the 5 nearest neighbors of ' x_i '. (ie., $\{x_1, x_2, x_3, x_4, x_5\}$)

$N_{K=3}(x_i) \rightarrow$ Set of all the 3 nearest neighbors of ' x_i '. (ie., $\{x_1, x_2, x_3\}$)

31.9 Reachability-Distance(A,B)

Reachability distance between the two given points ' x_i ' and ' x_j ' is given as the maximum among k-distance(x_j) and distance between the points ' x_i ' and ' x_j '.



$$\text{Reachability Distance}(x_i, x_j) = \max(\text{K-Distance}(x_j), \text{distance}(x_i, x_j))$$

Let us assume we have two given points ' x_i ' and ' x_j '.

- 1) If $x_i \in N(x_j)$, then it means ' x_i ' is one of the 'K' nearest neighbors of ' x_j '. In such a case,
 - a) If ' x_i ' is the Kth nearest neighbor (ie., if K=5, then ' x_i ' is the 5th nearest neighbor), then

$$\text{K-Distance}(x_j) = \text{distance}(x_i, x_j)$$

$$\text{Reachability Distance}(x_i, x_j) = \text{K-Distance}(x_j) = \text{distance}(x_i, x_j)$$

- b) If ' x_i ' is not the Kth nearest neighbor, (ie., if K=5, then ' x_i ' is not the 5th nearest neighbor), then

$$\text{K-Distance}(x_j) > \text{distance}(x_i, x_j)$$

$$\text{Reachability Distance}(x_i, x_j) = \text{K-Distance}(x_j)$$

- 2) If $x_i \notin N(x_j)$, then it means ' x_i ' is not one among the 'K' nearest neighbors of ' x_j ' (ie., if K=5 then ' x_i ' is not one among the 5 nearest neighbors of ' x_j '), then,

$$\text{distance}(x_i, x_j) > \text{K-Distance}(x_j)$$

$$\text{Reachability Distance}(x_i, x_j) = \text{distance}(x_i, x_j)$$

31.10 Local Reachability Density (A)

The Local Reachability Density of a point ' x_i ' is given as

$$LRD(x_i) = 1/(\sum_{x_j \in N(x_i)} \text{Reachability-Distance}(x_i, x_j)) / |N(x_i)|$$

$|N(x_i)| \rightarrow$ Number of points in the neighborhood of the point ' x_i '.

Local Reachability Density of a point ' x_i ' is defined as the inverse of the average reachability distances of the point ' x_i ' from all the points in its neighborhood.

The denominator term in the formula denotes the average reachability distance of the point ' x_i ' from all the points in its neighborhood.

The same above given formula can be written as

$$LRD(x_i) = |N(x_i)| / (\sum_{x_j \in N(x_i)} \text{Reachability-Distance}(x_i, x_j))$$

This equation is now in the form of (**Number of points/measure of distance**), which is the standard form of a density.

Local → Because we are looking only the neighborhood

Reachability → Because we are considering only the reachability distances

Density → As the formula is in the form of (**Number of points/measure of distance**)

Hence we got the name **Local Reachability Density**.

31.11 Local Outlier Factor (A)

Local Outlier Factor (LOF) is a technique used to detect the outliers in the given data. The LOF is given by the formula as mentioned below

$$\text{LOF}(x_i) = ((\sum_{x_j \in N(x_i)} \text{LRD}(x_j)) / |N(x_i)|) * (1/\text{LRD}(x_i))$$

Here we can see the LOF value is a product of two terms. The first term denotes the average LRD of points in the neighborhood of ' x_i '.

LOF(x_i) is large when LRD(x_i) is small (or) average LRD of the points in $N(x_i)$ is large.

LOF(x_i) is small when LRD(x_i) is large (or) average LRD of the points in $N(x_i)$ is small.

If **LOF(x_i) is large**, then we call it an **Outlier**.

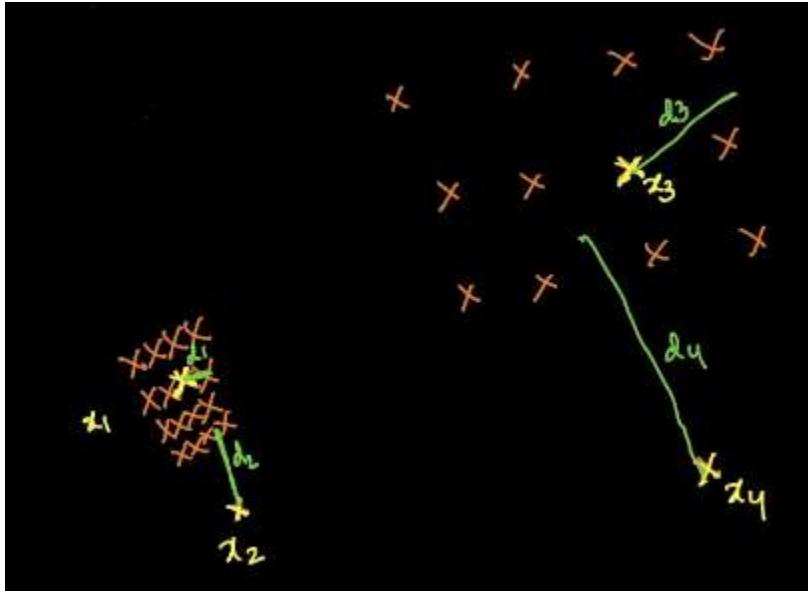
If **LOF(x_i) is small**, then we call it an **Inlier**.

So we can say a point is an outlier if the density around that point is low when compared to the density around its neighborhood.



In the above points, the point circled is our query point, and its K-nearest neighbors are present in the given cluster of points. Here we see the density around the K-nearest neighbors of our query point is so high, but the density around our query point is low. Hence LOF concluded this query point as an outlier.

Let us now look at another example.



$d_1 \rightarrow$ Average Reachability Distance of ' x_1 '

$d_2 \rightarrow$ Average Reachability Distance of ' x_2 '

$d_3 \rightarrow$ Average Reachability Distance of ' x_3 '

$d_4 \rightarrow$ Average Reachability Distance of ' x_4 '

From the figure, we can say that $d_1 < d_2 < d_3 < d_4$

Local Reachability Density $\propto 1/(\text{Average Reachability Distance})$

LSD $\propto 1/d$

So as $d_1 < d_2 < d_3 < d_4 \rightarrow 1/d_1 > 1/d_2 > 1/d_3 > 1/d_4 \rightarrow \text{LRD}(x_1) > \text{LRD}(x_2) > \text{LRD}(x_3) > \text{LRD}(x_4)$

From the above distribution of points,

$\text{LRD}(x_1) \approx \text{LRD}(x_j \in N(x_1))$

$\text{LRD}(x_2) \approx \text{LRD}(x_j \in N(x_2))$

$\text{LRD}(x_3) \approx \text{LRD}(x_j \in N(x_3))$

$\text{LRD}(x_4) \approx \text{LRD}(x_j \in N(x_4))$

The densities around the points ' x_1 ' and ' x_3 ' is almost same as the densities around $N(x_1)$ and $N(x_3)$ respectively, as ' x_1 ', $N(x_1)$ and ' x_3 ', $N(x_3)$ lie in the same clusters.

The densities around the points ' x_2 ' and ' x_4 ' are less than the densities around $N(x_2)$ and $N(x_4)$ respectively, as ' x_2 ' and ' x_4 ' are away from the clusters that contain $N(x_2)$ and $N(x_4)$ respectively.

So for

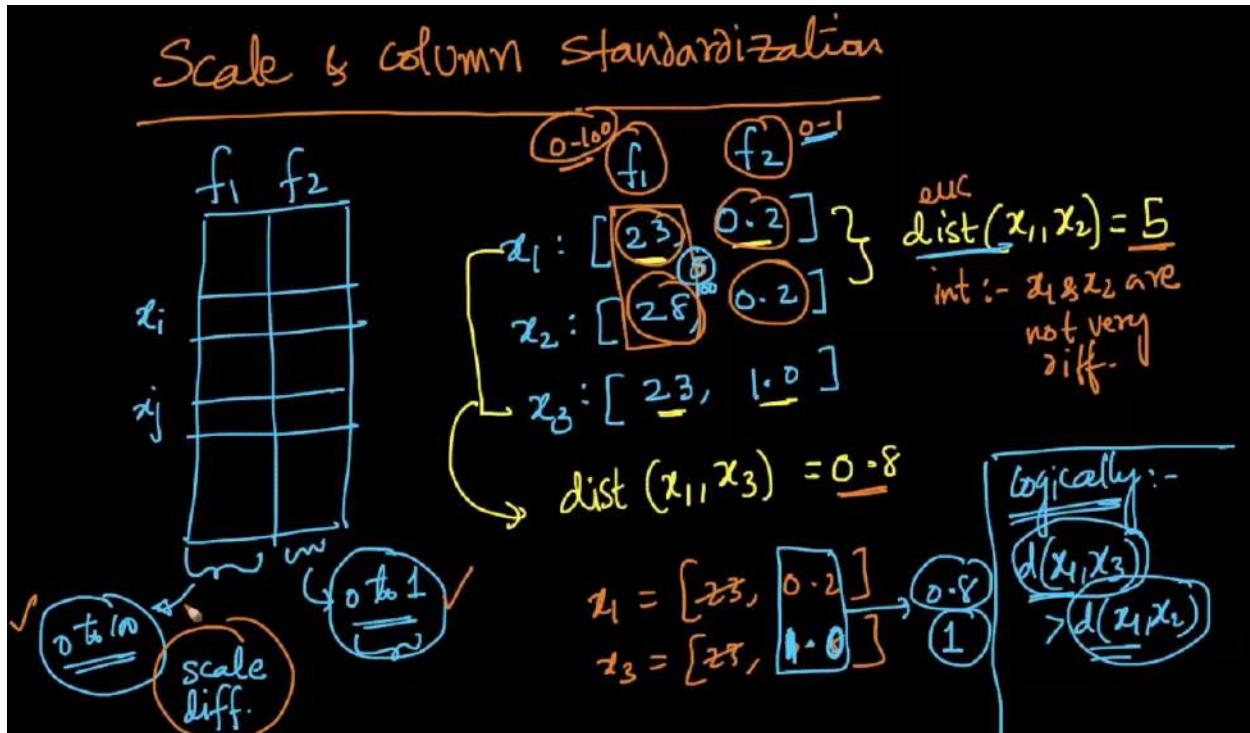
$x_1, x_3 \rightarrow \text{LOF is small}$

$x_2, x_4 \rightarrow \text{LOF is large}$

31.12 Impact of Scale and Column Normalization

Let us assume we have two features ' f_1 ' and ' f_2 '. Let us assume the values of ' f_1 ' lie in the range 0-100 and the values of ' f_2 ' lie in the range 0-1. Let us consider the below given points as an example

	f_1	f_2
$x_1:$	23	0.2
$x_2:$	28	0.2
$x_3:$	23	1



When we consider the pair (x_1, x_3) , we see the values of ' f_1 ' are the same. Only the values of ' f_2 ' differ.

So $\text{Distance}(x_1, x_3) = 1 - 0.2 = 0.8$ (min-max range of ' f_2 ' is 0-1). Here the maximum value is 1 and the difference value is 0.8, which is 80% of the maximum value.

When we consider the pair (x_1, x_2) , we see the values of ' f_2 ' are the same. Only the values of ' f_1 ' differ.

So $\text{Distance}(x_1, x_2) = 28 - 23 = 5$ (min-max range of ' f_1 ' is 1-100). Here the maximum value is 100 and the difference value is 5, which is 5% of the maximum value.

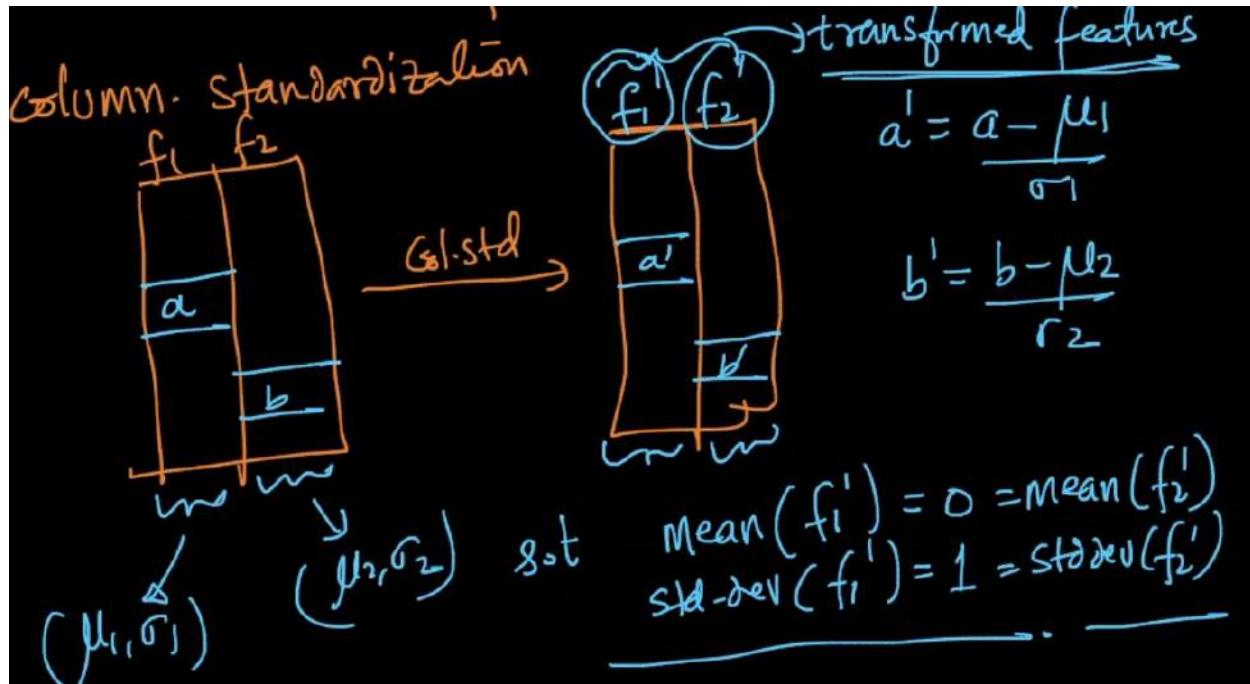
When we look into these differences, we can say **logically** $\text{Distance}(x_1, x_3) > \text{Distance}(x_1, x_2)$, but **mathematically** $\text{Distance}(x_1, x_2) > \text{Distance}(x_1, x_3)$.

So these mathematical and logical variations are due to change in the scales of the features. With such features, if we go ahead with model building, then definitely that

would mess up our models. Hence in order to get rid of this problem, we apply Column Standardization.

If we apply Column Standardization, all the features/columns will fall almost into the same range(not exactly the same range), and also the nature of the distribution of the features doesn't change at all.

Column Standardization is a technique which transforms our features in the dataset to a new set of features with each column having a mean '0' and standard deviation '1'.



If f_1 and f_2 are our original features, let us assume f'_1 , f'_2 are the transformed features, μ_1 , μ_2 are the means of f_1 , f_2 and σ_1 , σ_2 are the standard deviations of f_1 and f_2 respectively. Then

$$f'_1 = (f_1 - \mu_1)/\sigma_1$$

$$f'_2 = (f_2 - \mu_2)/\sigma_2$$

Since Euclidean distance can easily be impacted by the differences in the scales, it is important to perform column standardization. ML techniques like K-NN, Logistic Regression, SVM are scale dependent, whereas the techniques like Naive Bayes, Decision Trees, Ensemble models are scale independent.

Note: Standardization doesn't ensure that all the resulting values are in a fixed interval. It only removes the scale effect and ensures the mean = 0 and standard deviation = 1.

31.13 Interpretability (Model Interpretability vs Blackbox)

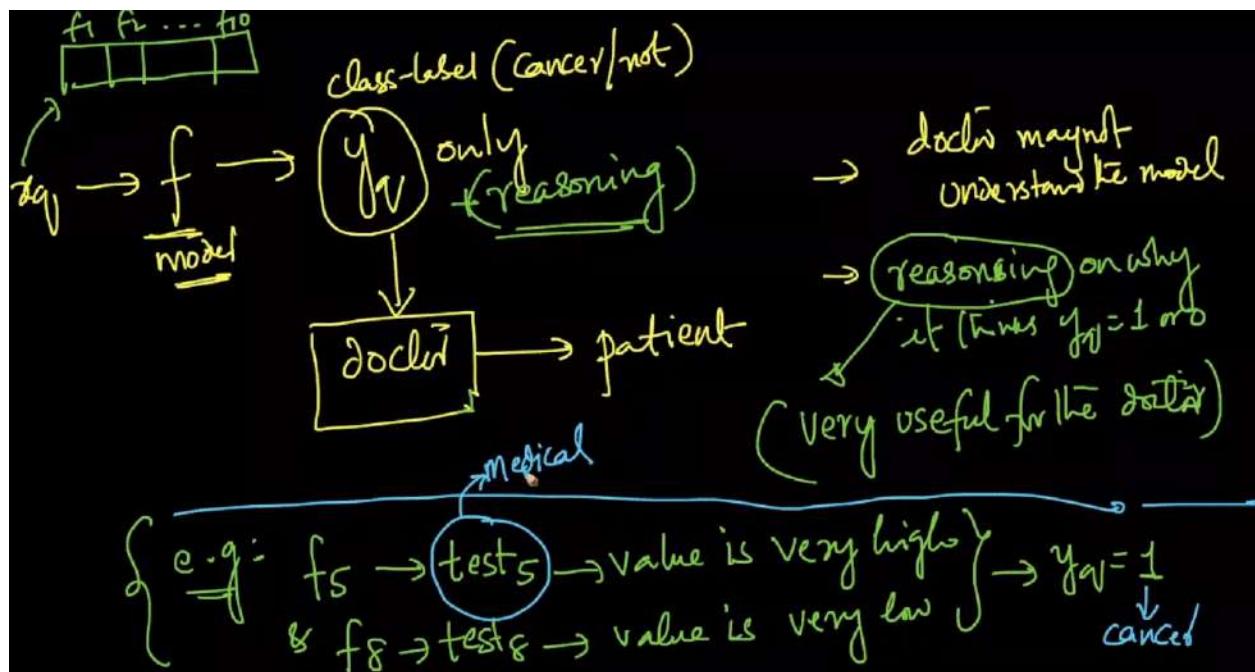
If a model just predicts the result without giving any reasoning, then we call it a **Blackbox model**.

If a model gives reasoning and explains the reason for the occurrence of a value as an output, such a model is called an **Interpretable model**.

Example

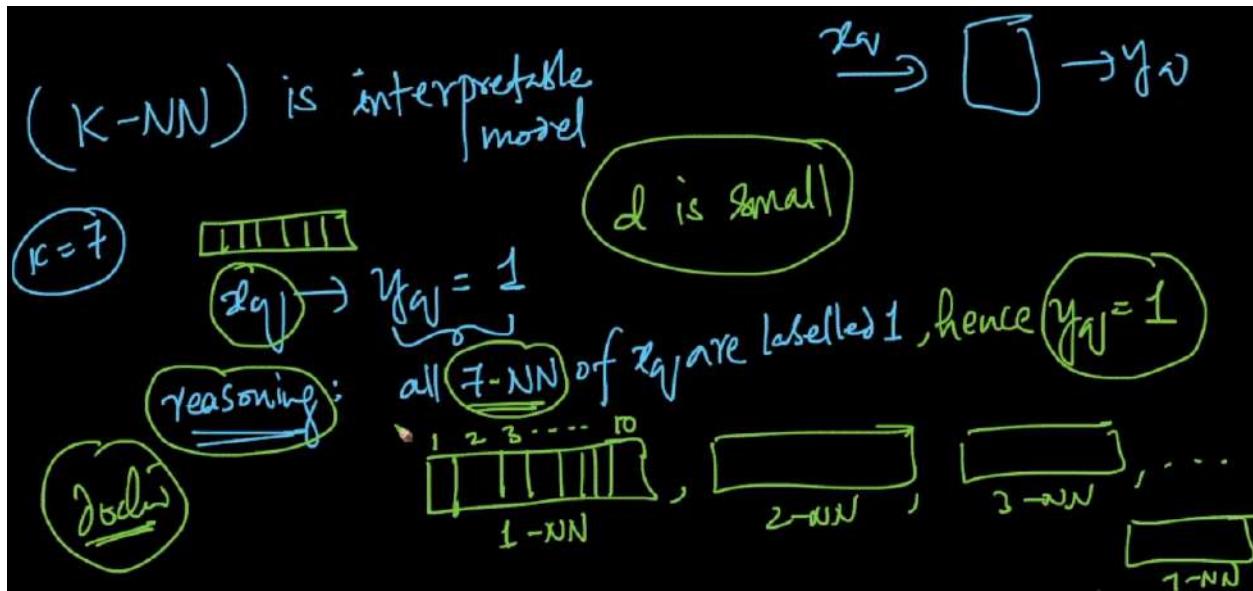
Let us take an example from the medical domain. We have to build a model that takes the patient data as input and predicts whether the patient has cancer or not.

The ML model predicts the class label and all the times the model may not give an accurate prediction. Sometimes it goes wrong as well. Also making a decision, without knowing what exactly is happening is also not correct. In order to confirm whether a patient has cancer or not, the patient has to go through some tests, and depending on the results of those tests, the doctor can make a confirmation.



So we pass the results of those prior tests as the input data to our model. Without knowing the information about the data and which features are contributing more in our predictions, even though the model gives the result, still it could not be considered sometimes. Hence it is required to know what all features are contributing more to our model at prediction, and what all features are contributing less, is also very important. Making interpretations will add huge weightage and confidence to our analysis and the results. Such models which can also give interpretation along with the predictions are called **Interpretable models** and the models which could not give interpretations, but can only give the predictions are called **Blackbox models**.

Is K-NN an interpretable model?



If we are working on a K-NN problem (with $K=7$), then let us assume all the 7 patients have the class label $y=1$ (it means all of them are diagnosed as Cancer). In such a case, by taking all the 7 nearest neighbors into consideration, we can confirm that the patient is also having cancer. (It is because those 7 neighbors picked are nearest to our query patient. In such a case, we do not find much difference in the data vectors of these 7 patients. The prior test results are similar in all these 7 patients). Hence we can confidently confirm that the given patient is having cancer.

K-NN is interpretable when 'd' and 'K' are small. (It differs from domain to domain). It is because when 'd' is small, it is easier to check each of the features to understand what is happening amongst the nearest neighbors, and understand the reason behind the decision.

When 'K' is small, it is easier to go through each of the data points individually, making the interpretation easier. For example, if $d=100$ and $K=21$, the need to go through 100×21 feature values to understand the reasoning behind the decision is sometimes not possible for human beings.

31.14 Feature Importance and Forward Selection

Feature Importance is the process of sorting the features in the order of importance for the classification/regression task.

Feature Importance is useful in understanding a model better. Once we understand a model better, the model interpretability increases.

For example, we are predicting the height of a person (which is a regression task). The given features in our data are weight, hair color, hair length, skin color, gender, country, etc. Let us assume, the features weight, gender, country and skin color are playing a major role in deciding the height value, then we can say these are the important features. These important features add more weightage to the model at the time of prediction.

Q) Is there a way to get the feature importance directly using K-NN?

Ans) K-NN couldn't give the feature importance easily. ML models like Logistic Regression, Decision Trees, Linear SVM, Naive Bayes, etc can give the feature importance easily.

There are certain hacks that are used to obtain the feature importance for any model. These hacks are model independent. It means these techniques/hacks can be applied on any ML model.

Feature Selection is the process of discarding the least important features from the model, and keeping the more important feature only. One of the techniques of Feature Selection is **Forward Feature Selection**.

Procedure of Forward Feature Selection

- 1) Take the dataset ' D_n^d ' (say $d=10$).
- 2) Fit a model(say K-NN) with only one dimension at a time. This way, we get 10 models. Choose the one which gives the highest accuracy. Let's say the model built using only ' f_2 ' gives the highest accuracy.
- 3) Now keep the feature ' f_2 ' in our model, and try fitting the model including the other features also, but only one at a time. (ie., (f_2, f_1) , (f_2, f_3) , (f_2, f_4) , (f_2, f_5) , (f_2, f_6) , (f_2, f_7) , (f_2, f_8) , (f_2, f_9) , (f_2, f_{10})). Pick the combination whichever gives the highest accuracy.
- 4) Let us assume we got ' f_{10} ' as the next important feature, then we say we have two important features ' f_2 ' and ' f_{10} '.
- 5) Now we have the two important features ' f_2 ' and ' f_{10} ', we have to try combinations with keeping these 2 features and adding one feature at a time as the third feature. (ie., (f_2, f_{10}, f_1) , (f_2, f_{10}, f_3) , (f_2, f_{10}, f_4) , (f_2, f_{10}, f_5) , (f_2, f_{10}, f_6) , (f_2, f_{10}, f_7) , (f_2, f_{10}, f_8) , (f_2, f_{10}, f_9)). Let us assume the model with the combination (f_2, f_{10}, f_5) gives the highest accuracy. Then ' f_5 ' is selected as the 3rd important feature. Next we have

to try fitting the models with 4 features, keeping ' f_2 ', ' f_5 ' and ' f_{10} ' constant in the model.

This way, we have to continue the process. At each stage, given that we already have some features, we have to check which new feature adds the most value to our model. These iterations are continued till we don't find any improvement in the model accuracy, even after adding new features.

Time Complexity

At each iteration, we are training and testing d-models.

1st iteration → 'd' models

2nd iteration → 'd-1' models

3rd iteration → 'd-2' models

The time complexity is very high in Feature Selection Techniques. They do not care about the model we are building. (ie., type of the algorithm used)

The advantage of Feature Selection techniques is they are independent of the type of ML algorithm used, and the disadvantage is the time complexity.

31.15 Handling Categorical and Numerical Features

Let us assume we are working on the problem of predicting the height of a given individual. If a given feature consists of numerical values, it can be taken directly into building a model. If it consists of categorical values, then it has to be converted into numerical form, and then should be taken into building a model.

Let us assume our features are weight, hair color, country, hair length, gender, etc. The weight feature consists of only numerical values, so it can be taken directly into building a model.

The hair-color feature consists of categorical features. Let those values be Hair-color = {'black', 'brown', 'red', 'golden', 'gray'}. As these values cannot be directly used in model building, we have to convert them into numerical form.

Ways to handle these categorical features

1) Giving a number to each value

Let us assign number to these categorical values as shown below
'Black' = 1, 'Brown' = 2, 'Red' = 3, 'Golden' = 4, 'Gray' = 5

When we give such numerical values, we see there is an ordering among the values. (ie., 3>1, 5>1 which mean 'Red'>'Black' and 'Gray'>'Black'). This is totally absurd and all these values here are discrete. Logically there doesn't exist any ordering among these categorical features. Hence this approach is not at all recommended.

2) One-Hot Encoding

One-Hot Encoding creates a binary vector of size equal to the distant number of elements. As we have 5 values of hair-color, we get a binary vector of 5-dimensions.

Black: 10000

Brown: 01000

Red: 00100

Golden: 00010

Gray: 00001

In case of one-hot encoding, only one bit will be set to 1, and the rest all bits are set to 0. Hence we get a sparse and large vector, if the number of distinct values of a categorical feature is high.

One-Hot Encoding approach doesn't work on the test data, if any unseen categorical value gets added in the test data.

3) Mean Replacement

For example, if we have 200 distinct values in the ‘country’ column, then if we apply one-hot encoding on it, we get a 200-dimensional vector. Here as the number of distinct values in the categorical feature is very high, we get a large and a sparse vector.

One hack to handle this problem is to replace the ‘country’ column values with the average height of all the individuals of those countries. For example, if we have one of the ‘country’ column values as ‘India’, then ‘India’ has to be replaced with the average of the height values of all the individuals whose ‘country’ column value in the dataset is ‘India’. This way, we could avoid occurrence of high dimensionalities.

So in this approach we are replacing each category of a categorical feature with the average of the output values(y_i) having the same category value. This approach is problem-specific.

4) Domain Knowledge

Taking a value as standard and replacing the value with the distance. For example, if we are considering ‘India’ as the standard value in the ‘country’ column, then we have to compute the distance from ‘India’ to that country, and fill this distance value in place of the country name.

This is because, we are taking a point as reference and are measuring/comparing the other values with respect to this point.

Ordinal Features

Ordinal features are the features which have categorical values, but there is an ordering between the values.

For example, if we want to rate a service/product, then the values could be

Very Good → 5

Good → 4

Average → 3

Bad → 2

Very Bad → 1

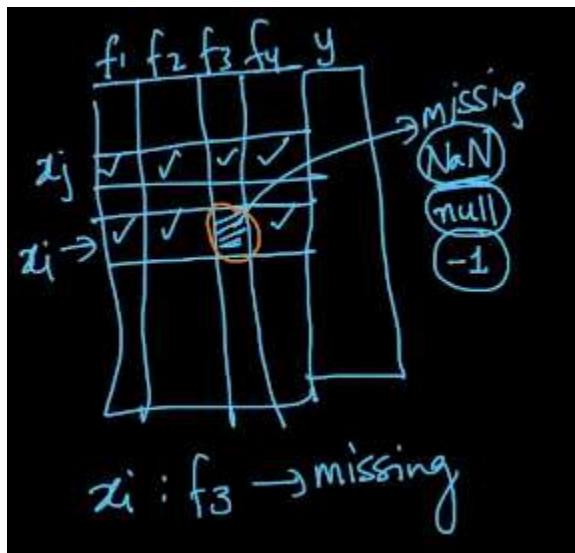
We can give numerical values for these ratings, and it is accepted.

Note: There is no strong rule of converting a categorical variable to numeric. We have to keep trying all the above discussed techniques and see which works best according to the problem context.

31.16 Handling Missing values by Imputation

Missing data occurs very frequently in the real-time. It could be due to

- a) Data getting corrupted
- b) Collection error (the researchers might have forgotten to collect)



1) Imputation in case of a Regression problem

One of the featurization techniques to handle the missing data is by **Imputation**. Imputation is the process of replacing the missing values with something else. There are 3 strategies in Imputation.

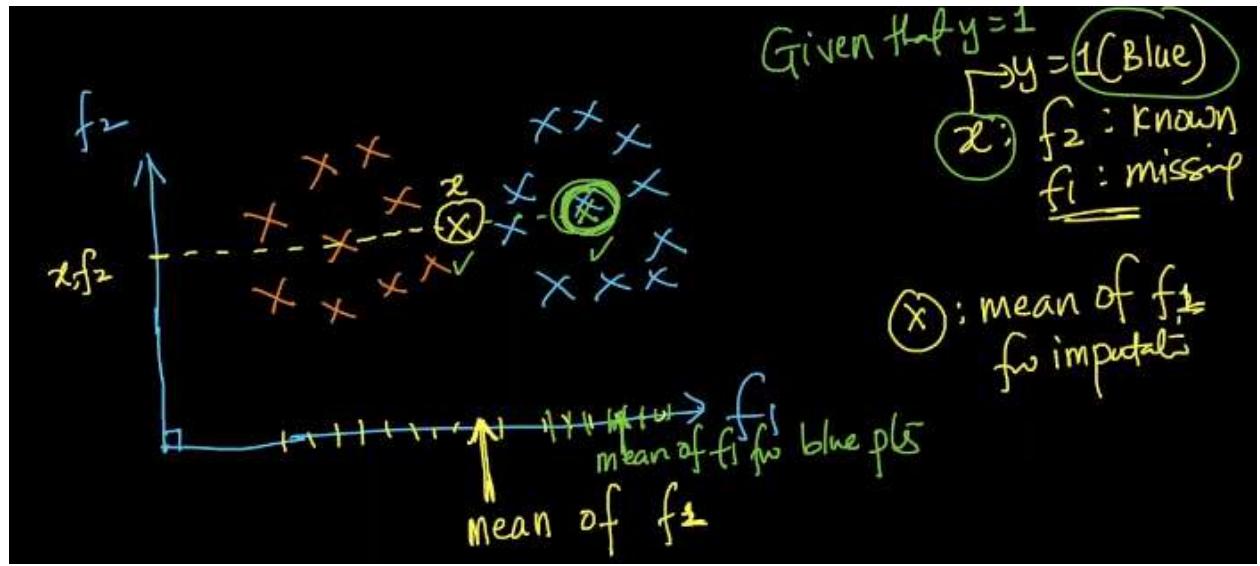
- (i) Replacement with mean
- (ii) Replacement with median
- (iii) Replacement with mode

2) Imputation in case of a classification problem

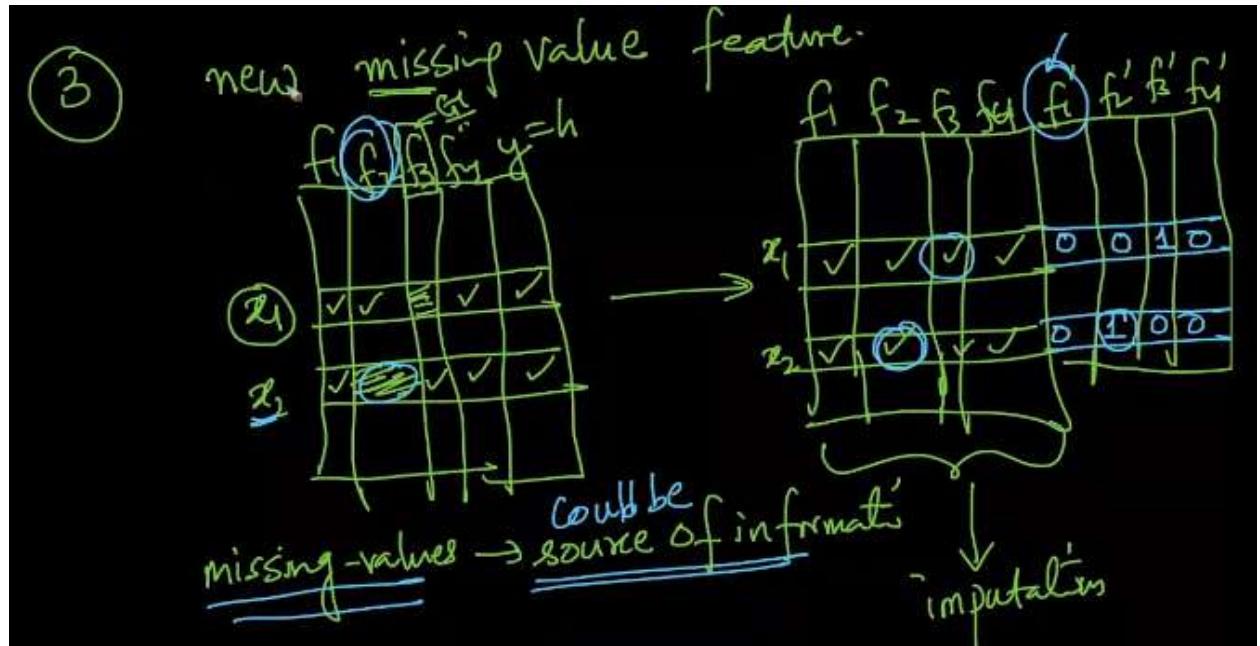
So far in the case of a regression problem, if $x_i \in R^d$ is our input, say a point ' x_i ' has got missing values. Then we used to replace those missing values with the mean/median/mode of the non-missing values among those features.

In case of a classification problem, we have to replace a missing value of a data point with the average of all the values of that column , which have the same class label as that of the missing point. This gives a better result.

So far we have discussed 2 ways. They are *Imputation without the class label* and *Imputation with the class label*.



3) Creating New missing Value Feature

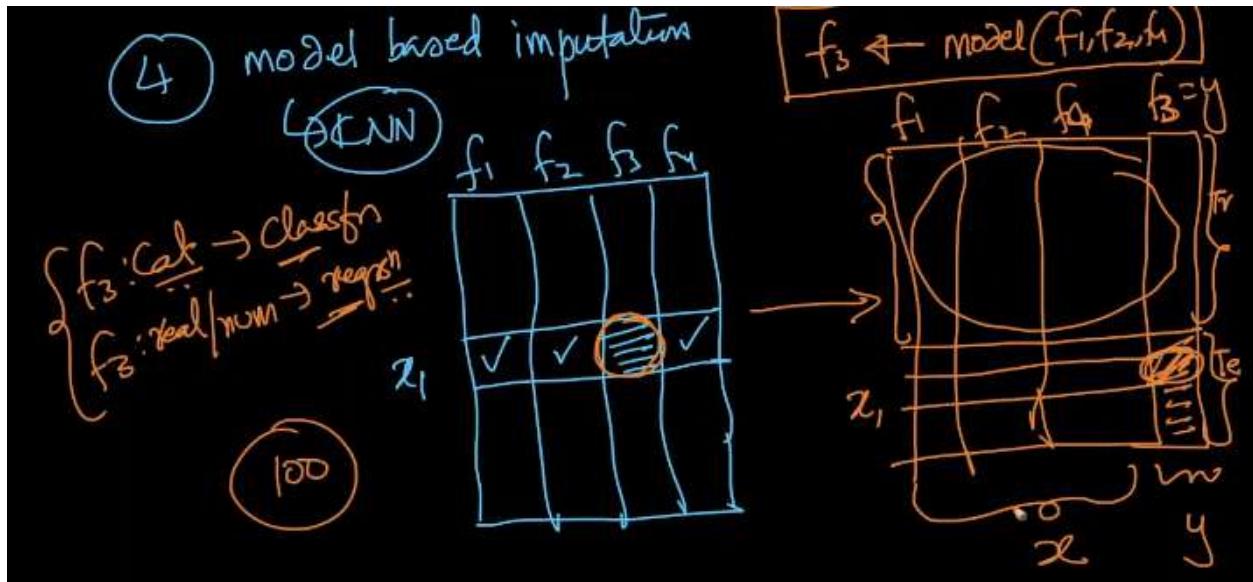


In this approach, we are first filling all the values by Imputation. We have to check how many features in the dataset contain missing values, and we have to create those many new features in the dataset. If we have the original features as ' f_1 ', ' f_2 ', ' f_3 ' and ' f_4 ', then if we have missing values only in ' f_2 ' and ' f_3 ' across the dataset, we then have to create only ' f_2' ' and ' f_3 '. If we have the missing values in all the features, then we have to create ' f_1' ', ' f_2' ', ' f_3 ' and ' f_4 '.

For a point ' x_i ', if a feature ' f_d ' is having a missing value, then only ' f_d' ' has to be filled with 1, and all new feature components should be 0. This is how we have to fill all the values in the data matrix, and then should go for model

building. This is because the missing values also could be a source of information.

4) Model based Imputation



Here we are taking the ' f_3 ' column as the output variable, and all the data points whose ' f_3 ' value is known are taken into the training dataset, and all those data points whose ' f_3 ' value is unknown are taken into the test dataset.

Now the model gets trained using the training data, and will predict the values of ' f_3 ' for the test data. The missing data values are replaced by the predictions obtained.

Here if ' f_3 ' is a categorical variable, we go with classification and if ' f_3 ' is a numerical variable, we go with regression. This is known as **Model based Imputation**.

The model based imputors take a lot of time when compared to the previous approaches.

As K-NN works on the concept of the neighborhood, if the points ' x_1 ', ' x_2 ', ' x_3 ' and ' x_4 ' are similar, then all features of those data points are also similar feature value wise.

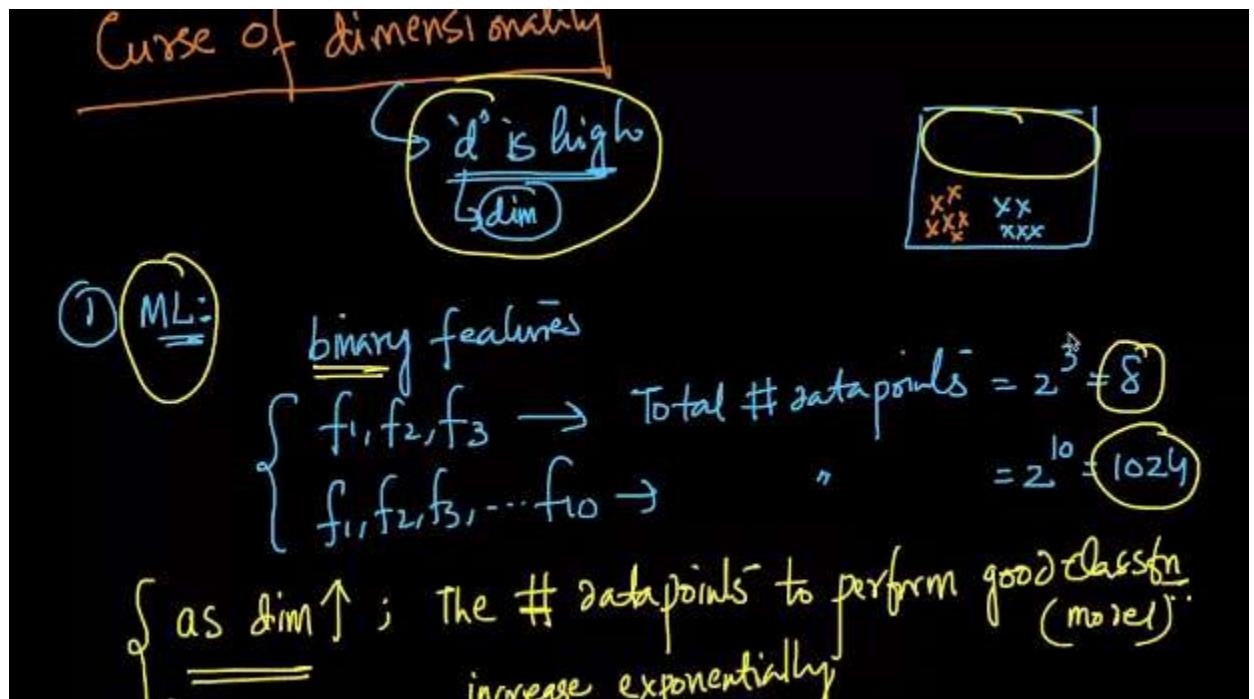
31.17 Curse of Dimensionality

The curse of dimensionality is a concept that explains a bunch of things that happen when our dimensionality in the given problem is very high. The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high dimensional space.

1) Machine Learning

Let us assume we have the binary features. (Each feature can take either 0 or 1) If we have 3 features(ie., ' f_1 ', ' f_2 ' and ' f_3 ') then total number of possible data points = $2^3 = 8$

If we have 10 features(ie., ' f_1 ', ' f_2 ', ' f_3 ', , ' f_{10} ') then total number of possible data points = $2^{10} = 1024$.



If there are no data points in the region, then we do not know what is happening in that region, and also we do not know to which class the points in that region belong to.

As the dimensionality increases, the number of data points needed for the model to perform well increases exponentially.

Hughes Phenomenon: If the size of the dataset is fixed, then the model performance decreases, as the dimensionality increases.

2) Distance Functions (Especially Euclidean Distance)

The intuition of distance in 3-D is not valid in high dimensional space. Let us assume we are working in 1-D space and we have 'n' random points.

(2) Distance functions (euclidean dist)

Curse of Dim:- intuition of dist in 3D
is not valid in high dim spaces

1D-world \rightarrow n-random pts

$$\left\{ \begin{array}{l} \text{dist_min}(x_i) = \min_{x_j \neq x_i} \left\{ \text{dist}^{\text{euc}}(x_i, x_j) \right\} \\ \text{dist_max}(x_i) = \max_{x_j \neq x_i} \left\{ \text{dist}^{\text{euc}}(x_i, x_j) \right\} \end{array} \right.$$

$$\text{dist_min}(x_i) = \min_{x_j \neq x_i} \{ \text{dist}(x_i, x_j) \} \rightarrow (1)$$

$$\text{dist_max}(x_i) = \max_{x_j \neq x_i} \{ \text{dist}(x_i, x_j) \} \rightarrow (2)$$

$$(\text{dist_max}(x_i) - \text{dist_min}(x_i)) / \text{dist_min}(x_i) > 0 \rightarrow (3)$$

(3) holds good only when d = 1 (or) 2 (or) 3

As the dimensionality increases, the above term (ie., (3)) tends to become zero.

$$\text{Lt}_{d \rightarrow \infty} (\text{dist_max}(x_i) - \text{dist_min}(x_i)) / \text{dist_min}(x_i) > 0$$

In this case, $\text{dist_max}(x_i) \approx \text{dist_min}(x_i)$

So in the high-dimensional case, $\text{dist_max}(x_i) \approx \text{dist_min}(x_i)$

It means, every pair of points are actually equidistant from each other. In high-dimensional space, euclidean distance doesn't make any logical sense. As K-NN relies on distance, especially euclidean distance doesn't make any logical sense in higher dimensions. So ultimately K-NN doesn't work well in higher dimensions.

Solution to this problem

In order to build a K-NN model on higher dimensions, instead of using euclidean distance, we have to use cosine similarity.

Even cosine similarity also gets affected by the higher dimensional space, but very less when compared to that of euclidean distance.

Note: Techniques like BOW, TF-IDF, etc result in high dimensions. Hence it is recommended to use cosine-similarity when working on such data.

High Dimensional Space & Dense Matrix/Vector → Impact/Curse of Dimensionality is very high

High Dimensional Space & Sparse Matrix/Vector → Impact/Curse of Dimensionality is very low

The expression (3) holds good only if the data is in the form of a sparse vector/matrix as the data points are not distributed randomly in a sparse vector. That too it holds good only for euclidean distance.

3) Overfitting and Underfitting

Let us assume we use only K-NN for the time being to build a model. If the number of dimensions (d) increases, then overfitting also increases.

If the dimensionality is very low, then the underfitting increases.

Solution to be attempted in case of higher dimensions

- 1) We can go for the feature selection techniques to reduce the number of dimensions. But the feature selection techniques take the class labels into consideration.
- 2) We can go for dimensionality reduction techniques, which do not use the class labels, to reduce the number of dimensions.
- 3) When we are using K-NN on the text data, in order to reduce the impact of higher dimensionality, we can
 - a) use cosine similarity instead of euclidean distance.
 - b) use sparse representation of the matrix instead of dense representation of the matrix.

31.18 Bias-Variance Tradeoff

Bias-Variance Tradeoff is an important concept in the theory of Machine Learning. Bias-Variance Tradeoff gives the idea of mathematically analyzing overfitting and underfitting.

Generalization Error

Generalization Error is the error that is made on the future unseen data. Generalization Error is composed of three terms.

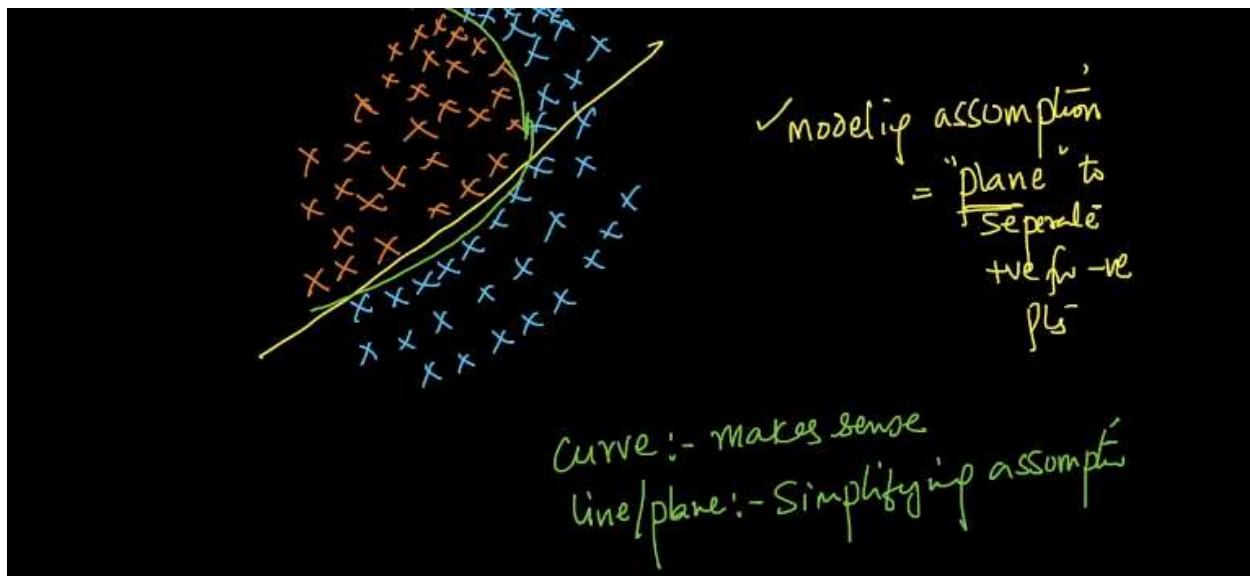
$$\text{Generalization Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

The main goal of Machine Learning is to keep the Generalization Error as low as possible. The irreducible error is the error that could not be reduced further, as there is no model that is perfect at predictions.

So if we want to reduce the generalization error, we have to reduce the Bias and the Variance.

Bias

Bias is the error occurred due to simplifying assumptions. High Bias means Underfitting. Let us assume we have the data points as shown below.



Modelling Assumption: We can use only a plane to separate the '+ve' and the '-ve' points.

So practically, a linear plane could not perfectly separate the '+ve' and the '-ve' classes. It leads to misclassifications.

Whereas a curve can perfectly classify both the classes and separates them. Here if we limit ourselves only to use lines/planes, then we are making simplifying assumptions about the separation of the two classes. The error occurred due to such simplifying assumptions is called **Bias**.

Let us assume we have 'n' data points in the training dataset with 80% of the points belonging to the -ve class and the remaining 20% of the points belonging to the +ve class.

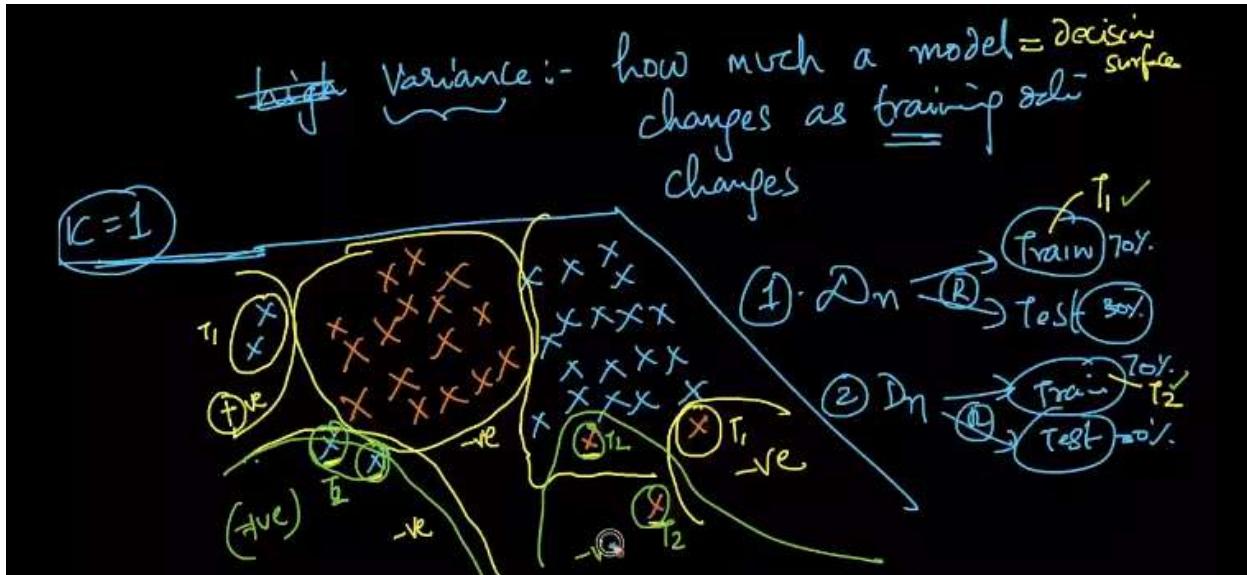
If we are using the K-NN model, with **K=n**, then each and every query point ' x_q ' will be classified as -ve, as the given dataset is an imbalanced one. Here the dominant class will be the class label for any newly arrived query point. We made an assumption that the points belonging to different classes can be separated using a linear surface, but here there's no separation at all, as all the points are being classified as -ve. In such a case, we end up having high bias, and this scenario is known as **Underfitting**.

If $K=n$, then we say the model is having high bias, and is underfitting.

Variance

Variance is the measure of how much a model changes as the training data changes.

Let us assume we have the data points as shown below



Let us assume we have the dataset ' D_n ' and we are splitting it randomly into ' D_{Train} '(70%) and ' D_{Test} '(30%). So 70% of the points come into ' D_{Train} ' and the remaining 30% of them go into ' D_{Test} '.

If we combine both the sets, and split them randomly again into ' D_{Train} '(70%) and ' D_{Test} '(30%). This time we get slightly a different set of points into ' D_{Train} ' when compared to the ' D_{Train} ' obtained in the previous split.

When we generate different training datasets, the small changes in the training datasets result in very different models. (ie., The changes are the data points in one training set that do not occur in the other training sets. This leads to different decision boundaries for different training datasets)

Variance is the measure of how a model changes as the training data changes. So in K-NN, for the lower values of 'K' (ie., K=1), there will be large changes in the decision boundaries, for small changes in the training dataset. We call this High Variance (or) Overfitting and such a model is known as a High Variance model (or) Overfitting model.

As our main goal is to reduce the generalization error, in order to get it done, we have to reduce the bias and the variance (as it is not possible to reduce the irreducible error).

$$\text{Generalization Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Reducing the Bias in our model means ensuring there is **no underfitting**.
Reducing the Variance in our model means ensuring there is **no overfitting**.

Example

K-value	Bias	Variance	Irreducible Error	→	Generalization Error
K=1	10	100	3	→	113 (overfit)
·	·	·	·	·	·
K=5	12	10	3	→	25 (best-fit)
·	·	·	·	·	·
K=n	100	2	3	→	105 (underfit)

The best solution occurs if the 'K' value is in between 1 and 'n'.

High Bias, Low Variance → Underfitting

Low Bias, High Variance → Overfitting

As the 'K' value tends to increase towards 'n', then the K-NN model underfits.

As the 'K' value tends to decrease towards '1', then the K-NN model overfits.

31.19 Intuitive Understanding of Bias-Variance

Let us assume we have a dataset 'D' that is split into ' D_{Train} ' and ' D_{Test} '. We build a model using ' D_{Train} ', and compute two types of errors. They are the **Training Error** and the **Test Error**.

Training Error = difference(y_i, \hat{y}_i) on D_{Train}

Test Error = difference(y_i, \hat{y}_i) on D_{Test}

1) When do we say the model has High Bias (or) the model Underfits

If the Training Error is high, then the bias is also high in the model. It means the model underfits.

In K-NN, when $K=n$, the Bias is high and we say the model underfits. If the Training Error is low, then the Bias in the model is also low.

2) When do we say the model has High Variance (or) the model Overfits

a) If the Training Error is low and the Test Error is high, then the model is an overfit model, and the variance in the model is high.

b) If the Training data changes slightly, and the model performance changes drastically due to the change in the Training data, then we can say the model has high variance.

In K-NN, if a single point changes in the Training data, and if it leads to severe changes in the model performance, even then we say the model overfits, having high variance.

Note: As the section 31.20 is on the revision question, we are not giving any notes for them. You can find the links on the web page itself.

31.21 Best and Worst case of an algorithm

1) High Dimensionality

If the dimensionality of the data is small(say $d < 10$), then K-NN is the best algorithm to go for.

If the dimensionality is high, then the model will face the problem of curse of dimensionality(especially when we use Euclidean Distance as the distance metric)

If the dimensionality is high, the model interpretability also reduces and also the runtime complexity of K-NN/KD-Tree/LSH increases.

2) Low Latency

Low Latency Systems are the systems that give the results faster. (Example: Search Engines)

K-NN should never be used in Low Latency Systems, as the runtime complexity of K-NN is very high. Even if there is a requirement to use K-NN, we should prefer either KD-Tree or LSH.

Order of runtime complexities: LSH>KD-Tree>K-NN

- 3) If we can find the appropriate distance measure, then it is good to apply K-NN.
If someone gives us the similarity matrix (or) distance measure (or) the formula to compute the distance, then K-NN is better.
But in cases, when we do not know which distance measure has to be used, then K-NN will not be a good choice, irrespective of the context you are using.

In this chapter we will look at an algorithm called Naive Bayes.

32.1 Naive Bayes Algorithm

Probabilistic model [edit]

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities $p(C_k | x_1, \dots, x_n)$ for each of K possible outcomes or classes C_k .^[7]

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

Timestamp 0:57

Naive Bayes algorithm is based on the Bayes Theorem. Bayes theorem plays a very crucial rule in the world of statistics, in fact there is one entire branch called Bayesian Statistics.

Probabilistic model [edit]

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities $p(C_k | x_1, \dots, x_n)$ for each of K possible outcomes or classes C_k .^[7]

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

Timestamp 2:28

In this chapter we will discuss the theory behind this algorithm.

Say we have a datapoint x , we can represent it in vector form as $\langle x_1, x_2, \dots, x_n \rangle$, here x contains n features, each datapoint can belong to one of the K classes. So, we are dealing with a multi class classification problem, if $K = 2$ we have a binary classification problem.

The screenshot shows a web browser window with the URL https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=111911111. The page content discusses the probabilistic model of Naive Bayes. It starts by defining the model as a conditional probability given a feature vector $\mathbf{x} = (x_1, \dots, x_n)$. The formula is $p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$. A handwritten-style annotation highlights the numerator $p(C_k) p(\mathbf{x} | C_k)$ with a red circle and a bracket. Below this, it says "In plain English, using Bayesian probability terminology, the above equation can be written as posterior = prior × likelihood / evidence". The page also notes that the denominator $p(\mathbf{x})$ is constant in practice. The browser's address bar, toolbar, and status bar are visible at the top and bottom of the screenshot.

Timestamp 4:25

Now we want to find out $p(C_k | x)$, it means that given a feature x , what is the probability that the feature x belongs to class(C) k , denoted by subscript. So, we can find this probability for all classes from 1 to k , and say that the point belongs to that class which has the maximum probability.

Now, using Bayes theorem the conditional probability can be written as

$$p(C_k | x) = (p(C_k) \times p(x | C_k)) / p(x)$$

In Bayesian terminology we have,

$$\text{Posterior} = (\text{prior} \times \text{likelihood}) / (\text{evidence})$$

Where,

$$\text{Posterior} = p(C_k | x)$$

$$\text{Prior} = p(C_k)$$

$$\text{Likelihood} = p(x | C_k)$$

$$\text{Evidence} = p(x)$$

$\rightarrow \text{class } 3$

$p(C_1|x)$
 $p(C_2|x)$
 $p(C_3|x)$ *target*
 $p(C_k|x)$

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})} \rightarrow p(C_k) p(x|C_k) = p(x \cap C_k)$$

In plain English, using Bayesian probability terminology, the above equation can be written as

posterior = $\frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \\ &= \dots \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C . This means that

Timestamp 6:18

So our main task is to calculate the probabilities for each class and determine which one is the maximum. Now, if we carefully notice the denominator, it is the same for all the classes as shown in the image, so for determining the maximum we don't need the denominator, we can only focus on the numerator.

The numerator part is $\Rightarrow p(C_k) \times p(x | C_k)$, this term calculates the probability of an event where both C_k and x occurs. So, we can simply write this as $p(x \cap C_k)$.

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | x) = \frac{p(C_k) p(x | C_k)}{p(x)}$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3 | x_4, \dots, x_n, C_k) \\ &\quad \vdots \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C . This means that

Timestamp 9:29

Now, $p(C_k | x) \propto p(x \cap C_k)$. The proportionality comes because we have ignored the denominator.

$p(x \cap C_k)$ can also be written as $p(C_k, x)$. So whichever class possesses the maximum value we can select that class as our prediction. Here we are calculating the probability of two events, this is often called a joint probability model.

$p(C_k, x)$ can be written as $p(C_k, x_1, x_2, \dots, x_n)$, here we have just written x in its components of n - features/dimensions.

$$p(x_1, x_2, \dots, x_n, C_k) = p(x_1 | x_2, \dots, x_n, C_k) * p(x_2, \dots, x_n, C_k)$$

$$\downarrow p(A, B) = p(A|B) * p(B)$$

$$=$$

$$p(A|B) = \frac{p(A, B)}{p(B)}$$

defn. of cond. prob.
Bayes thm.

Timestamp 12:21

Our task has boiled down to finding this => $p(C_k, x_1, x_2, \dots, x_n)$.

Here we will use something known as the chain rule of conditional probability, where we use the conditional probability recursively.

Conditional probability for an event A given B, is given by => $p(A | B) = p(A \cap B) / p(B)$. Also $p(A \cap B) = p(A | B) * p(B)$

We will apply this simple formula recursively on this term => $p(C_k, x_1, x_2, \dots, x_n)$.

Now,

$$p(C_k, x_1, x_2, \dots, x_n) = p(x_1, x_2, \dots, x_n, C_k) \quad [\text{we can exchange terms because } A \cap B = B \cap A]$$

Now, we can break our elements into two components say $x_1 = A$ and $x_1, x_2, \dots, x_n, C_k = B$, and apply conditional probability.

So, we have

$$p(x_1, x_2, \dots, x_n, C_k) = p(x_1 | x_2, \dots, x_n, C_k) * p(x_2, \dots, x_n, C_k)$$

values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned}
 p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\
 &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \\
 &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \\
 &= \dots \\
 &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k)
 \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C . This means that

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k).$$

Thus, the joint model can be expressed as

$$\begin{aligned}
 p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\
 &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\
 &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k).
 \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k | x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

Timestamp 15:45

We will continue applying conditional probability recursively.

This is what we get after the first expansion.

$$p(x_1, x_2, \dots, x_n, C_k) = p(x_1 | x_2, \dots, x_n, C_k) \times p(x_2, \dots, x_n, C_k)$$

Now we will keep the first part $\Rightarrow p(x_1 | x_2, \dots, x_n, C_k)$ as it is and expand the second one $\Rightarrow p(x_2, \dots, x_n, C_k)$, as shown in the image above.

Continuing like this, the final term that we get is,

$$= p(x_1 | x_2, \dots, x_n, C_k) \times p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) \times p(x_n | C_k) \times p(C_k)$$

Now we need to solve for components of this probability terms from the data.

values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model:

$$p(C_k, x_1, \dots, x_n) = p(x_1, \dots, x_n, C_k)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3 | x_4, \dots, x_n, C_k) p(x_4 | x_5, \dots, x_n, C_k) \\ &\dots \end{aligned}$$

Note: the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C . This means that

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k)$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k | x_1, \dots, x_n) = \frac{1}{C} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

Timestamp 18:00

Now, finding these individual terms is very difficult because we have to count those data points for which our condition matches exactly. For that we need lots of data, rather lots of combinations of data, which may not be always possible.

Naive Bayes

values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model:

$$p(C_k, x_1, \dots, x_n) = p(x_1, \dots, x_n, C_k)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3 | x_4, \dots, x_n, C_k) \\ &= \dots \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k) \end{aligned}$$

Now the "Naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C . This means that

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k).$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

P(A|B) = P(A)

P(A|B,C) = P(A|C)

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k | x_1, \dots, x_n) = \frac{1}{C} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

Timestamp 20:05

In order to overcome this problem of finding an exact match we make a naive assumption. That's why this is called Naive Bayes. The assumption is that of conditional independence of the features.

Two events are said to be independent if the occurrence of one doesn't affect the other.

$$p(A|B) = p(A)$$

Similarly conditional independence can be written as

$$p(A|B, C) = p(A|C),$$

where $p(A|B, C)$ is the probability of A given both B and C. Since the probability of A given C is the same as the probability of A given both B and C, this equality expresses that B contributes nothing to the certainty of A. In this case, A and B are said to be conditionally independent given C.

values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \dots, x_n) = p(x_1, \dots, x_n, C_k)$$

$$= p(x_1 | x_2, \dots, x_n, C_k)p(x_2, \dots, x_n, C_k)$$

$$= p(x_1 | x_2, \dots, x_n, C_k)p(x_2 | x_3, \dots, x_n, C_k)p(x_3, \dots, x_n, C_k)$$

$$= \dots$$

$$= p(x_1 | x_2, \dots, x_n, C_k)p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k)p(x_n | C_k)p(C_k)$$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C . This means that

$\checkmark p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k) \Rightarrow (x_i \text{ is indep of } (x_{i+1}, \dots, x_n) \text{ given } C_k)$

Thus, the joint model can be expressed as

$$p(C_k | x_1, \dots, x_n) \propto p(C_k, x_1, \dots, x_n)$$

$$\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots$$

$$\propto p(C_k) \prod_{i=1}^n p(x_i | C_k).$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k | x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

Timestamp 21:44

So our naive bayes assumption is that feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the class k .

Now this idea will be used for every term in our probability. In general,

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k)$$

Here x_i is conditionally independent of $x_{i+1}, x_{i+2}, \dots, x_n$ given C_k . This vastly simplifies our calculation.

values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model:

$$p(C_k, x_1, \dots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \end{aligned}$$

$p(x_1 | C_k)$

$p(x_2 | C_k)$

$p(x_3 | C_k)$

\vdots

$p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k)$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C . This means that

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k).$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots p(x_n | C_k) \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

$p(C_k | C_k)$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k | x_1, \dots, x_n) = \frac{1}{\sum_{k=1}^n p(C_k)} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

Timestamp 23:00

Now the conditional independence can be applied to each of the terms. After following the steps as shown in the image we reach to our final probability, which is

$$p(C_k, x_1, x_2, \dots, x_n) \propto p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

The proportionality comes because we are ignoring the denominator. It is a product of individual terms along with the probability of the class.

$$\propto p(C_k) p(x_1 \mid C_k) p(x_2 \mid C_k) p(x_3 \mid C_k) \cdots$$

$$\propto p(C_k) \prod_{i=1}^n p(x_i \mid C_k).$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k \mid x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i \mid C_k)$$

where the evidence $Z = p(\mathbf{x}) = \sum_k p(C_k) p(\mathbf{x} \mid C_k)$ is a scaling factor dependent only on x_1, \dots, x_n , that is, a constant if the values of the feature variables are known.

Constructing a classifier from the probability model [edit]

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or MAP decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label $\hat{y} = C_k$ for some k as follows:

$$\checkmark \hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} p(C_k) \prod_{i=1}^n p(x_i \mid C_k).$$

Parameter estimation and event models [edit]

A class's prior may be calculated by assuming equiprobable classes (i.e., priors = 1 / (number of classes)), or by calculating an

Timestamp 25:28

If we want an exact term, we can remove the proportionality and bring in the denominator as shown in the image.

Now for a classifier we will find the probability for each of the classes, and select that class which gives the maximum probability.

$$\hat{y} = \operatorname{argmax}_{k \in \{1, \dots, K\}} p(C_k) \prod_{i=1}^n p(x_i \mid C_k)$$

This rule is known as *maximum a posteriori* or MAP decision rule, as we are selecting the argmax of the maximum posterior.

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

32.2 Toy example: Train and test Stages

<http://shatterline.com/blog/2013/09/12/not-so-naive-classification-with-the-naive-bayes-classifier/>

	Predictors				Response
	Outlook	Temperature	Humidity	Wind	Class Yes/No
Day1	Sunny	Hot	High	Weak	Play/No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

The Learning Phase

In the learning phase, we compute the table of likelihoods (probabilities) from the training data. They are:

- $P(\text{Outlook} = \alpha | \text{Class} = \text{Yes})$, where $\alpha \in \{\text{Sunny}, \text{Overcast}, \text{Rain}\}$ and $b \in \{\text{yes}, \text{no}\}$.
- $P(\text{Temperature} = \beta | \text{Class} = \text{Yes})$, where $\beta \in \{\text{Hot}, \text{Mild}, \text{Cool}\}$ and $b \in \{\text{yes}, \text{no}\}$.
- $P(\text{Humidity} = \gamma | \text{Class} = \text{Yes})$, where $\gamma \in \{\text{High}, \text{Normal}\}$ and $b \in \{\text{yes}, \text{no}\}$.
- $P(\text{Wind} = w | \text{Class} = \text{Yes})$, where $w \in \{\text{Weak}, \text{Strong}\}$ and $b \in \{\text{yes}, \text{no}\}$.

Timestamp 2:28

In this chapter we will look at a toy dataset and apply naive bayes algorithm on this dataset. It's a binary classification problem with target variable play, $y_i \in \{\text{Yes}(Y), \text{No}(N)\}$. Our dataset contains 4 features => Outlook(f_1), Temperature(f_2), Humidity(f_3), Wind(f_4). All these features are categorical features, naive bayes can also be used for continuous features but it is easy to visualize for categorical data. So we will stick to categorical data.

Outlook(f_1) can have 3 categories: sunny, overcast, rain. Temperature(f_2) can have 3 categories: hot, mild, cool. Humidity(f_3) can have 2 categories: high, normal. Wind(f_4) can have 2 categories: strong, weak.

Here we are trying to predict whether it is suitable to play tennis outside given various weather parameters.

The class label is the variable, Play and takes the values yes or no.

Play: [Yes, No]

We read-in training data below that has been collected over 14 days.

	Predictors				Response
	Outlook	Temperature	Humidity	Wind	Class
					Play=Yes Play=No
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes

MAP

$p(\text{class} = \text{play} = \text{yes} | x_q)$

$\checkmark p(\text{play} = \text{no} | x_q)$

$x_q \rightarrow y_q = (\text{play} = \text{yes})$

if

Timestamp 4:32

So, here we need to calculate two probabilities,
 $p(\text{Play} = \text{Yes} | x_q)$, it says what is the probability that we should outside play outside given weather params x_q . Similarly $p(\text{Play} = \text{No} | x_q)$, what is the probability that we should not play outside given weather params?

Out of both these terms whichever term possesses the highest value we declare that to be our class label, MAP rule.

The screenshot shows a Google Doc with a table of weather data and handwritten notes for Naive Bayes classification.

Table Data:

	Predictors				Response
	Outlook	Temperature	Humidity	Wind	
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Handwritten Notes:

- $p(C = \text{No}) = 5/14$
- f_1, f_2, f_3, f_4 (labeled as features)
- $p(C | f_1, f_2, f_3, f_4) \propto p(f_1 | C) * p(f_2 | C) * p(f_3 | C) * p(f_4 | C) * p(C)$
- $p(C | f_1, f_2, f_3, f_4) \propto p(f_1 | C) * p(f_2 | C) * p(f_3 | C) * p(f_4 | C) * p(C)$ (repeated)
- $p(C | f_1, f_2, f_3, f_4) \propto p(f_1 | C) * p(f_2 | C) * p(f_3 | C) * p(f_4 | C) * p(C)$ (repeated)
- $p(C | f_1, f_2, f_3, f_4) \propto p(f_1 | C) * p(f_2 | C) * p(f_3 | C) * p(f_4 | C) * p(C)$ (repeated)
- $p(C = \text{Yes}) = 9/14$

Timestamp 8:02

As per Naive Bayes,

$$p(C | f_1, f_2, f_3, f_4) \propto p(f_1 | C) * p(f_2 | C) * p(f_3 | C) * p(f_4 | C) * p(C)$$

We have written the above term directly from Naive Bayes where f_i 's are our features and C is our class.

$p(C = \text{Yes}) = 9/14$, we have 9 yes out of 14 data points.

$p(C = \text{No}) = 5/14$, we have 5 no out of 14 data points.

This are called prior probabilities.

The screenshot shows a presentation slide with the following content:

Train

In the learning phase, we compute the table of likelihoods (probabilities) from the training data. They are:

- $P(\text{Outlook} = o | \text{Class}_{\text{play}} = b)$, where $o \in [\text{Sunny, Overcast, Rainy}]$ and $b \in [\text{yes, no}]$
- $P(\text{Temperature} = t | \text{Class}_{\text{play}} = b)$, where $t \in [\text{Hot, Mild, Cool}]$ and $b \in [\text{yes, no}]$,
- $P(\text{Humidity} = h | \text{Class}_{\text{play}} = b)$, where $h \in [\text{High, Normal}]$ and $b \in [\text{yes, no}]$,
- $P(\text{Wind} = w | \text{Class}_{\text{play}} = b)$, where $w \in [\text{Weak, Strong}]$ and $b \in [\text{yes, no}]$.

Handwritten notes include circled terms like $p(f_i | c)$ and $p(\text{outlook} = \text{sunny} | C = \text{Yes}) = 2/9$.

$P(\text{Outlook} = o \text{Class}_{\text{play}} = \text{Yes/No})$	Frequency		Probability in Class	
Outlook =	Play=Yes	Play=No	Play=Yes	Play=No
→ Sunny	2	3	2/9	3/5
→ Overcast	4	0	4/9	0/5
→ Rain	3	2	3/9	2/5
	total 9	total 5		

$P(\text{Temperature} = t \text{Class}_{\text{play}} = \text{Yes/No})$	Frequency		Probability in Class	
Temperature =	Play=Yes	Play=No	Play=Yes	Play=No

Timestamp 12:33

Now we need to calculate the feature probabilities(likelihood), for calculating these we will use the train data. This phase is called the learning/training phase.

Let's calculate one of the terms => $p(\text{Outlook} = \text{sunny} | C = \text{Yes})$

Now here we need to find the probability of an event where outlook = sunny given $C = \text{yes}$. So, at first we will count all the data points which have class Yes and out of those data points we need to count those features where outlook = sunny.

So, after counting we got $p(\text{Outlook} = \text{sunny} | C = \text{Yes}) = 2/9$, Out of 9 data points which have class label = yes, 2 of them had an outlook = sunny.

Similarly $p(\text{Outlook} = \text{sunny} | C = \text{No}) = 3/5$, Out of 5 data points which have class label = No, 3 of them had an outlook = sunny.

We need to calculate the likelihood of each of the features, and store it in some data structure. As shown in the image above. Please follow the shatterline link to get a complete list of all the probabilities.

$P(F|C)$

n

c

Training ph:

$\rightarrow \text{likelihood prob}$

$\rightarrow P(\text{class})$

$O(ndc)$

$O(nd)$

	Predictors				Response		
	Outlook	Temperature	Humidity	Wind	Class	Play=Yes	Play=No
Day1	Sunny	Hot	High	Weak	No		
Day2	Sunny	Hot	High	Strong	No		
Day3	Overcast	Hot	High	Weak	Yes		
Day4	Rain	Mild	High	Weak	Yes		
Day5	Rain	Cool	Normal	Weak	Yes		
Day6	Rain	Cool	Normal	Strong	No		
Day7	Overcast	Cool	Normal	Strong	Yes		
Day8	Sunny	Mild	High	Weak	No		
Day9	Sunny	Cool	Normal	Weak	Yes		
Day10	Rain	Mild	Normal	Weak	Yes		
Day11	Sunny	Mild	Normal	Strong	Yes		
Day12	Overcast	Mild	High	Strong	Yes		
Day13	Overcast	Hot	Normal	Weak	Yes		
Day14	Rain	Mild	High	Strong	No		

The Learning Phase

Timestamp 15:19

Let's digress for a minute and calculate the complexities.

Time Complexity

Say we have n data points in our dataset and each data point has d dimension. Also say we have c classes. Then if we use a brute force algorithm our time complexity is $O(ndc)$.

This is because for calculating the individual probabilities we need to go through each feature i.e. d , n times because we have n data points. This we need to do for c classes.

If we use some optimization, time complexity can be reduced to $O(n)$. if d is small. Otherwise it is $O(nd)$.

Contents - Google Docs Shutterline Blog - Home Native Bayes classifier - W1

shatterline.com/blog/2013/08/12/not-an-naive-classification-with-the-naive-bayes-classifier/

The Learning Phase

In the learning phase, we build a table of training data.

	Predictors				Response
	Outlook	Temperature	Humidity	Wind	Class Play=Yes Play=No
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Timestamp 17:00

Space Complexity is $O(dc)$, because we need to store the probabilities of each feature i.e. d , c times i.e. no of classes. Also we need to store class probabilities.

Contents - Google Docs Shutterline Blog - Home Native Bayes classifier - W1

shatterline.com/blog/2013/08/12/not-an-naive-classification-with-the-naive-bayes-classifier/

Temperature=Cool, Humidity=High, Wind=Strong) that will have to be classified (i.e., are we going to play tennis under the conditions specified by x').

With the MAP rule, we compute the posterior probabilities. This is easily done by looking up the tables we built in the learning phase.

$$\begin{aligned} P(\text{Class}_{\text{Play=Yes}}|x') &= [P(\text{Sunny}|\text{Class}_{\text{Play=Yes}}) \times P(\text{Cool}|\text{Class}_{\text{Play=Yes}}) \times \\ &\quad P(\text{High}|\text{Class}_{\text{Play=Yes}}) \times P(\text{Strong}|\text{Class}_{\text{Play=Yes}})] \times \\ &\quad P(\text{Class}_{\text{Play=Yes}}) \\ &= 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053 \end{aligned}$$

$$\begin{aligned} P(\text{Class}_{\text{Play=No}}|x') &= [P(\text{Sunny}|\text{Class}_{\text{Play=No}}) \times P(\text{Cool}|\text{Class}_{\text{Play=No}}) \times \\ &\quad P(\text{High}|\text{Class}_{\text{Play=No}}) \times P(\text{Strong}|\text{Class}_{\text{Play=No}})] \times \\ &\quad P(\text{Class}_{\text{Play=No}}) \\ &= 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0205 \end{aligned}$$

Since $P(\text{Class}_{\text{Play=Yes}}|x')$ less than $P(\text{Class}_{\text{Play=No}}|x')$, we classify the new instance x' to be Play=No .

Timestamp 18:30

Now, given a data point during test time we need to classify the data point. Say we have a data point $x_q = (\text{sunny, cool, high, strong})$, here we are given values for the 4 features. We need to calculate two posteriors, because we have two classes.

$$p(\text{class} = \text{Yes} | x_q) \text{ and } p(\text{class} = \text{No} | x_q).$$

We can write these posteriors in terms of their features as shown in the image.

The screenshot shows a Google Docs spreadsheet with handwritten annotations. The annotations explain the calculation of posteriors for a new instance x' (Cool, High, Strong) based on training data. A dictionary is used to store feature combinations and their probabilities. The Python code shown uses a dictionary to get the probability of a specific combination like 'Sunny, Yes'.

Handwritten Notes:

- Dictionary - python:** A table showing probability calculations for different feature combinations.
- Calculation of $P(\text{ClassPlay}=\text{Yes}|x')$:**

$$P(\text{ClassPlay}=\text{Yes}|x') = [P(\text{Sunny}|\text{ClassPlay}=\text{Yes}) \times P(\text{Cool}|\text{ClassPlay}=\text{Yes}) \times P(\text{High}|\text{ClassPlay}=\text{Yes}) \times P(\text{Strong}|\text{ClassPlay}=\text{Yes})] \times P(\text{ClassPlay}=\text{Yes})$$

$$= 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$
- Calculation of $P(\text{ClassPlay}=\text{No}|x')$:**

$$P(\text{ClassPlay}=\text{No}|x') = [P(\text{Sunny}|\text{ClassPlay}=\text{No}) \times P(\text{Cool}|\text{ClassPlay}=\text{No}) \times P(\text{High}|\text{ClassPlay}=\text{No}) \times P(\text{Strong}|\text{ClassPlay}=\text{No})] \times P(\text{ClassPlay}=\text{No})$$

$$= 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0205$$
- Classification Decision:** Since $P(\text{ClassPlay}=\text{Yes}|x') < P(\text{ClassPlay}=\text{No}|x')$, the new instance x' is classified as No.

Timestamp 20:02

For individual terms we need to use the probabilities that we have calculated during the training phase. For example, one of the feature is $p(\text{Outlook} = \text{Sunny} | \text{class} = \text{Yes})$, so during training we can build a dictionary with feature combinations as key and the probabilities as value. We can have a key $\langle \text{Sunny}, \text{Yes} \rangle$ and its probability i.e. $2/9$ as its value. We can use python's inbuilt dictionary methods to implement these.

Temperature=Cool, Humidity=High, Wind=Strong) that will have to be classified (i.e., are we going to play tennis under the conditions specified by x').

With the MAP rule, we compute the posterior probabilities. This is easily done by looking up the tables we built in the learning phase.

$$P(\text{Class}_{\text{Play}}=\text{Yes}|x') \propto [P(\text{Sunny}|\text{Class}_{\text{Play}}=\text{Yes}) \times P(\text{Cool}|\text{Class}_{\text{Play}}=\text{Yes}) \times P(\text{High}|\text{Class}_{\text{Play}}=\text{Yes}) \times P(\text{Strong}|\text{Class}_{\text{Play}}=\text{Yes})] \times P(\text{Class}_{\text{Play}}=\text{Yes})$$

$$\propto 2/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

$$P(\text{Class}_{\text{Play}}=\text{No}|x') = [P(\text{Sunny}|\text{Class}_{\text{Play}}=\text{No}) \times P(\text{Cool}|\text{Class}_{\text{Play}}=\text{No}) \times P(\text{High}|\text{Class}_{\text{Play}}=\text{No}) \times P(\text{Strong}|\text{Class}_{\text{Play}}=\text{No})] \times P(\text{Class}_{\text{Play}}=\text{No})$$

$$= 3/5 \times 1/5 \times 4/5 \times 5/14 = 0.0205$$

Since $P(\text{Class}_{\text{Play}}=\text{Yes}|x')$ less than $P(\text{Class}_{\text{play}}=\text{No}|x')$, we classify the new instance x' to No.

Timestamp 21:24

After putting all the likelihood values and priors in we get,

$$p(\text{class} = \text{Yes} | x_q) \approx 0.0053$$

$$p(\text{class} = \text{No} | x_q) \approx 0.0205$$

So as per MAP rule we can say that our class instance is No.

Classification Phase

Let's say, we get a new instance of the weather condition, $x'=(\text{Outlook}=\text{Sunny}, \text{Temperature}=\text{Cool}, \text{Humidity}=\text{High}, \text{Wind}=\text{Strong})$ that will have to be classified (i.e., are we going to play tennis under the conditions specified by x').

With the MAP rule, we compute the posterior probabilities. This is easily done by looking up the tables we built in the learning phase.

$$P(\text{Class}_{\text{Play}}=\text{Yes}|x') = [P(\text{Sunny}|\text{Class}_{\text{Play}}=\text{Yes}) \times P(\text{Cool}|\text{Class}_{\text{Play}}=\text{Yes}) \times P(\text{High}|\text{Class}_{\text{Play}}=\text{Yes}) \times P(\text{Strong}|\text{Class}_{\text{Play}}=\text{Yes})] \times P(\text{Class}_{\text{Play}}=\text{Yes})$$

$$= 2/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

$$P(\text{Class}_{\text{Play}}=\text{No}|x') = [P(\text{Sunny}|\text{Class}_{\text{Play}}=\text{No}) \times P(\text{Cool}|\text{Class}_{\text{Play}}=\text{No}) \times P(\text{High}|\text{Class}_{\text{Play}}=\text{No}) \times P(\text{Strong}|\text{Class}_{\text{Play}}=\text{No})] \times P(\text{Class}_{\text{Play}}=\text{No})$$

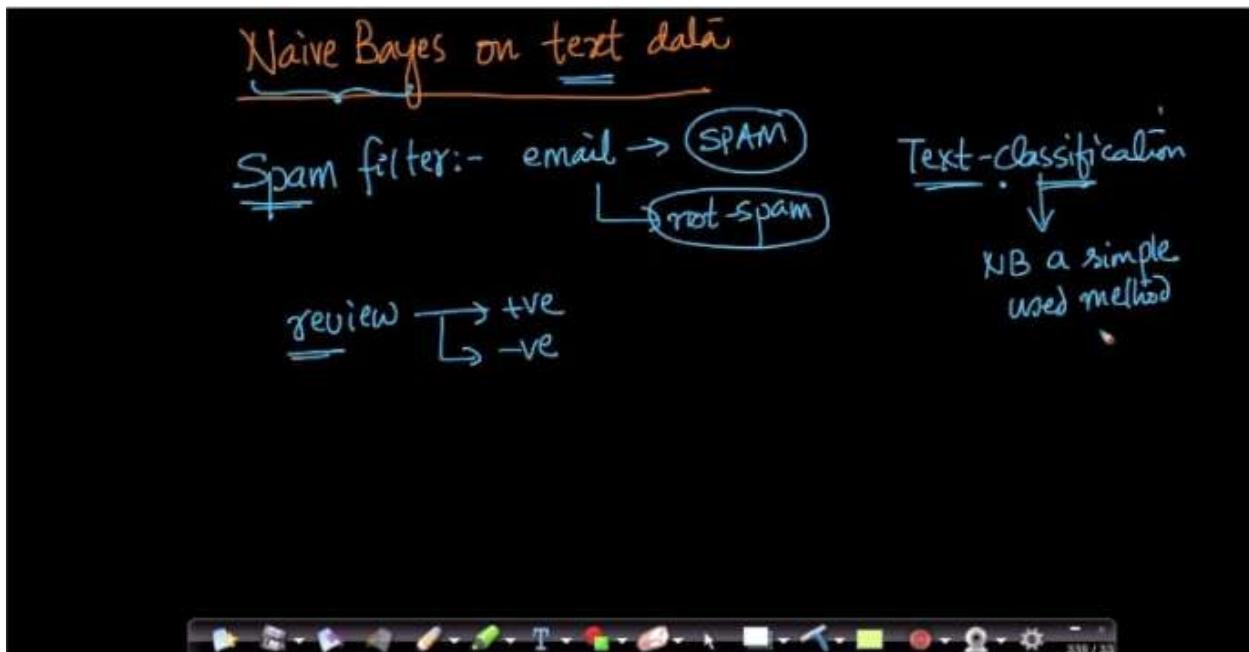
Timestamp 24:17

Now let's look at complexity during test time.

Time Complexity = $O(d*c)$, where d is the dimension of the data, c is the no of classes. This is because we need to look up d times i.e. for each feature, for each class i.e. c times. We are also considering our lookup time to be constant.

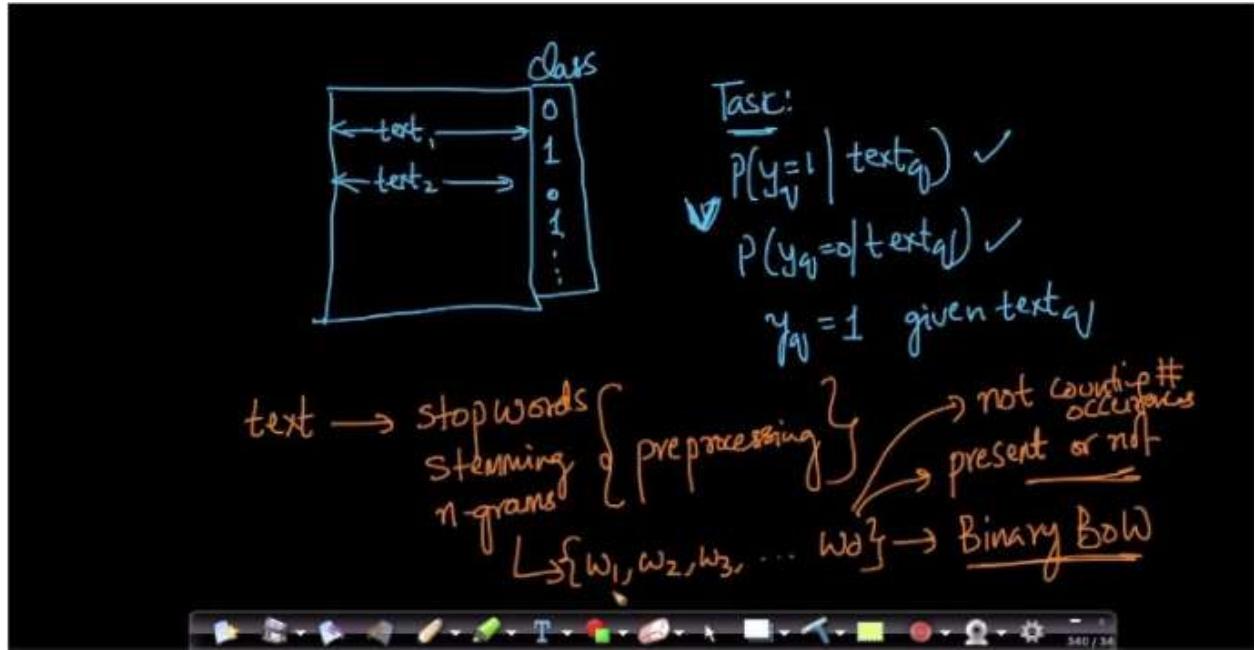
If we compare NB with KNN, we see that NB is extremely space efficient at runtime because in KNN we need all the data points in memory, whereas NB requires only the look up tables to be in the memory.

32.3 Naive Bayes on Text Data



Timestamp 1:31

NB is extremely useful when it comes to text data. One of the earliest spam filters was built using NB. A spam filter classifies a mail into $\{\text{Spam, Not Spam}\}$. NB is very efficient in classifying text data into various classes.



Timestamp 4:36

Suppose we are given a dataset where we have textual data; each text belonging to a certain class. Let's keep it to binary classification; it can be easily extended to multi-class classification problems. So our class labels are $\epsilon\{0, 1\}$. As shown in the image above.

Now we want to calculate,

$p(y_q = 1 | text_q)$ and $p(y_q = 0 | text_q)$ i.e. probability that the class is 0 or 1, given text $text_q$.

For handling text data we need to preprocess it, we can remove stopwords, do stemming, calculate n-grams.etc. All these steps are application dependent and not concrete.

After doing the preprocessing we need to convert this text data into numerical vectors. In NB we typically use binary bag of words(BOW). Binary BOW contains cells for each unique word in the corpus. Each cell can have a value of 0 or 1, representing whether that particular word is present in the current text or not. We will consider only those cells where its value is 1, i.e. the words present in the text.

$\text{text} \xrightarrow{\text{preproc}} \{w_1, w_2, \dots, w_d\}$

$$p(y=1 | \text{text}) = p(y=1 | \underbrace{w_1, w_2, \dots, w_d}_{\text{features}})$$

class prob
 $p(y=1) * p(w_1 | y=1) * p(w_2 | y=1)$
 ...
 likelihood

$$p(y=1 | \text{text}) \Leftarrow p(y=1) * \prod_{i=1}^d p(w_i | y=1)$$

Timestamp 7:00

Say we have a text $text_q$, after preprocessing we get $\{w_1, w_2, \dots, w_d\}$. These individual words are features.

Now we need to calculate, $p(y = 1 | text_q)$

So,

$$p(y = 1 | text_q) \propto p(y = 1) \prod_{i=1}^d p(w_i | y = 1)$$

The above term can be derived as shown in the image above.

$$p(y=0 | \text{text}) \propto p(y=0) \prod_{i=1}^d p(w_i | y=0)$$

$p(y=1) = \frac{\# \text{ Train pts with } y=1}{\text{Total } \# \text{ Train pts}}$
 $p(y=0) = \frac{\# \text{ train pts with } y=0}{\text{Total } \# \text{ train pts}}$

Timestamp 9:10

Similarly,

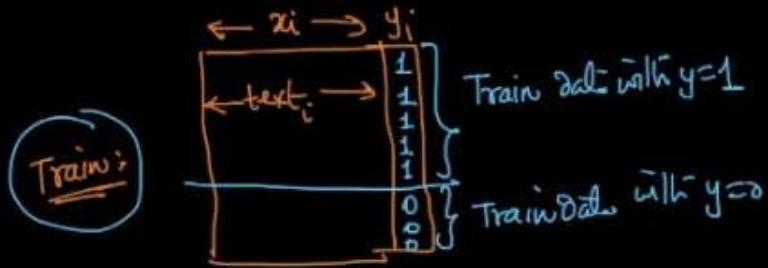
$$p(y=0 | \text{text}_q) \propto p(y=0) \prod_{i=1}^d p(w_i | y=0)$$

The prior probabilities i.e. $p(y=1)$ and $p(y=0)$ can be calculated as,

$$p(y=1) = (\# \text{train points with } y=1) / (\text{total } \# \text{ of points})$$

$$p(y=0) = (\# \text{train points with } y=0) / (\text{total } \# \text{ of points})$$

$$P(w_i | y=0) = \frac{\# \text{Train data pts with } w_i \text{ & } y=0}{\# \text{Train data pts with } y=0}$$



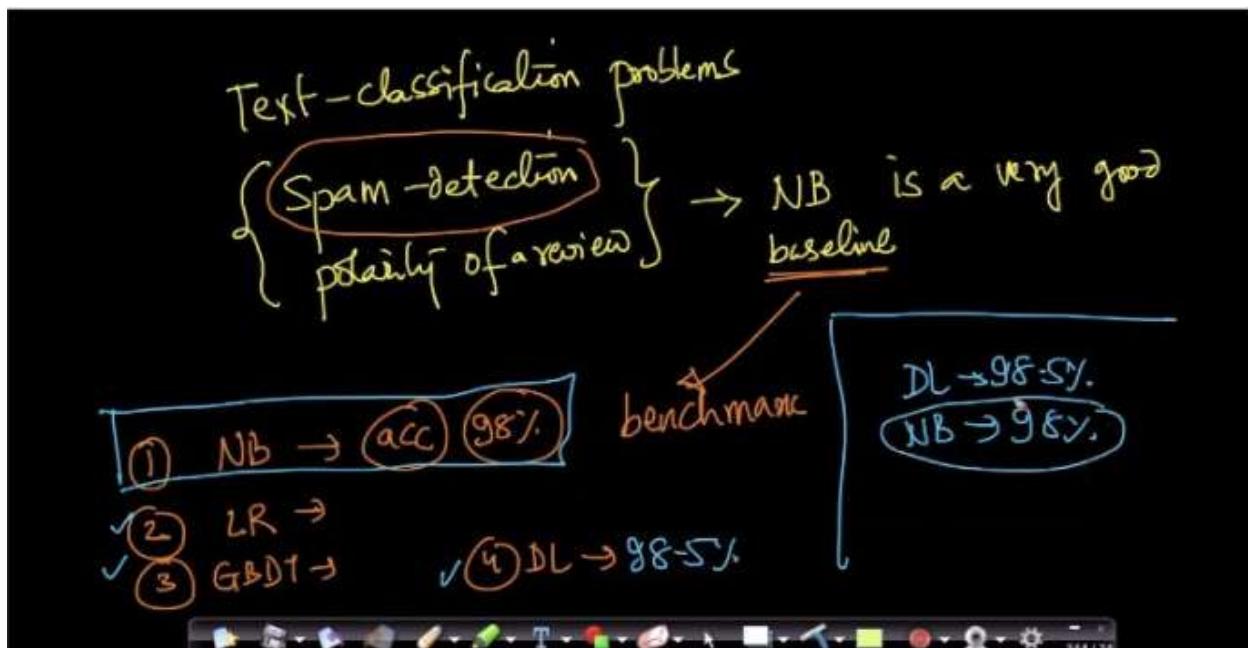
Timestamp 12:19

Now we can calculate the likelihoods, it can be calculated as

$$p(w_i | y = 0) = (\#\text{Train data pts which contains } w_i \text{ and } y = 0) / (\#\text{Train data pts with } y = 0)$$

Similarly it can be done for $y = 1$.

Now we have all the pieces for calculating the probabilities.



Timestamp 14:46

Naive Bayes serves as a very good baseline mode for problems like text classification, polarity of a review.etc. A baseline model is like a benchmark. We can train other models for text classification and compare with NB.

32.4 Laplace/Additive Smoothing

Laplace Smoothing:

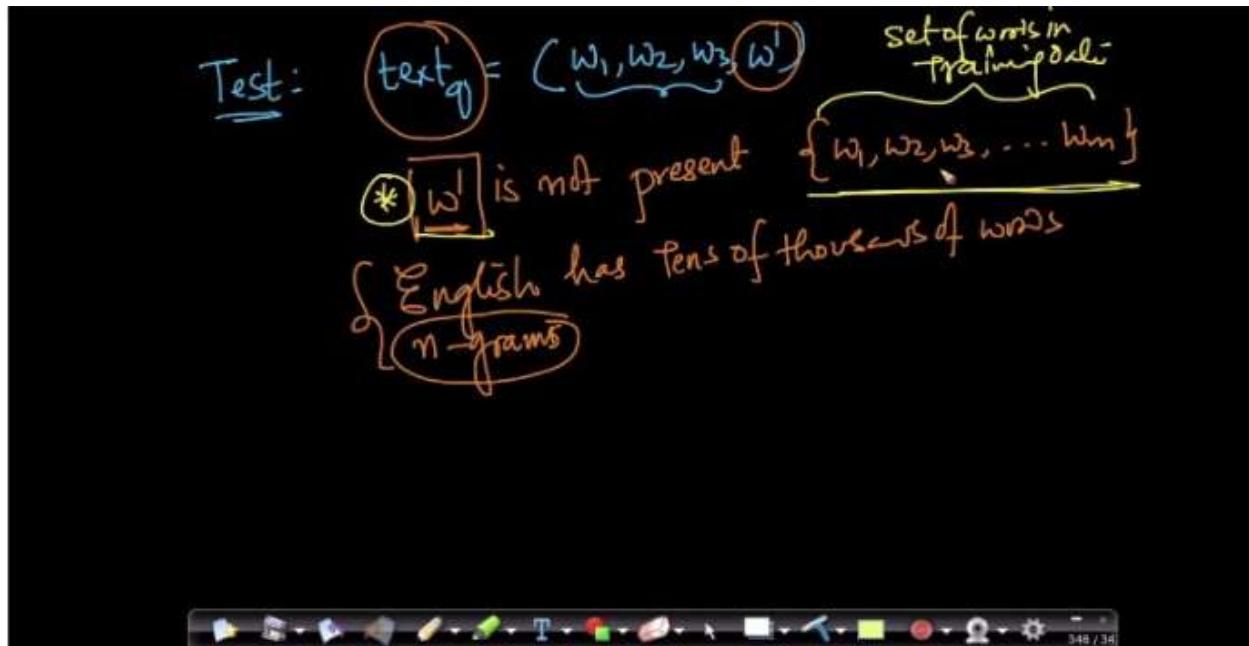
Training:-

$$\left\{ \begin{array}{ll} p(y=1) & ; p(y=0) \leftarrow \text{class priors} \\ p(w_1|y=1) & p(w_1|y=0) \\ p(w_2|y=1) & p(w_2|y=0) \\ \vdots & \vdots \\ p(w_m|y=1) & p(w_m|y=0) \end{array} \right\} \text{likelihoods}$$

Timestamp 1:14

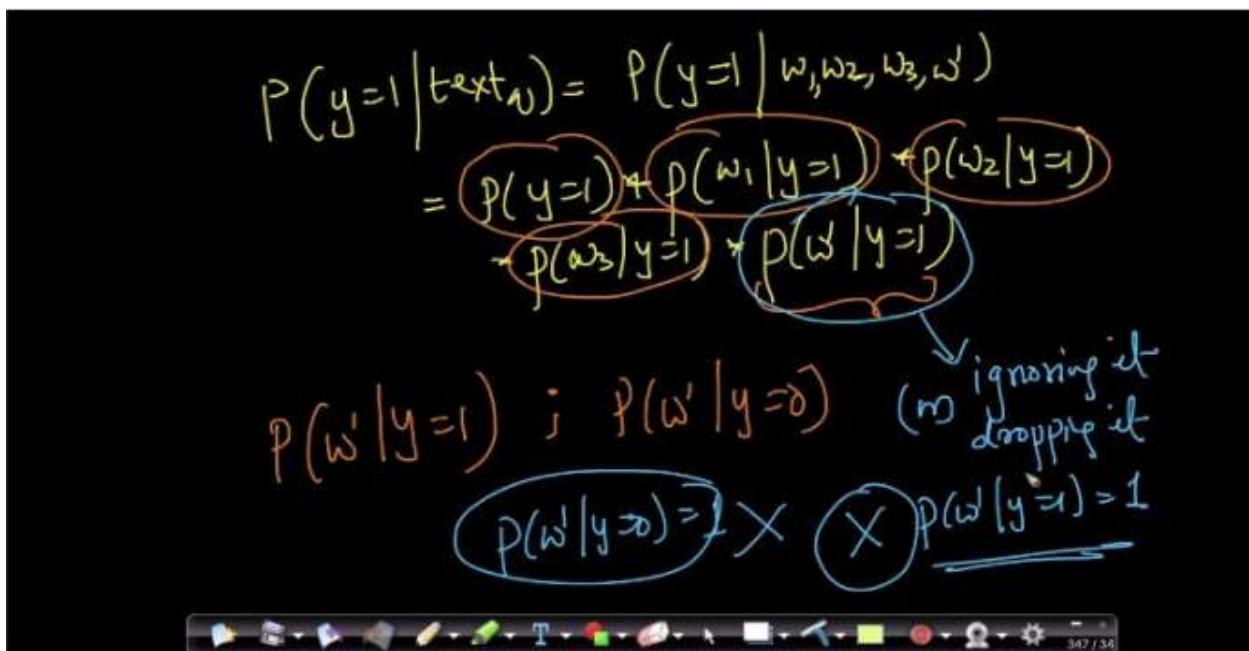
Here we will discuss something known as laplace smoothing/additive.

During the training phase we compute all the class priors and the likelihood of the features as shown in the image above. Now during train say we have a corpus of words $\{w_1, w_2, w_3, \dots, w_m\}$



Timestamp 3:24

Now during test time suppose we get a text $\text{text}_q = \{w_1, w_2, w_3, w^1\}$, it contains word w^1 , now this word is not present in our train dataset. English language contains lots of words and it is not always possible to get all the words in our corpus. Also if we are using n-grams, it may happen that some combination of words may not be present in our train data.



Timestamp 5:30

Now in order to calculate the probability of this text_q , we need to compute the likelihoods of individual words. One of the words that this text_q contains is w^l , which is not present in our train dataset.

So, the question is how to deal with this situation and what value should be used for $p(y = 1 | w^l)$ or $p(y = 0 | w^l)$.

One solution is to ignore or drop these unknown words. But this is not correct, because ignoring means that we are replacing it with 1. Meaning, $p(y = 1 | w^l) = 1$, which is incorrect.

$$\left\{ \begin{aligned} P(w^l | y=1) &= \frac{P(w^l, y=1)}{P(y=1)} \\ &= \frac{\# \text{train pts st } w^l \text{ occurs in } \text{y}=1}{n_1} \\ &\rightarrow \# \text{pts where } y=1 \\ &= \frac{0}{n_1} = 0 \end{aligned} \right.$$

Timestamp 12:17

Another solution is to calculate the actual probability, here n_1 is the no of data points where $y = 1$.

We have to calculate the term $p(w^l | y = 1)$, here the numerator denotes no. of terms where word w^l occurs and $y = 1$, which is 0, because there are no texts in the train dataset which contains the word w^l .

If we include this term in our calculation of probability the whole term becomes 0, which is not desirable.

Laplace smoothing (not Laplacian smoothing)

Additive smoothing

$$P(w^i | y=1) = \frac{o + \alpha}{n_1 + \alpha k}$$

$\alpha = 1$ Typically

w^i can take 2 distinct values

1 → present
0 → not present

Timestamp 10:23

Now in order to deal with this problem we use something known as laplace/additive smoothing. Remember not laplacian smoothing, which is used in image processing.

In this technique we add constant terms to both numerator and denominator. In numerator we add a α whose value is typically 1(not always), and in the denominator we add $\alpha \times k$, where k is the number of distinct values a feature can take.

In our example $k = 2$ as a word w_i can be either present or not.

So our probability for w^i becomes,

$$P(w^i | y=1) = (o + \alpha) / (n_1 + \alpha \times k)$$

$$p(\underline{\omega}^1 | y=1) = \frac{0 + \alpha}{100 + 2\alpha}$$

Let $n_1 = 100$

$\alpha = 0.00$

$k=2$ because $\underline{\omega} = 0 \text{ or } 1$

case 1:- $\alpha = 1 = \frac{1}{102} \neq 0$

$0 \neq p(y=1 | \text{test}_1) \Leftrightarrow p(\omega | y=1) \neq 0$

Timestamp 12:28

Now, we have gotten around the problem of getting 0 values as we are adding constants. Let's take a case where $\alpha = 1$, $n_1 = 100$ and in our example we had $k = 2$.

So, $p(w^1 | y = 1) = (0 + 1)/(100 + 1 \times 2) = 1/102 \neq 0$.

Now, the question arises why this is the correct way to deal with this problem, we could have simply replaced our probability for words which don't occur in our train data with some small value(ε).

Case 2 :- $\alpha = 10000$

$$P(\omega^1 | y=1) = \frac{0 + 10000}{100 + 20000} = \frac{10000}{20100} \approx \frac{1}{2}$$

$n_1 = 100$

$\checkmark \omega^1 = 0 \rightarrow \text{two possibilities}$

$\checkmark \omega^1 = 1$

$$\boxed{P(\omega^1 | y=1) = P(\omega^1 | y=0) = \frac{1}{2}}$$

Timestamp 15:35

We will look at another case where α is large, say = 10000. In this case we get the probabilities as 1/2 for both the classes as shown in the image above which is much better than ignoring it. As we are saying that we don't know about this unknown word and how it affects class values so let's take equal probabilities for both the classes. Introducing large α 's means introducing high bias.

Laplace Smoothing

$P(\omega_i | y=1) = \frac{\# \text{data pts with } \omega_i \text{ by } y=1 + \alpha}{\# \text{data pts } y=1 + \alpha k}$

$\checkmark \omega_i \text{ present in my training data}$

Timestamp 17:08

So we will introduce the laplace smoothing for all the words in our training dataset, not just for unknown words. So, for every term we are adding something to the likelihoods, that's the reason it is called additive smoothing.

$$p(w_i | y=1) = \frac{2 + \alpha}{50 + \alpha}$$

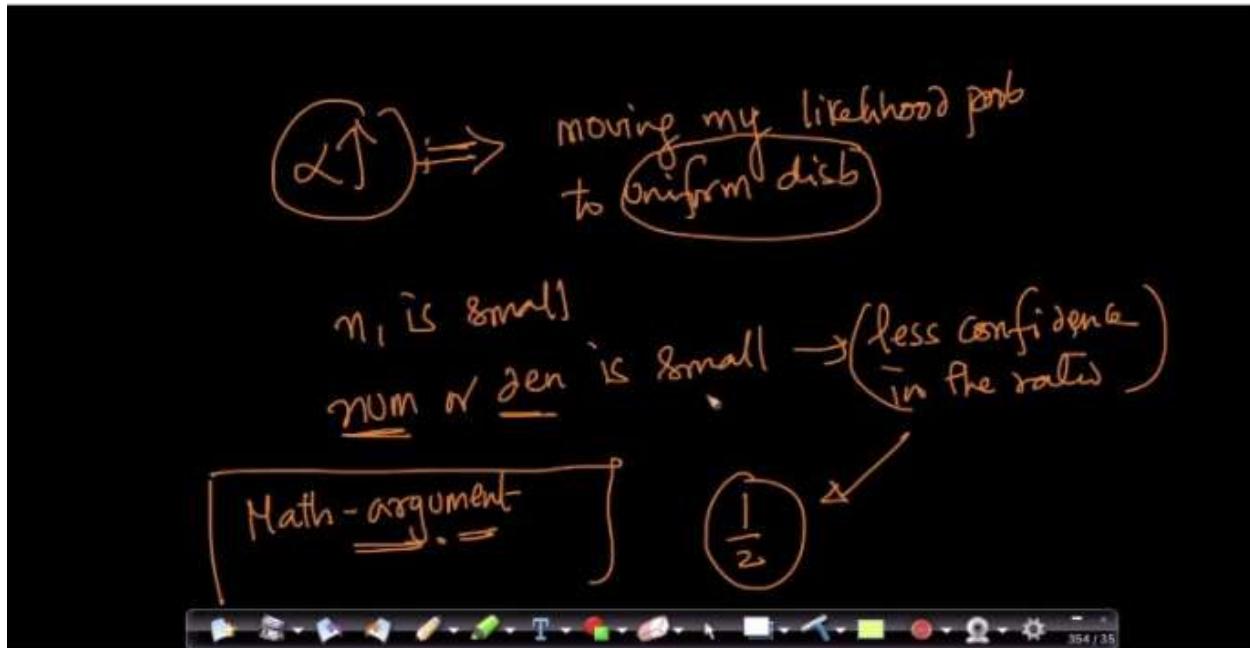
$$\left. \begin{array}{l} \alpha = 1 \Rightarrow \frac{2+1}{50+1} = \frac{3}{54} \\ \alpha = 10 \Rightarrow \frac{2+10}{50+20} = \frac{12}{70} \\ \alpha = 100 \Rightarrow \frac{2+100}{50+200} = \frac{102}{250} \end{array} \right| \quad \begin{array}{l} \alpha = 1000 \Rightarrow \frac{2+1000}{50+2000} = \frac{1002}{2050} \\ \qquad\qquad\qquad \downarrow \\ \qquad\qquad\qquad \frac{1}{2} \end{array}$$

Timestamp 17:26

Here we will try to answer why it is called smoothing. For that let's take a word w_i , and calculate the probability $p(w_i | y = 1)$,

Say $p(w_i | y = 1) = 2/50$.

Now we will introduce laplace smoothing and calculate the probability values for different α 's. We have calculated various probabilities based on different values α as it can be seen in the image above. We can observe from the values that we got that as the value of α 's are increasing the probability values are moving towards 1/2.



Timestamp 21:38

As the value of α increases we are moving our likelihood probabilities to uniform distribution. Remember in uniform distribution all the values have equal probabilities. So it causes a smoothing of the probabilities values. In our case we had $k = 2$, that's why we were getting $1/2$. We can also use laplace smoothing if we have very few points of a certain class. The mathematical proof of laplace smoothing is beyond the scope of this course.

32.5 Log-probabilities for numerical stability

Log-Probabilities vs numerical stability

$$P(y=1 \mid w_1, w_2, \dots, w_d)_{0 \text{ to } 1}$$

$$= p(y=1)_{0 \text{ to } 1} * p(w_1|y=1) * p(w_2|y=1) * p(w_3|y=1) * \dots * p(w_d|y=1)$$

d is large (let $d=600$)

The bottom of the screen shows a toolbar with various icons.

Timestamp 0:57

In order to calculate the probability we need to multiply the likelihoods of different features. Say we have a probability term $p(y = 1 | w_1, w_2, \dots, w_d)$ as shown in the above image, where d is the number of features, which say is very large like 100. Also keep in mind that all the terms are probability terms so their value lie between 0 & 1.

A handwritten note on a black background. At the top, there is a curly brace grouping four numbers: 0.2, 0.1, 0.2, and 0.1. Below this group is the multiplication symbol \times . To the right of the group is the result: 0.0004. Below this result, there is a large curly brace grouping 100 such numbers, with the text "all of which 0 to 1" written above it. Below this large group, there is another curly brace grouping two items: "numerical stability" and "numerical underflow". To the right of this brace, there is a vertical curly brace grouping "python double precision" and "(16 significant digits)". Below "double precision", the word "values" is written, with an arrow pointing to the word "rounding". A small circle highlights the number 16. At the bottom of the slide, there is a standard presentation navigation bar.

Timestamp 2:40

Suppose we take four probability values 0.2, 0.1, 0.2, 0.1. If we multiply these values we get 0.0004. This is a very small number and if we multiply 100 such numbers it can lead to issues of numerical stability. In python if we have a variable of double precision it can have upto 16 significant digits. If we get more significant digits than this, it starts rounding the number. This causes numerical underflow.

Log-probabilities:

$$\log \left(\frac{P(y=1 | w_1, w_2, \dots, w_d)}{P(y=0 | w_1, w_2, \dots, w_d)} \right) = P(y=1) + \prod_{i=1}^d p(w_i | y=1) - P(y=0) - \prod_{i=1}^d p(w_i | y=0)$$

$\boxed{x \uparrow ; \log x \uparrow}$
 ↳ monotonic fn

Timestamp 5:25

In order to deal with this numerical instability we use log probabilities. Log probabilities are nothing but we use log on either side of the equation. We can do this because $\log x$ is a monotonic function, as x increases $\log x$ also increases. If we take $\log(0.0004) = -3.39$, which is better than dealing with high precision numbers. As this negative no. has lesser significant digits.

$$\log \left(P(y=1 | w_1, w_2, \dots, w_d) \right) = \log(P(y=1)) + \sum_{i=1}^d \log(p(w_i | y=1))$$

$\leftarrow \boxed{\log(a+b) = \log a + \log b}$
 ↳ mult ↳ add

Timestamp 6:53

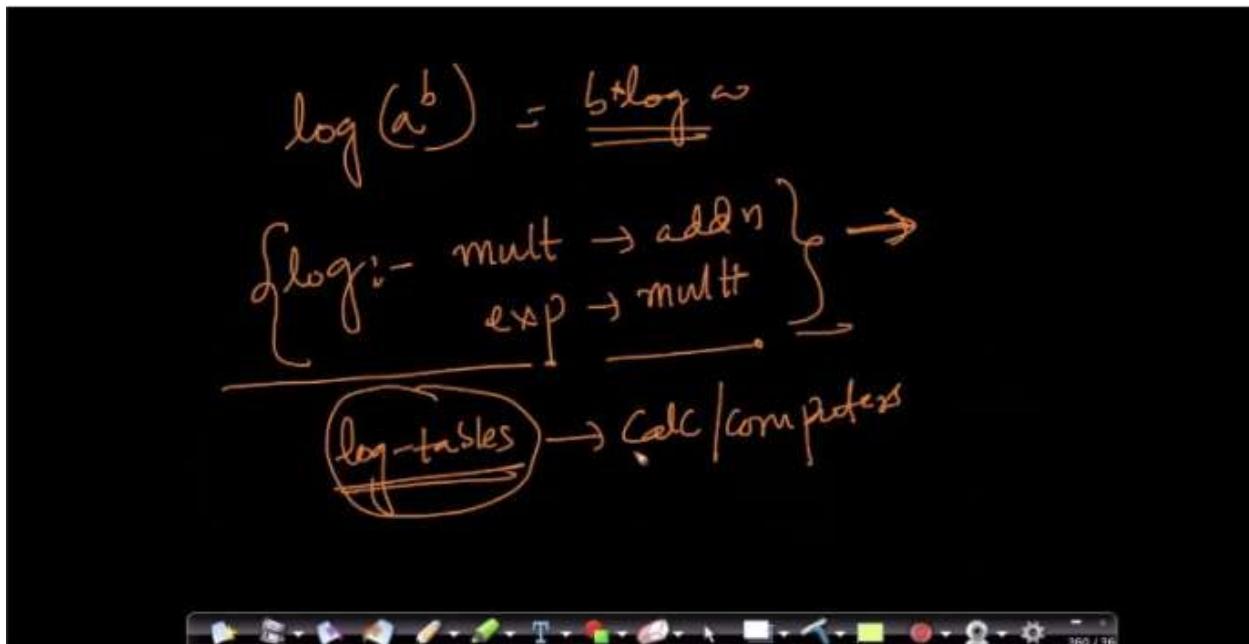
So, our probability equation is,

$$p(y = 1 | w_1, w_2, w_3, \dots, w_d) = p(y = 1) \prod_{i=1}^d p(w_i | y = 1)$$

Taking log on both sides,

$$\log(p(y = 1 | w_1, w_2, w_3, \dots, w_d)) = \log(p(y = 1)) + \sum_{i=1}^d \log(p(w_i | y = 1))$$

The product term is converted into a sum term because $\log(a*b) = \log(a) + \log(b)$. So we can now add log values without worrying about numerical instability.



Timestamp 9:40

Logs are extremely useful because it converts multiplication into addition

$$\log(a*b) = \log(a) + \log(b),$$

It also converts exponentiation into multiplication

$$\log(a^b) = b * \log(a)$$

In earlier times we used to use log-tables for calculating these log - values.

Log-probabilities help us prevent numerical instability because it's easier to deal with negative numbers than numbers of high precision.

32.6 Bias and Variance Tradeoff

Bias - Variance tradeoff

{ Naive Bayes :- $\begin{cases} \text{high bias} \rightarrow \text{underfitting} \\ \text{high variance} \rightarrow \text{overfitting} \end{cases}$ }

{ in Laplace smoothing
 ↳ high bias
 ↳ high variance }

Timestamp 0:50

In this chapter we will deal with the bias - variance tradeoff for NB.

Recall that high bias means underfitting and high variance means overfitting. In NB only α determines whether we have high bias or high variance.

Case 1:- $\alpha = 0$

$$P(w_i | y=1) = \frac{\# \text{Train data pts } w_i \text{ occurs as } y=1}{\# \text{Train pts with } y=1}$$

$$(n = 2000 \text{ pts} \rightarrow 1000 \text{ +ve}) = \frac{\text{②} \rightarrow \text{only 2 times} \Rightarrow \text{rare}}{1000 \rightarrow \text{true data pts}}$$

{ words that are rare } $\hookrightarrow P(w_i | y=1)$ \rightarrow overfitting

Timestamp 3:55

Let's look at various cases indicating various values of α ,

Case 1: $\alpha = 0$

Here we have considered the value of α to be 0. Let's say we want compute the likelihood of w_i ; in our dataset we have 2000 data points, where 1000 = positive and 1000 = negative.

$$p(w_i | y = 1) = (\# \text{Train data pts which contains } w_i \text{ and } y = 1) / (\# \text{Train data pts with } y = 1)$$

Say we got this probability value as = 2/1000. We can see that this particular word w_i is very rare as it occurs in only two positive documents and also note that we don't have any smoothing as α is set to 0. Now even for this word which is very rare we will have a probability value. If we change our model data i.e. train data slightly and remove these two texts where w_i occurs, these probability changes from 2/1000 to 0/1000. This means slight change in data is changing our model considerably indicating high variance. High variance means overfitting.

So lower α values means overfitting.

Case 1: $\alpha = 0 \Rightarrow$ small change in D_{train} results in large change in the model \Rightarrow high variance overfitting

Case 2: α is very large: - ($\alpha = 10000$)

$$p(w_i | y = 1) = \frac{2 + 10000}{1000 + 20000} \approx \frac{1}{2}$$

\downarrow
D n' 1
 $K=2$

Timestamp 7:14

Case 2: α is very large say 10000

Again we calculate the probability of the word w_i , but this time with high smoothing. Also in our example $k = 2$.

So,

$$p(w_i | y = 1) = (2 + 10000) / (1000 + 20000) \approx 1/2$$

$$\text{For } \alpha \gg 1 \Rightarrow p(w_i | y=1) \approx p(w_i | y=0) \approx \frac{1}{2}$$

$$p(y_{\text{obs}} | \underline{x}_{\text{q}}) \approx p(y_{\text{obs}} = 0 | \underline{x}_{\text{q}}) \approx \frac{1}{2}$$

(Underfitting)

K-NN: $K = n$

$x_q \rightarrow$ Same class label (+ve)

$n_1 > n_2$

$n_1: +ve$

$n_2: -ve$

Timestamp 11:13

For every class we will get the same probability. As our α term dominates the probability values and less importance is given to actual probability values. Our actual prediction depends only on the class priors. So, whichever class is present in the majority our model will always predict that class. This means we are underfitting.

So higher α values means underfitting.

We can compare this situation with KNN, where we can set our neighbours $K = n(\# \text{data points})$, then for every query we will get the same prediction. The prediction will be the class, which is present in majority in the train dataset. Leading to underfitting.

(Q) How to find the right α

→ KNN; right $K \rightarrow$ using SimpleCV
or CV

→ right α : → SimpleCV
or CV

Timestamp 12:45

Now how to find the right α , we can find this using cross validation, like we did in KNN. α is called a hyperparameter. Similarly the value of k in KNN is also a hyperparameter.

32.7 Feature Importance and Interpretability

Feature importance & interpretability

(NB) - feature imp

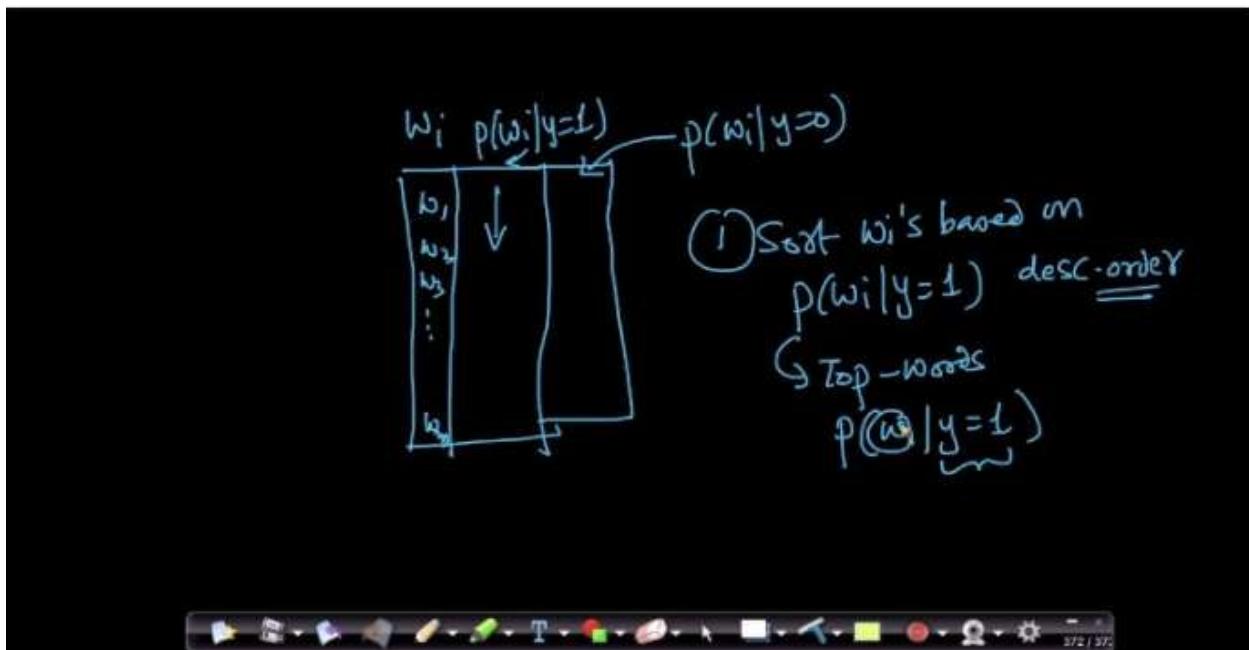
Likelihood:

$w_i, p(w_i | y=1)$

for all $w_i, p(w_i | y=0)$

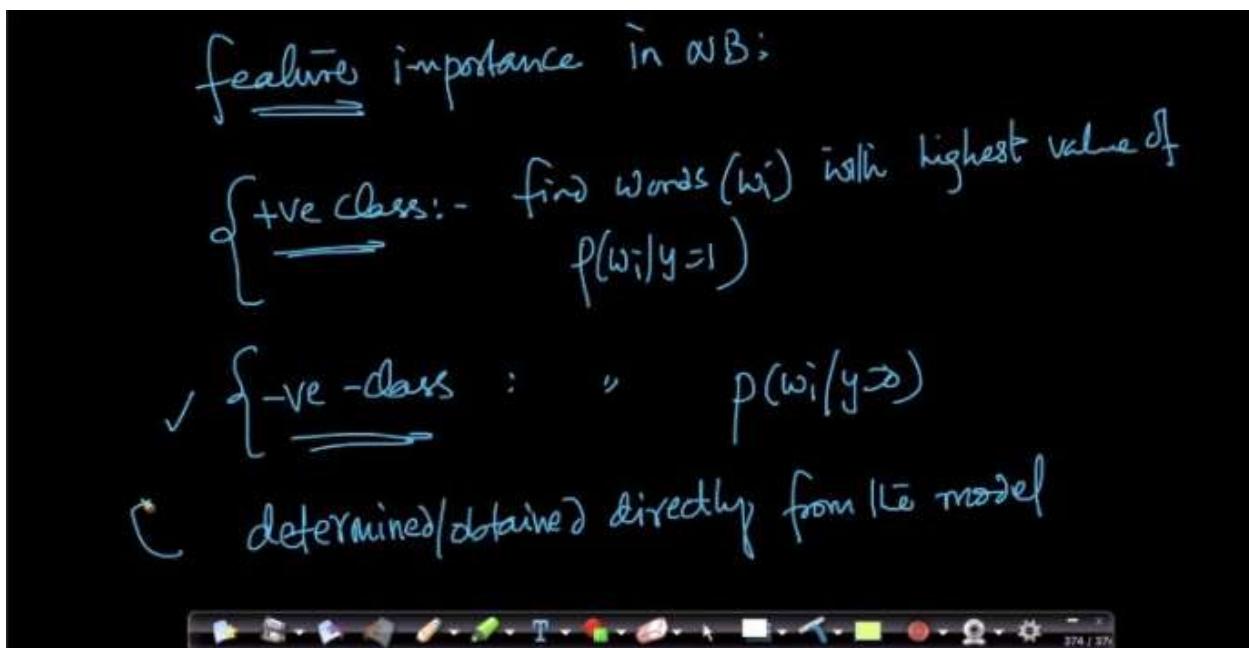
Timestamp 1:07

In this chapter we will try to find the feature importance and interpretability for a prediction. In order to find these we need to calculate the likelihood for all words.



Timestamp 2:35

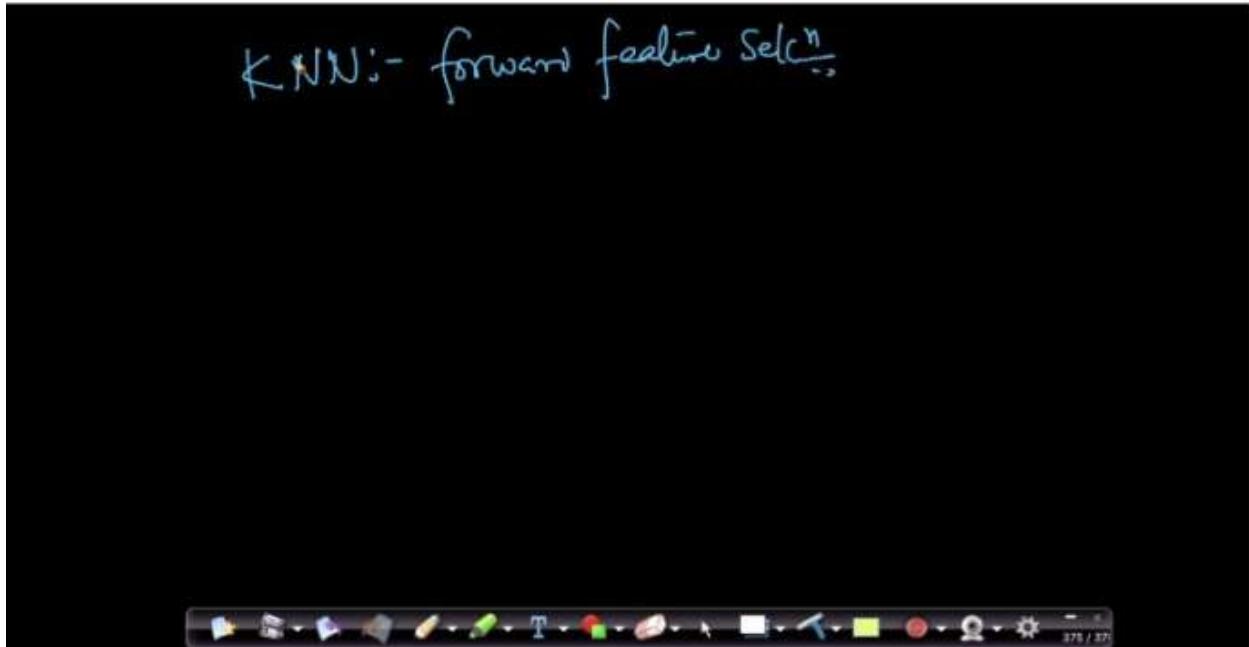
We will calculate the likelihood i.e. $p(w_i|y=1)$ and $p(w_i|y=0)$ for each word and store it into a table. Now we need to sort this table as per their probability values in descending order. Words which occur on top of this sorted table i.e. highest probabilities are the most frequent words in a particular class.



Timestamp 5:29

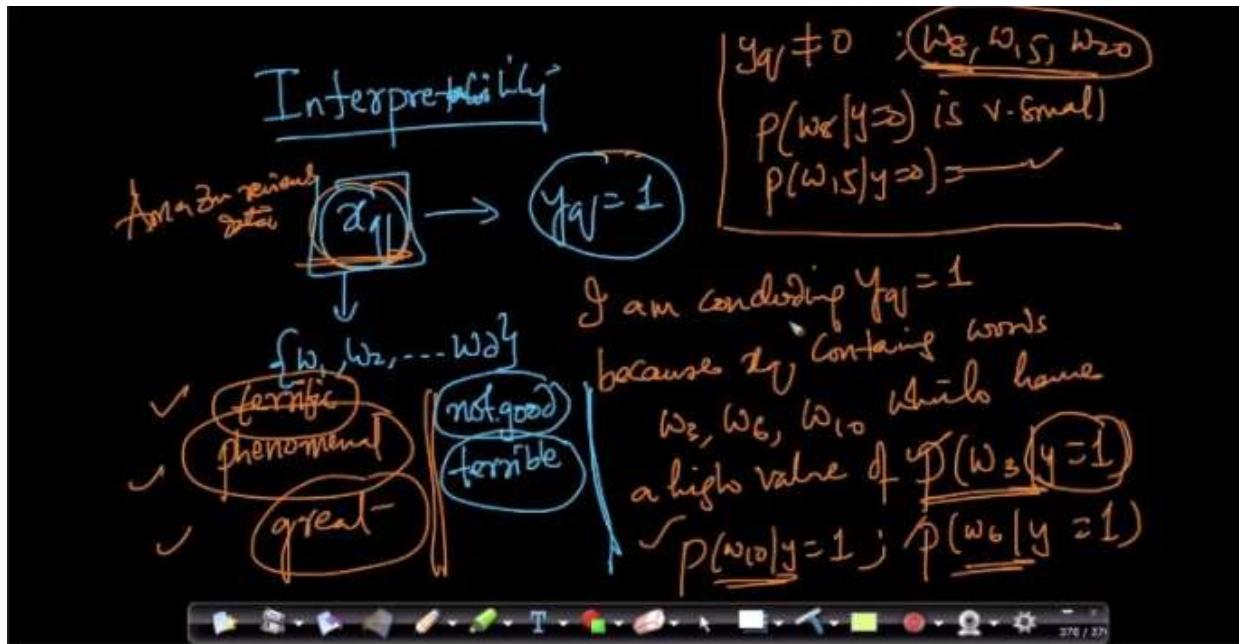
So features/words which possess the highest probability values are the most important features for a particular class. As when we compute the total probability, features/words which have the highest likelihood have the highest impact in total probability value.

Feature importance in case of NB can be directly computed from the model.



Timestamp 5:44

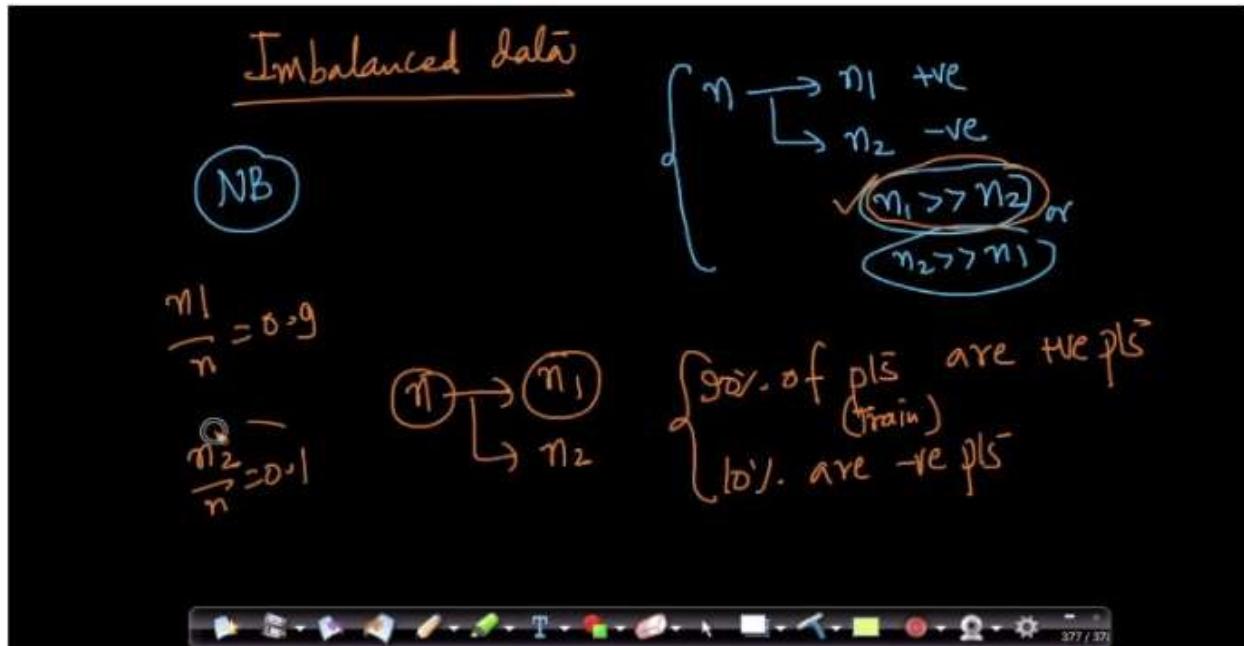
This is unlike KNN where we used techniques like forward and backward feature selection for determining the feature importance.



Timestamp 9:40

Now let's talk about interpretability, suppose we have a query point x_q , containing words $\{w_1, w_2, w_3, \dots, w_d\}$, suppose we got $y_q = 1$. Now we can look at the individual probability terms and find those words which have the highest values, say we got $\{w_3, w_6, w_{10}\}$. From this we can conclude that we got the class label $y_q = 1$, because of the high likelihood of these words. These words determined our class label.

32.8 Imbalanced data



Timestamp 1:48

In this chapter we will look at the effect of an imbalanced dataset on Naive Bayes. Let's quickly define imbalanced dataset, say we have dataset containing n data points, where n_1 is the number of positive points and n_2 is the number of negative points. Now if n_1 is significantly greater than n_2 i.e. $n_1 \gg n_2$ or n_2 is significantly greater than n_1 i.e. $n_1 \ll n_2$, we call such a dataset as an imbalance dataset.

For our case let's assume that $n_1 \gg n_2$, say 90% of the data are positive and the remaining 10% are negative.

$$\text{So, } n_1/n = 0.9 \text{ and } n_2/n = 0.1$$

$$p(y=1 | w_1, w_2, \dots, w_d) = \frac{p(y=1)}{\prod_{i=1}^d p(w_i | y=1)}$$

$$p(y=0 | w_1, w_2, \dots, w_d) = \frac{p(y=0)}{\prod_{i=1}^d p(w_i | y=0)}$$

imbalanced data
① class priors → majority / dominating class has an advantage

Timestamp 3:35

If we calculate the probabilities for each of the classes, we need to calculate two terms: class priors and likelihoods. Suppose the likelihood is same for both the classes then we are left with priors. Prior for the majority class will dominate over the prior for the minority class. In our case the positive class will dominate over the minority class.

So, in case of an imbalanced dataset, the majority class has an undue advantage.

solutions:

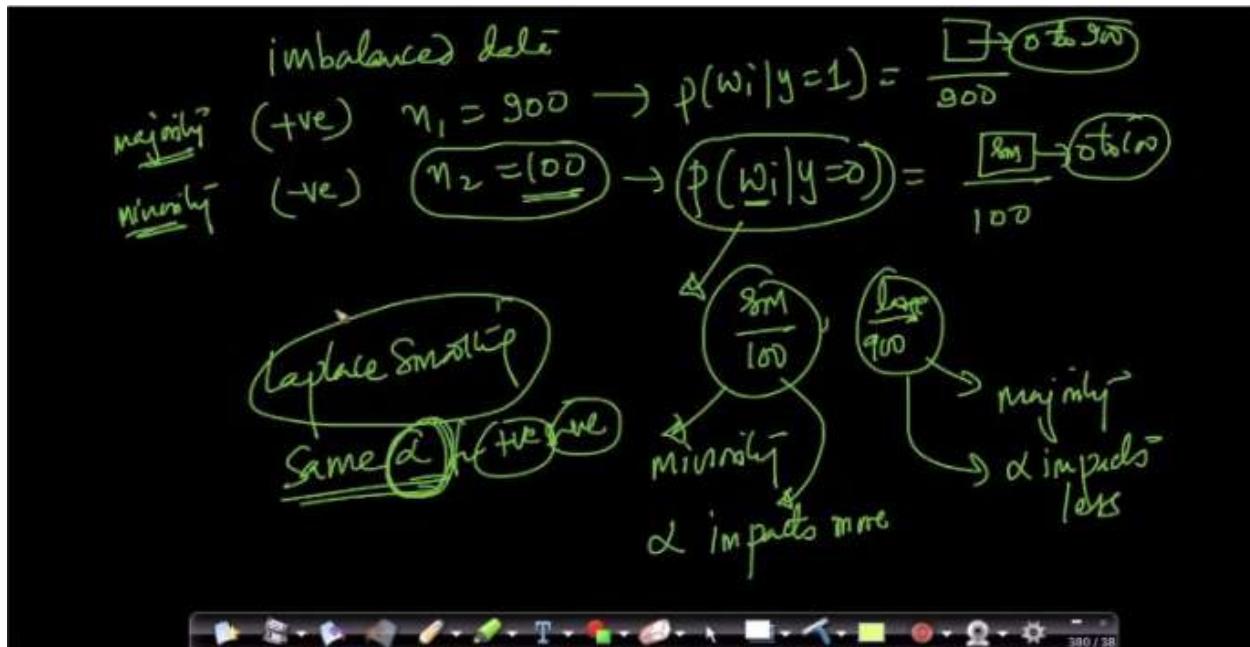
- ① upsampling or down sampling
 $\hookrightarrow n_1 \approx n_2 \quad \boxed{p(y=1) = p(y=0) = \frac{1}{2}}$
- ② drop $p(y=1)$ & $p(y=0)$
 $p(y=1) = p(y=0) = 1$
- ③ modified NB to account for class imbalance
 \hookrightarrow not often used

Timestamp 6:18

Here we will look at ways to handle imbalance,

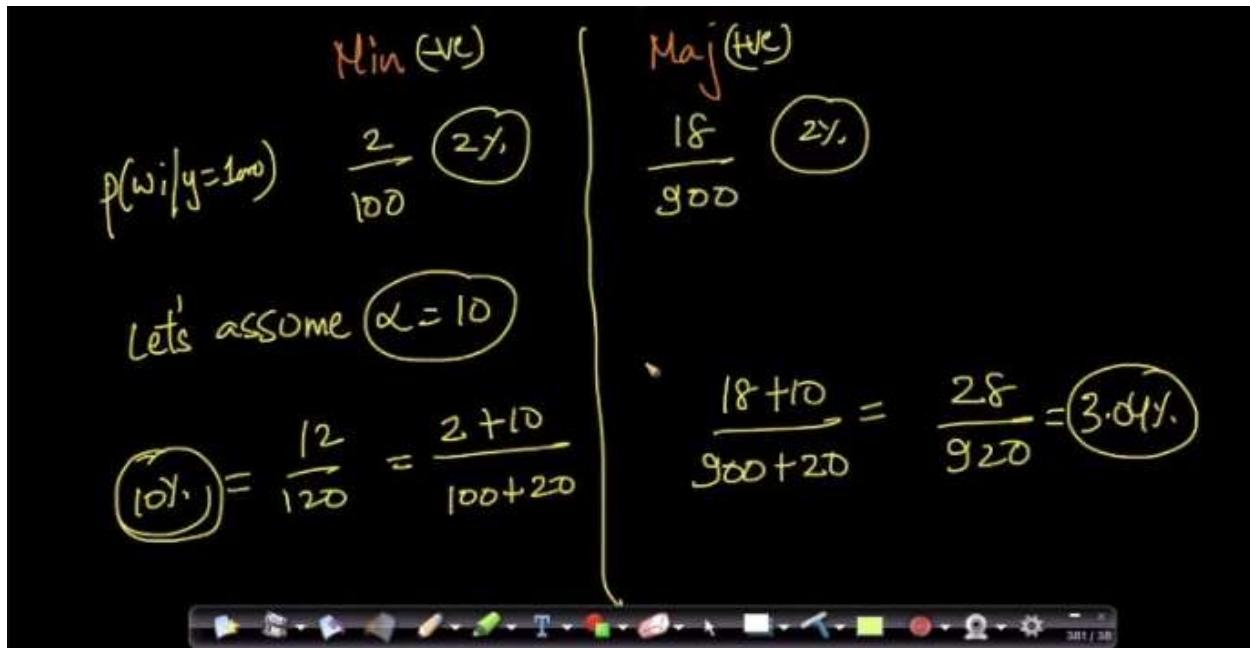
1. Upsampling or Downsampling - We can either upsample the minority class or downsample the majority class to make both these classes equal, i.e. $n_1 = n_2$. If we do this we are effectively making our class priors = 1/2
2. We can simply drop class priors. Dropping class priors effectively means setting them to 1. If we observe carefully both point 1 and 2 are equivalent as in both the cases we are setting the class priors to the same value. Since we only compare two probabilities it doesn't matter what priors we use as long as they are the same.
3. Modified NB, we can modify our NB to account for class imbalance, however it is seldom used.

Mostly point 1 and 2 are used in practice.



Timestamp 8:56

Here we will look at the effect of laplace smoothing in an imbalance dataset. Let's say we have a dataset having 900 positive points and 100 negative points, here the positive class is the majority class. Now let's say we calculate the likelihood of the word w_i for both these classes as shown in the image. If we apply laplace smoothing on both these two terms the minority class is impacted more than the majority class.



Timestamp 11:15

Let's see it with an example.

As we can see from the image above we calculated the probabilities for both the classes without using any smoothing. For both the majority and minority class we got the probability % to be 2%. Now after applying laplace smoothing with $\alpha = 2$, the probability % for the minority class shot up to 10% from 2% whereas for the majority class it went to 3.04%, showing that laplace smoothing affects the minority class more.

- ① Upsampling or downSampling
- ② Hacks: $\alpha_{+ve}, \alpha_{-ve}$

Timestamp 12:24

One of the hacks we can use to handle the impact of laplace smoothing is to use different α 's for different classes. However this technique is not grounded in theory. It's better to follow techniques like upsampling/downsampling to handle class imbalance.

32.9 Outliers

Outliers :-

NB

Text-classfni:-

outlier at test time $\rightarrow z_{qj} = w_1, w_2, w_3, w'$

$w' \notin \{w_1, w_2, \dots, w_n\}$ ← Set of words that I have seen in Train

✓ Laplace Smoothing ✓

Timestamp 1:42

In this chapter we will see how to deal with an outlier.

Suppose we are doing text classification and during test time we see a word w^l , which was not present during train time. If we calculate the likelihood of this value we will get 0 making the whole probability 0. So in order to deal with this situation we use laplace smoothing which we learnt in earlier chapters.

Outliers in Training Data:

$$\{w_1, w_2, w_3, \dots, w_m\} \leftarrow \text{Set of words in } D_{\text{train}}$$

w_8 → occurs very few times
in the 5 classes

outlier

Timestamp 2:54

There can be outliers even in the training data. An outlier in training data can be defined as a word/feature which occurs very few times in training data. It is such a point around which there are no points.

Handle Outliers:

- ① If a word (w_i) occurs fewer than $c=10$ times, then just ignore that word.
- ② Laplace Smoothing (α)

$$P(w_i | y = l) = \frac{\alpha}{100}$$

Timestamp 5:06

We can deal with these problems in training data by simply ignoring the features/words. We can set a threshold and say if a particular feature occurs below a certain threshold we will ignore such words.

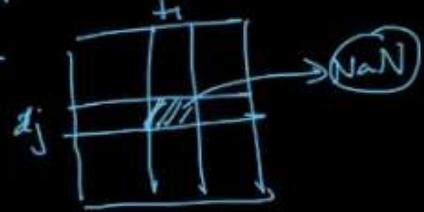
Another solution is to use laplace smoothing with an appropriate value of α .

32.10 Missing values

Missing-Values :-

① Text-data :- Text :- $\{w_1, w_2, \dots, w_n\}$
↳ no case of missing data

② Categorical features :-
 $f_i \in \{a_1, a_2, a_3\}$
↳ consider NaN as a category
 $f_i \in \{a_1, a_2, a_3, \text{NaN}\}$

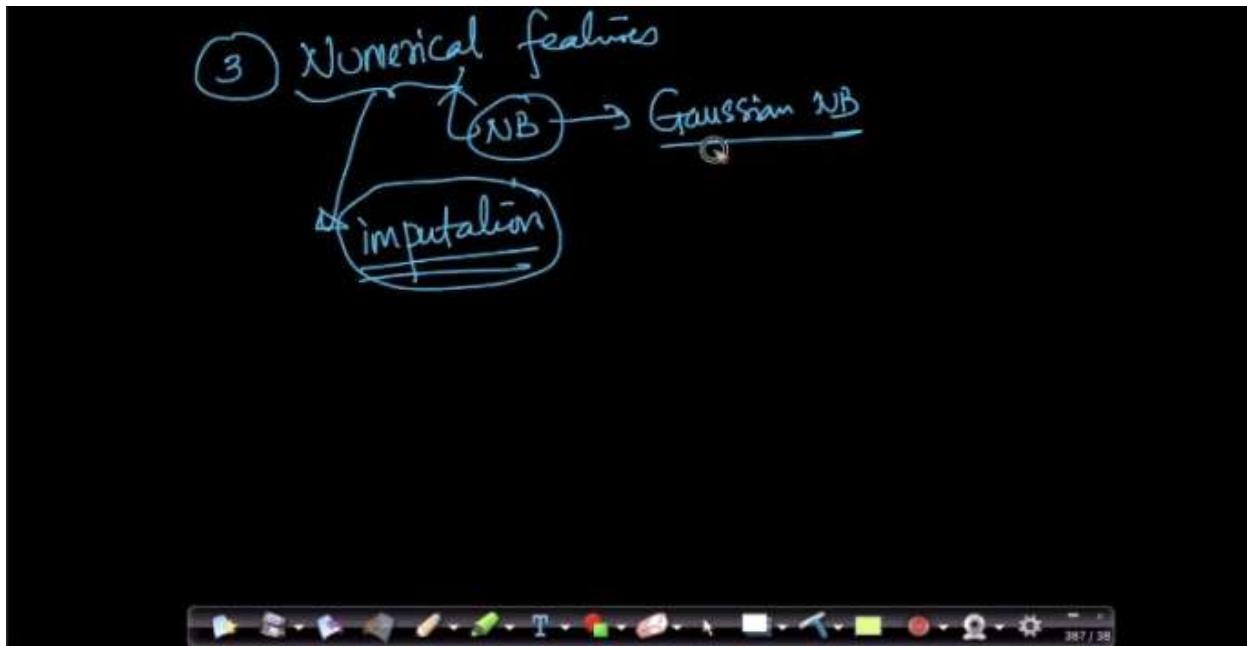


385 / 38

Timestamp 1:56

In this chapter we will talk about dealing with missing values.

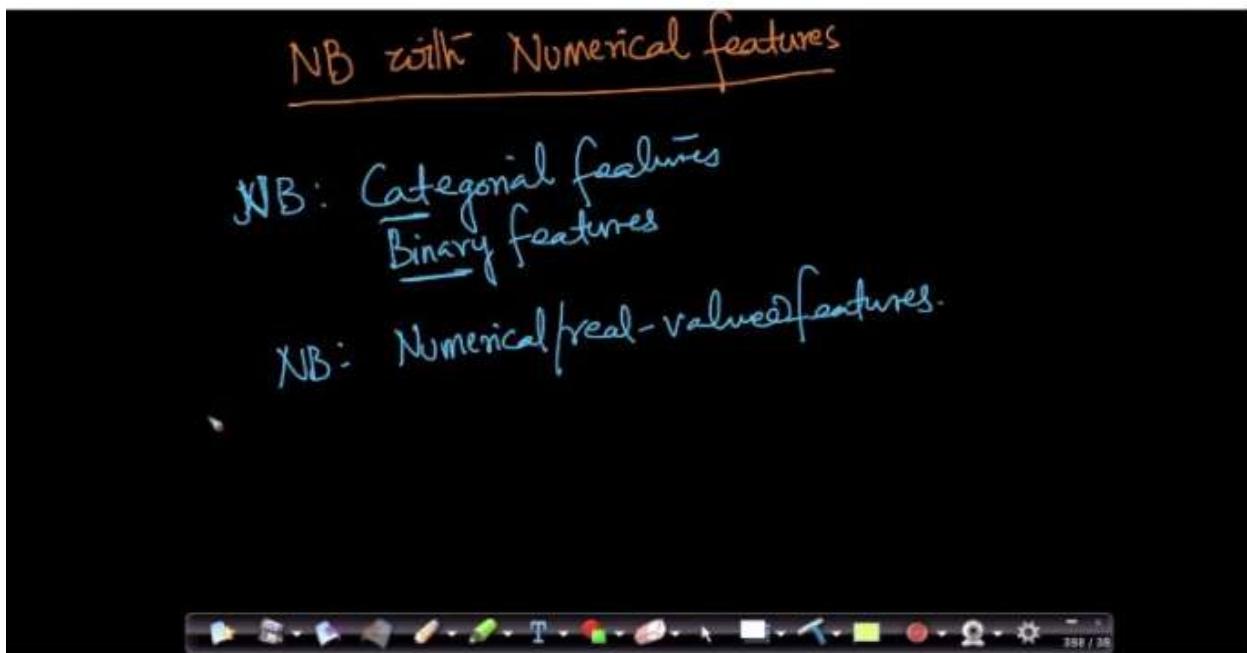
1. Text data - If we are dealing with text data there is no case of missing data, either the word is present or not present.
2. Categorical features - Say we have feature f_i , having categories $\{a_1, a_2, a_3\}$. Now suppose in one of the rows this feature value is missing. So we can consider the missing value NaN, itself as a feature. So, our feature set becomes $\{a_1, a_2, a_3, \text{NaN}\}$. This is one of the hacks that one can use. Also it is not limited to Naive Bayes but can be used anywhere.



Timestamp 2:55

3. Numerical Features - We can handle missing numerical features by using various imputations like mean, median.etc.

32.11 Handling Numerical Features(Gaussian NB)

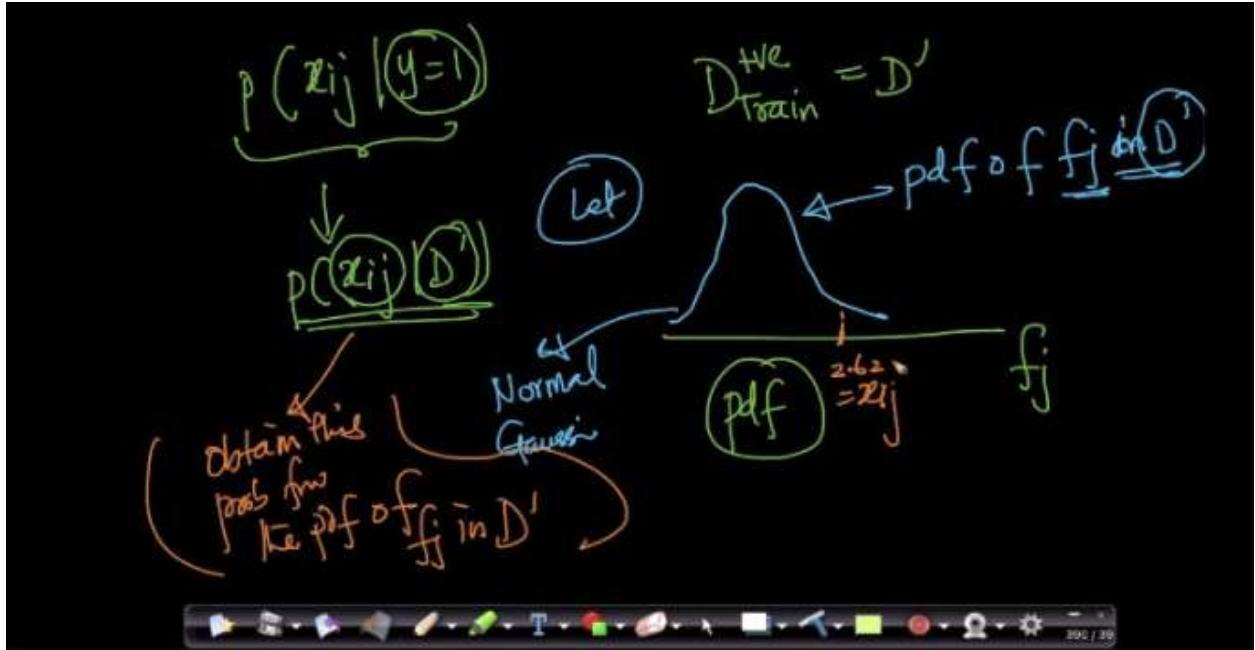


Timestamp 0:36

In this chapter we will learn how to deal with numerical(real valued) features in NB. Till now we have only worked with categorical and binary(words) features. In order to deal with numerical features we will use something known as Gaussian NB.

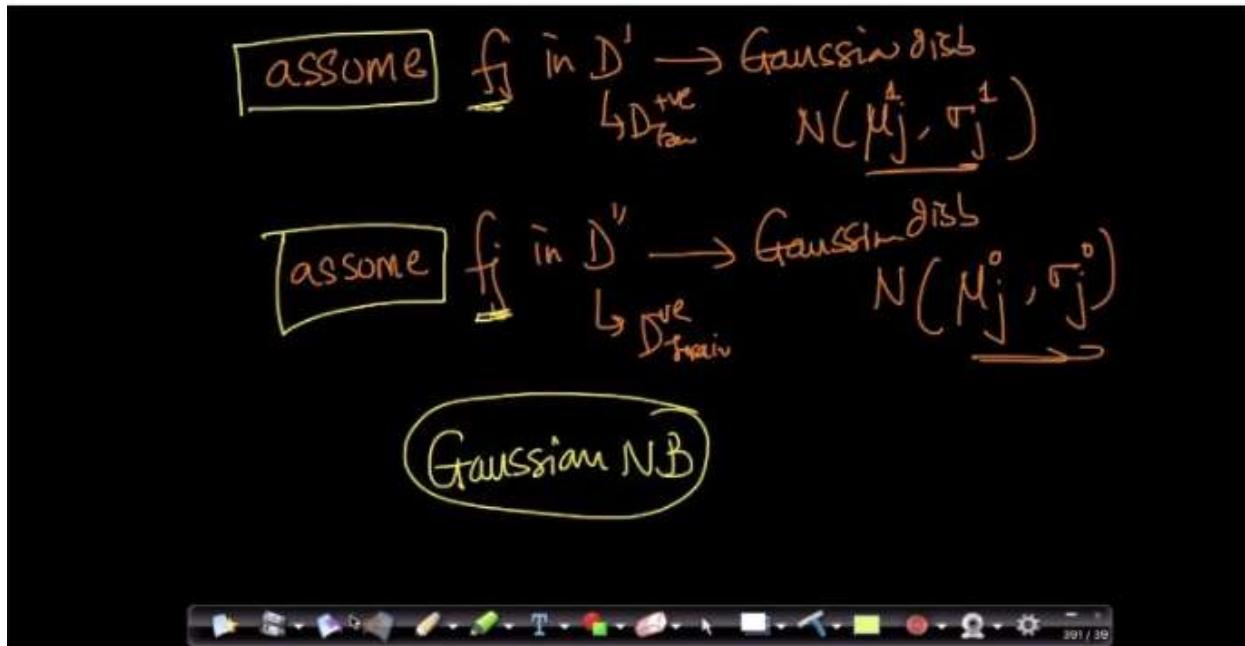
Timestamp 3:56

Consider a dataset where we have d features out of which say feature f_j is real valued. Now let's divide our dataset into groups one for all the positive labels(n_1) and another for all the negative labels(n_2). Consider a positive data point x_i , and it's j th feature i.e. x_{ij} , likelihood for this term can be written as $p(x_{ij} | y_i = 1)$. Now how should we calculate this term?



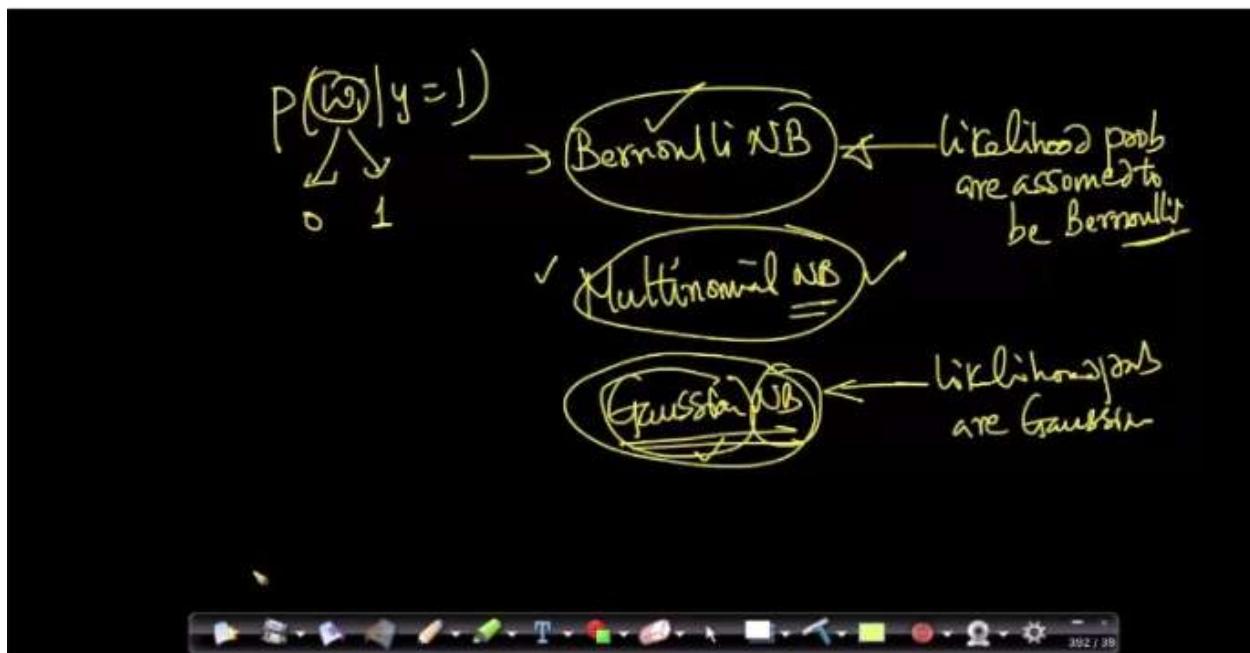
Timestamp 7:16

In order to calculate this term $p(x_{ij} | y_i = 1)$, where x_{ij} corresponds to feature j of datapoint x_i , we will first take only the positive data points from the whole dataset, let's call this D^+ . Now from all the data points of D^+ , we take feature f_j and draw a pdf corresponding to its values as shown in the image above. Now, in order to get a probability for any value say x_{ij} , we can calculate the pdf value from the pdf curve as shown in the image above.



Timestamp 10:08

When dealing with numerical features we assume that these values follow a Gaussian Distribution. For the positive dataset D^+ , we assume that the real valued feature f_j containing only the values from D^+ follows a Gaussian Distribution with mean μ_j^+ and standard deviation σ_j^+ , computed only from the values of the D^+ . Similar steps can be followed for the negative dataset, as shown in the image above. This is called a Gaussian NB.



Timestamp 12:08

While dealing with words w_i , it could take two values either 0 or 1. It follows Bernoulli Distribution. It is also called Bernoulli NB. So, we can figure out what distribution a feature is following and based on the pdf/pmf of these distributions and we can calculate likelihood probabilities.

32.12 Multiclass classification

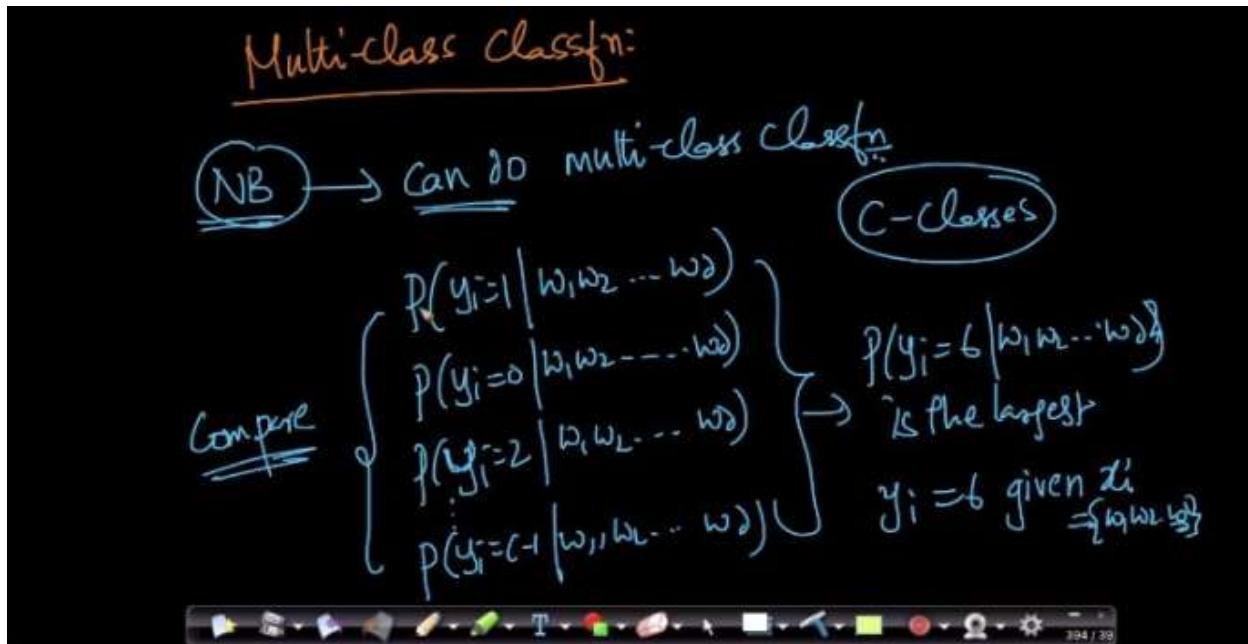
Multi-class classfn:

NB → can do multi-class classfn

C-classes

Compare

$$\left\{ \begin{array}{l} p(y_i=1 | w_1, w_2, \dots, w_d) \\ p(y_i=0 | w_1, w_2, \dots, w_d) \\ p(y_i=2 | w_1, w_2, \dots, w_d) \\ \vdots \\ p(y_i=C-1 | w_1, w_2, \dots, w_d) \end{array} \right\} \rightarrow \begin{array}{l} p(y_i=6 | w_1, w_2, \dots, w_d) \\ \text{is the largest} \\ y_i = 6 \text{ given } z_i = \{w_1, w_2, \dots, w_d\} \end{array}$$



Timestamp 1:44

Naive Bayes can easily handle Multi Class Classification. We just need to calculate the probability for each of the classes, compare them and then select the class which has the highest probability.

32.13 Similarity or Distance Matrix

✓ Similarity matrix or Distance Matrix

NB → classification → distance based method
NB → probability based method

Distance | Similarity matrix ← KNN

↳ Cannot we use dist/Sim matrices

$$p(y_i=1 | f_1 f_2 \dots) = p(y_i=1) \prod_{j=1}^k p(f_j | y_i=1)$$

Timestamp 2:15

Similarity or Distance matrix is usually used when it is difficult to represent a data point in the form of a vector, like we saw in KNN while doing pharma example. Naive Bayes cannot handle similarity or distance matrix in general, because it is not a distance based method rather a probability based method here we do not calculate the distance between two data points, rather we find out their likelihood probabilities.

32.14 Large Dimensionality

Large dimensionality

NB → Text classification → high dim

↳ log-probabilities

$$p(y=1 | w_1 w_2 \dots w_k) = p(y=1) \prod_{i=1}^k p(w_i | y=1)$$

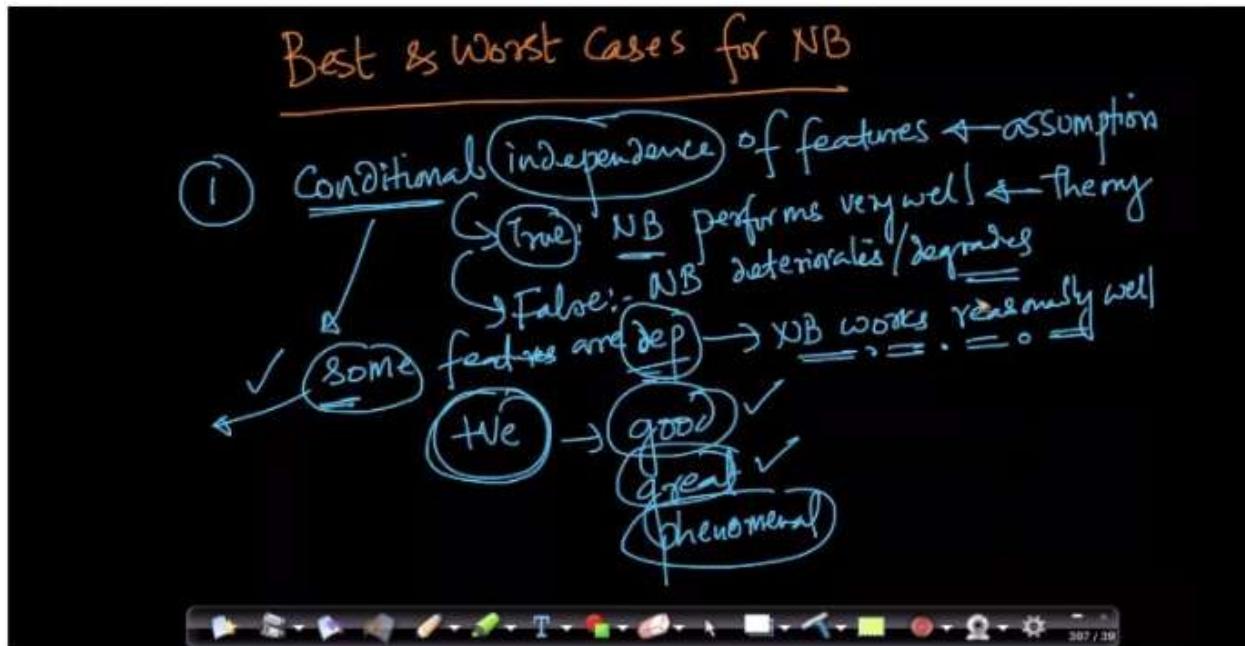
numerical stability

0 < x < 1

Timestamp 1:40

Naive Bayes can easily handle data of large dimensions. While dealing with text classification we had text data which itself is of very high dimension. Only thing to keep in mind is to use log-probabilities to avoid numerical instability.

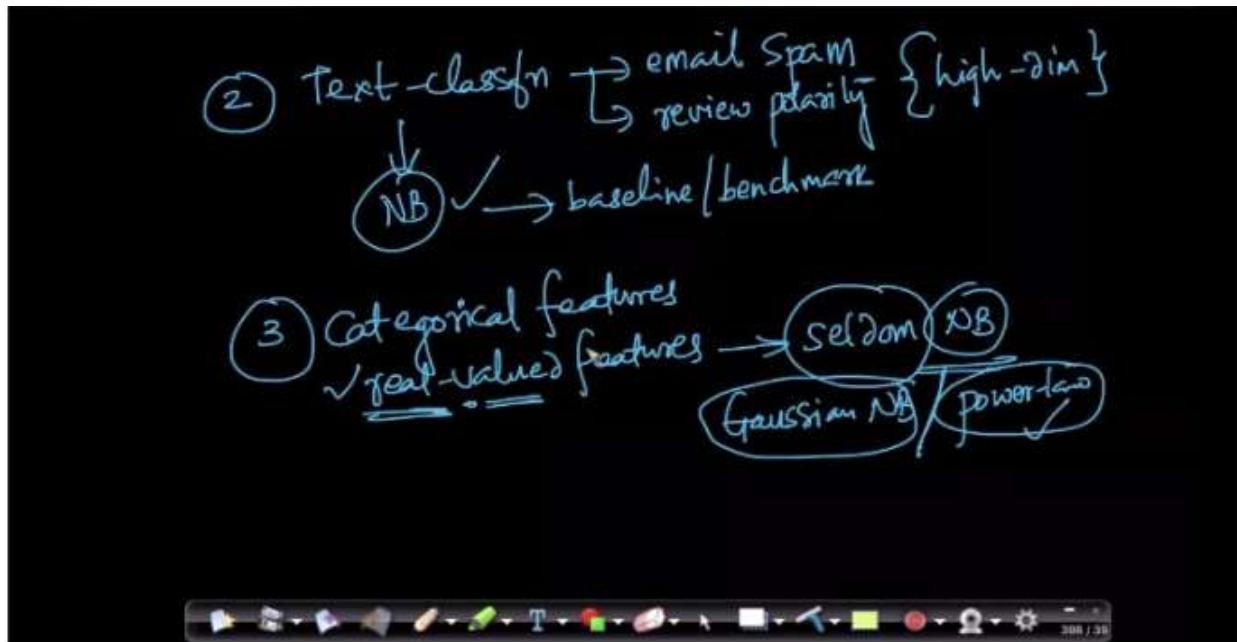
32.15 Best and worst cases



Timestamp 2:50

Now let's look at the cases where NB performs best and worst.

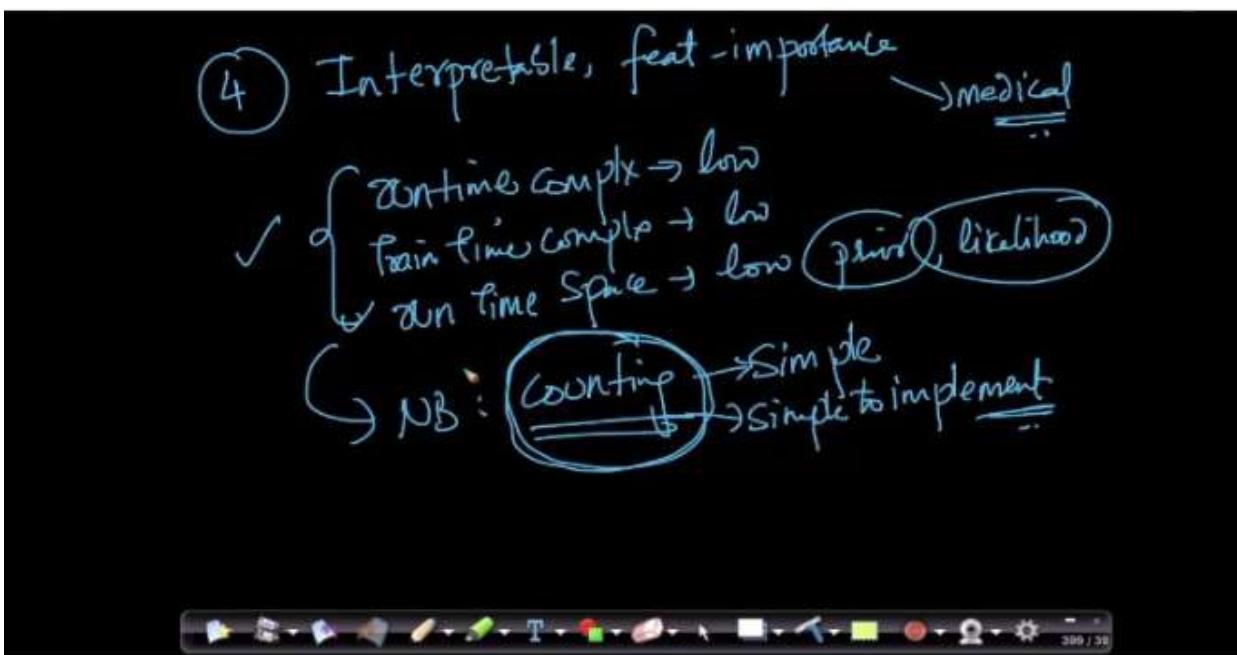
1. If the features are conditionally independent (the fundamental assumption of NB), then our model performs very well. Even if some of the features are slightly correlated then also our model performs reasonably well. Consider a positive case where words like good, great, phenomenal exist, even though these words are related our model will perform reasonably well in this situation. There are some mathematical proofs justifying this.



Timestamp 4:37

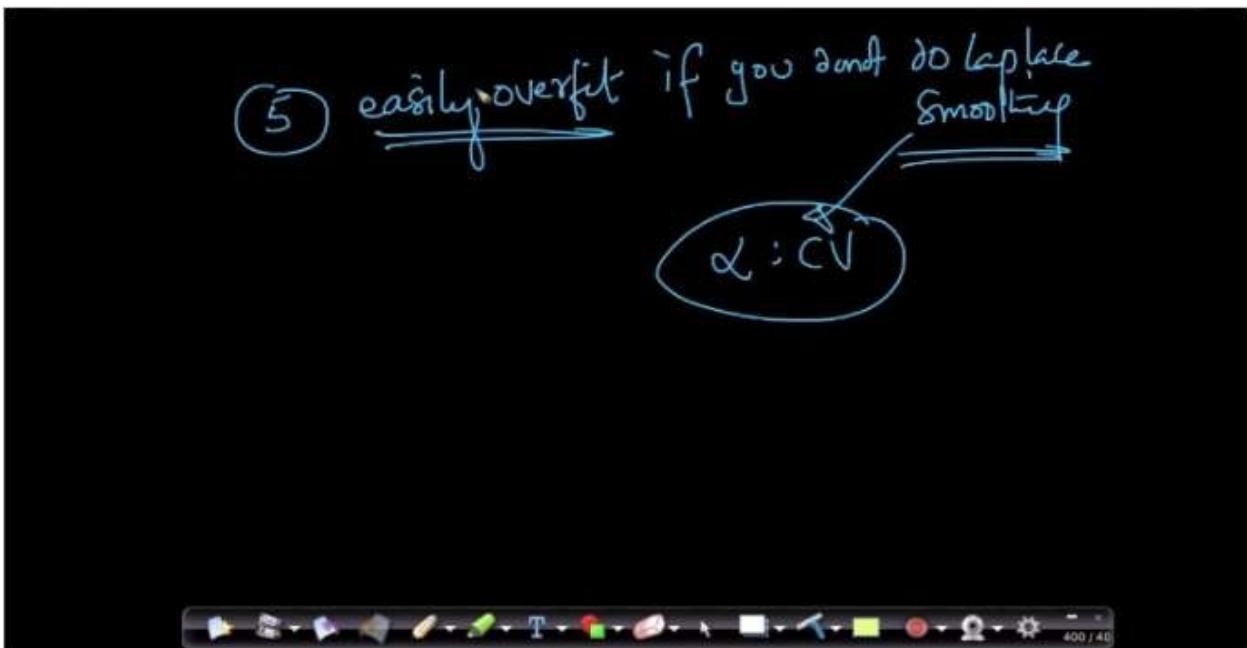
2. NB is extremely useful in dealing with high dimensional data as we saw during text classification. It serves as a good benchmark for problems like spam detection, polarity of a review, etc.

3. NB performs very well with categorical features. For real valued features we can use Gaussian NB. However it is seldom used at least in internet domains.



Timestamp 6:27

4. NB is highly interpretable. Feature importance can also be easily calculated. This makes it suitable for problems where interpretability is important like medical purposes. Also the time and space complexities for NB is very low when compared to KNN, which makes it useful in low latency solutions. It is also very easy to implement.



Timestamp 7:00

5. One of the disadvantages of NB is that it can easily overfit. In order to tackle this we need to do laplace smoothing. We can find the correct value of α using cross-validation.

32.16 Code example

We can see the scikit learn's implementation of Naive Bayes here =>

https://scikit-learn.org/stable/modules/naive_bayes.html

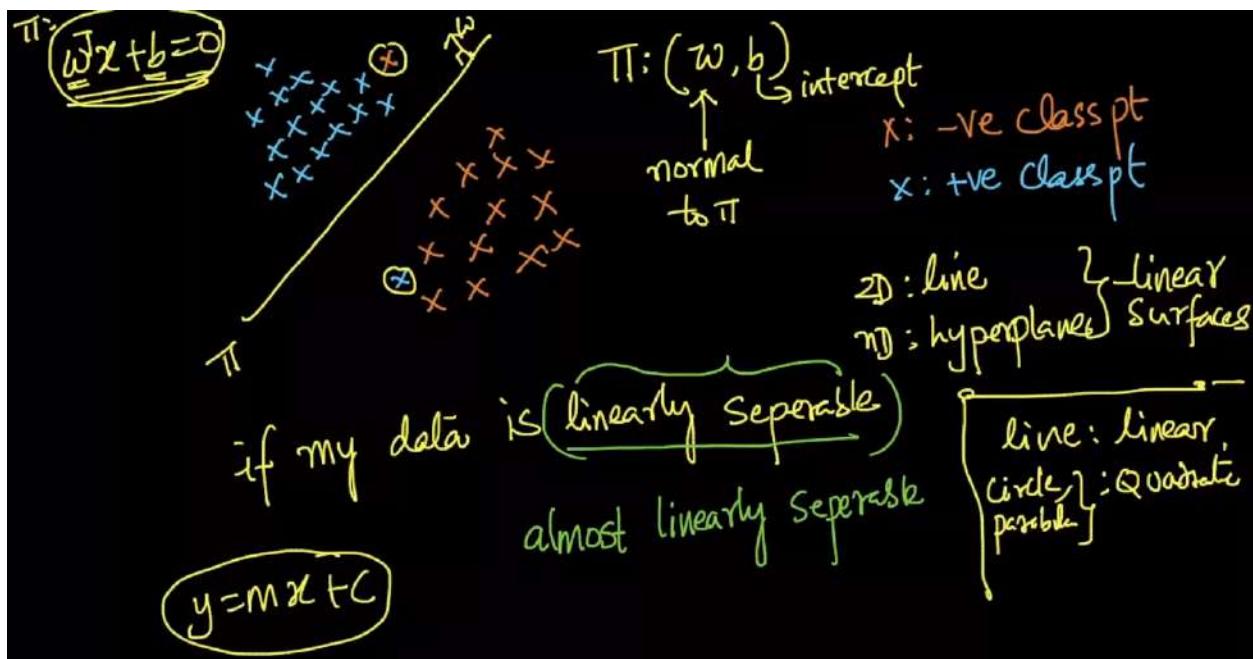
33.1 - Geometric Intuition of Logistic Regression

Logistic Regression is one of the classification techniques used in Machine Learning. So far we have seen Naive Bayes, which was a Probability oriented technique, whereas Logistic Regression is a geometry-oriented technique.

Logistic Regression can be derived using any one of the 3 below perspectives. They are

- a) Geometry
- b) Probability
- c) Loss Function

Let us assume we have the points belonging to 2 classes(positive and negative), as shown in the figure below explained starting from the timestamp 3:20.



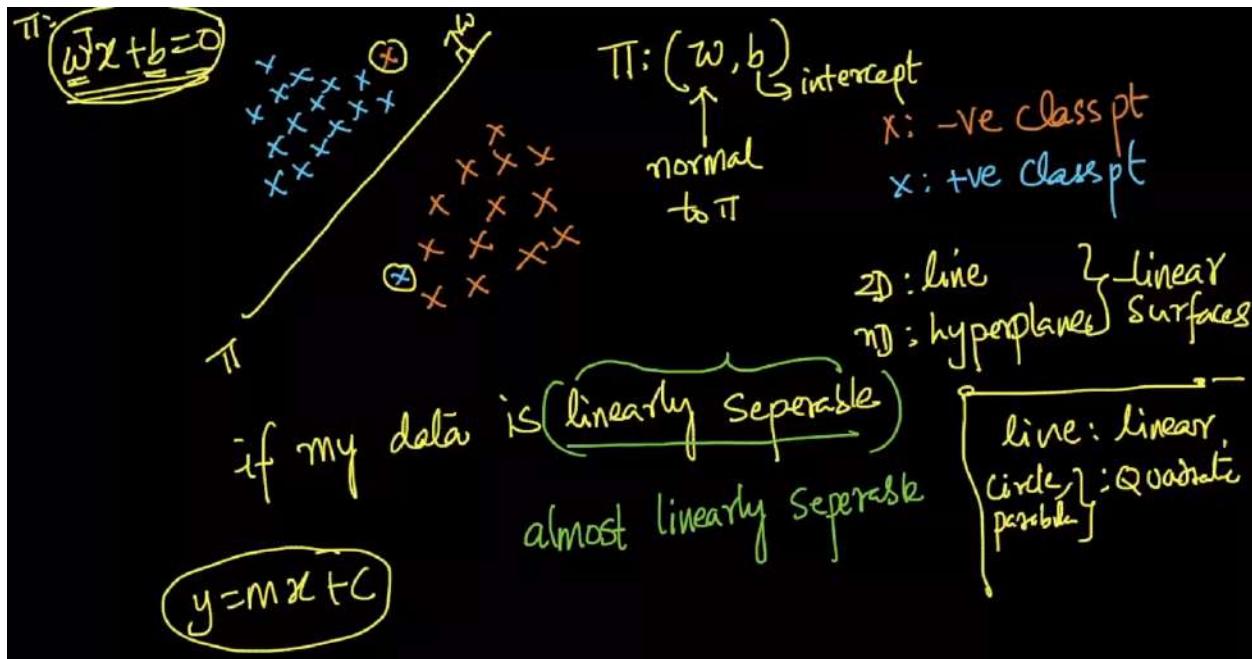
If the data points of different classes can be separated by using a linear surface, then we say the data is linearly separable. The linear surfaces in different dimensions are called as mentioned below

2D → Line

3D → Plane

nD → Hyperplane

Circle, Parabola, Ellipse, Sphere are the examples of Quadratic Surfaces.



In the above representation of the points,

$\pi \rightarrow$ Hyperplane

$w \rightarrow$ Perpendicular to the plane

The equation of the hyperplane ' π ' is represented in the form of ' w ' as

$w^T x + b = 0$ where $b \rightarrow$ intercept.

Here $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}^1$.

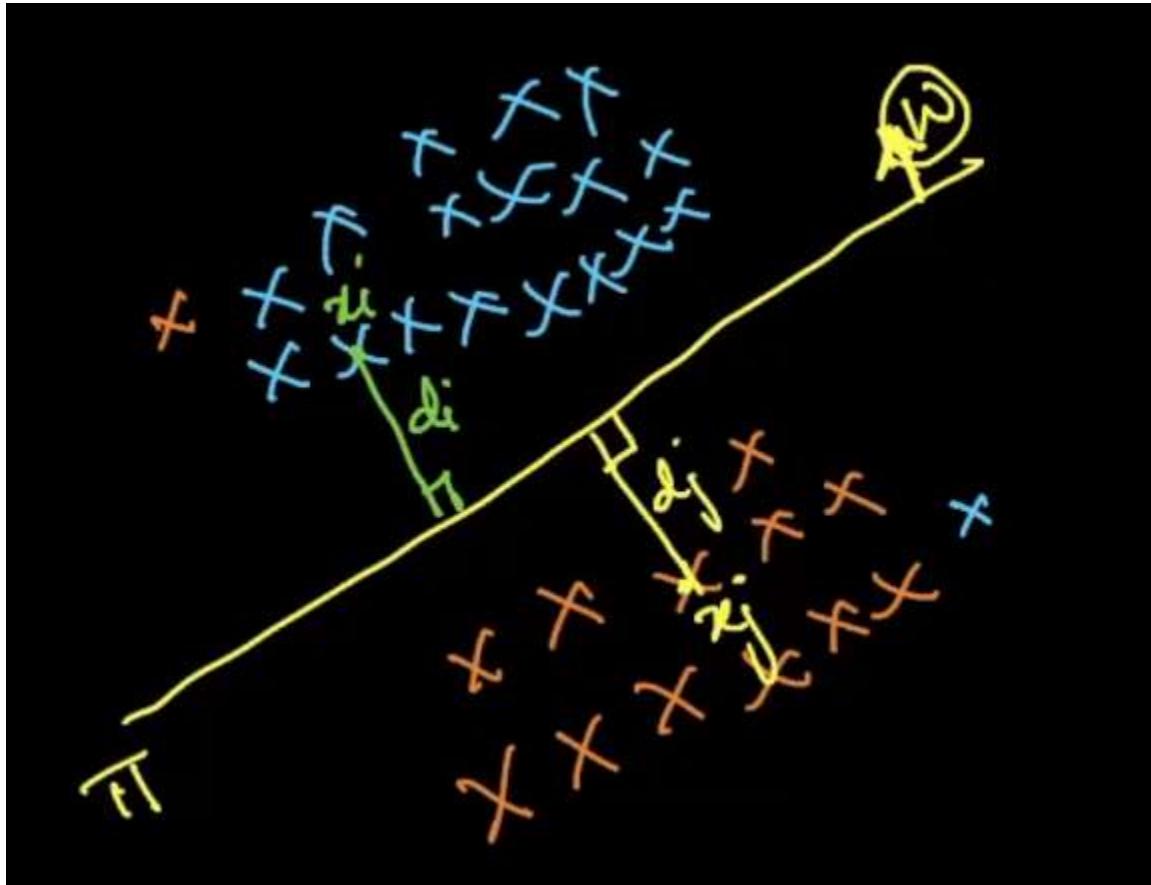
If the hyperplane ' π ' passes through the origin, then $b=0$ and the equation becomes $w^T x = 0$.

Assumption of Naive Bayes: All the features are conditionally independent of each other.

Assumption of K-NN: Neighborhood is the same as the query point.

Assumption of Logistic Regression: All the classes in the dataset are almost/perfectly linearly separable.

Geometric Intuition



For all the positive points, let $y_i = +1$

For all the negative points, let $y_i = -1$

So far, we have seen

If $y_i = 1 \rightarrow$ class = +ve

If $y_i = 0 \rightarrow$ class = -ve

But for now,

If $y_i = 1 \rightarrow$ class = +ve

If $y_i = -1 \rightarrow$ class = -ve

So for Logistic Regression, $y_i \in \{-1, +1\}$

Distance of the point ' x_i ' to the hyperplane ' π ' = $d_i = w^T x_i / \|w\|$

Let us assume ' w ' is a unit vector, so then $\|w\| = 1$.

So now the distance ' d_i ' becomes $w^T x_i$

From the above figure, as 'w' and ' x_i ' are on the same side of the hyperplane ' π ', we have $w^T x_i > 0$

As 'w' and ' x_j ' are on the opposite sides of the hyperplane ' π ', we have $w^T x_j < 0$

So for any point ' x_q ', the classifier works as

If $w^T x_q > 0$, then $y_q = +1$

If $w^T x_q < 0$, then $y_q = -1$

Different Cases in classification using Logistic Regression

Case 1: If $y_i = +1$ and $w^T x_i > 0$

It means the point ' x_i ' is actually a positive point and is also classified as positive. Hence the classifier has classified this data point correctly.

Case 2: If $y_i = -1$ and $w^T x_i < 0$

It means the point ' x_i ' is actually a negative point and is also classified as negative. Hence the classifier has classified this data point correctly.

Case 3: If $y_i = +1$ and $w^T x_i < 0$

It means the point ' x_i ' is actually a positive point but is classified as negative. This point has been misclassified.

Case 4: If $y_i = -1$ and $w^T x_i > 0$

It means the point ' x_i ' is actually a negative point but is classified as positive. This point has been misclassified.

Note

If a point (x_i, y_i) is classified correctly, then $y_i * (w^T x_i) > 0$.

If a point (x_i, y_i) is misclassified, then $y_i * (w^T x_i) < 0$.

For a classifier to be very good, the number of misclassified points should be minimum, and the number of correctly classified points should be maximum.

So the objective is to have as many points as possible with $y_i^*(w^T x_i) > 0$.

So we need to maximize $\sum_{i=1}^n y_i^*(w^T x_i)$

Where 'n' → total number of points in the training dataset (D_{Train}).

But from the given dataset, ' x_i ' and ' y_i ' are fixed. We can only find ' w ' that is variable. So we can find an optimal ' w ' that could maximize $\sum_{i=1}^n y_i^*(w^T x_i)$.

Let w^* be the optimal ' w ' that could maximize $\sum_{i=1}^n y_i^*(w^T x_i)$.

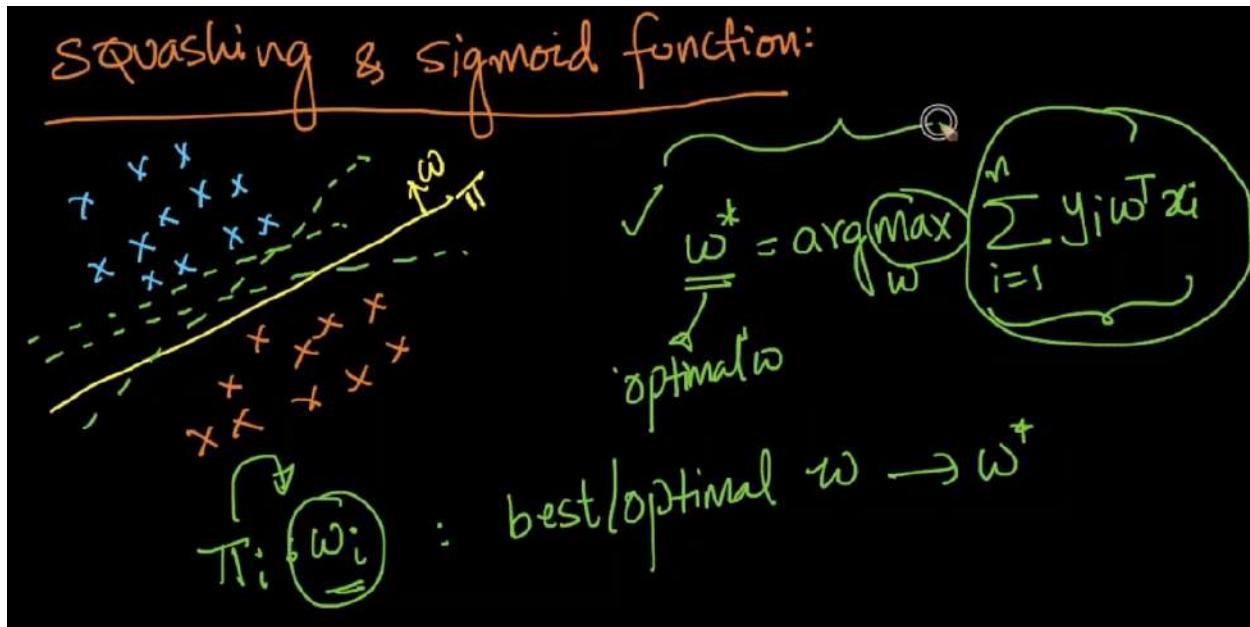
$w^* = \arg\max(\sum_{i=1}^n y_i^*(w^T x_i))$

This is the mathematical optimization problem we have to solve, in order to obtain the optimal ' w '.

$y_i^*(w^T x_i) = +ve \rightarrow$ If ' x_i ' is correctly classified.

$y_i^*(w^T x_i) = -ve \rightarrow$ If ' x_i ' is mis-classified.

33.2 - Sigmoid Function: Squashing



The objective of Logistic Regression is to find a plane ' π ' with an optimal ' w ' that could maximize $\sum_{i=1}^n y_i^*(w^T x_i)$

The function that has to be maximized is $\arg\max \sum_{i=1}^n y_i^*(w^T x_i)$
 $w^T x_i \rightarrow$ distance from the point ' x_i ' to the plane ' π '.

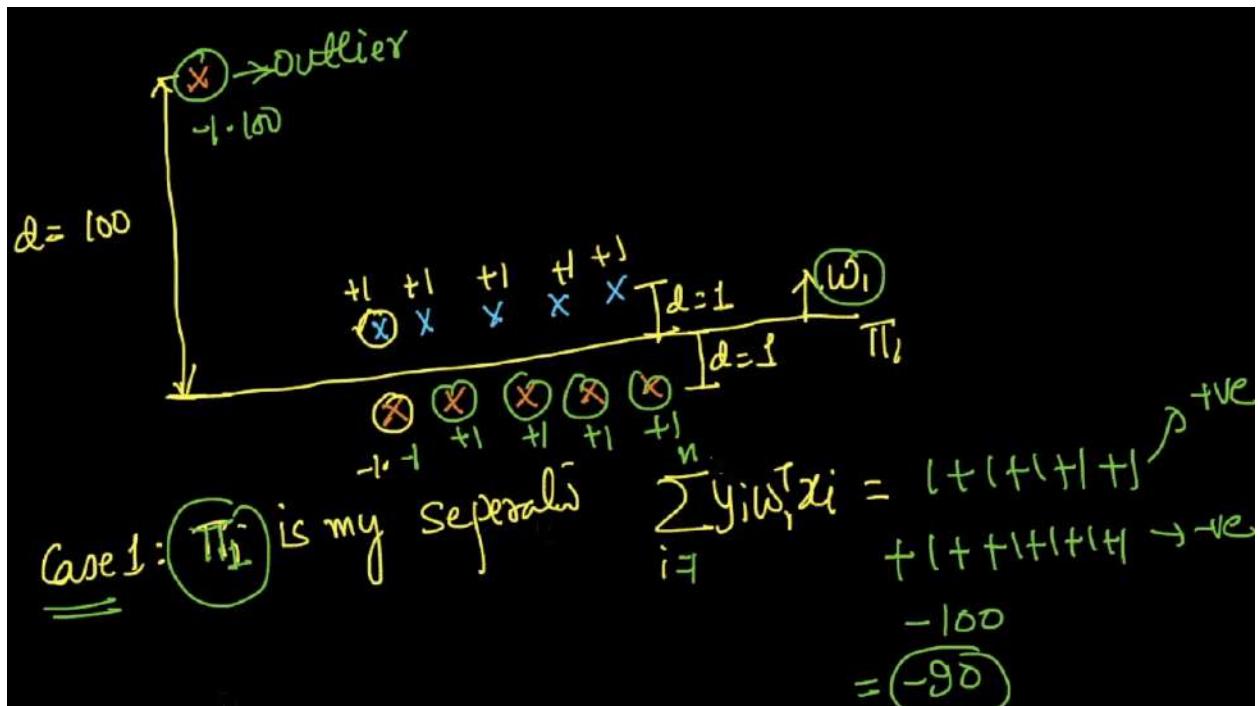
$$y_i \in \{-1, +1\}$$

If $y_i^*(w^T x_i)$ is +ve \rightarrow ' π ' as defined by ' w ' correctly classifies ' x_i '.

If $y_i^*(w^T x_i)$ is -ve \rightarrow ' π ' as defined by ' w ' incorrectly classifies ' x_i '.

We have to maximize the number of correctly classified points and minimize the incorrectly classified points.

Case 1



In the above figure, we can see the 5 positive points are present in the same side as that of the vector 'w'. We also can see the 5 negative points present on the opposite side of the vector 'w'.

So for all the given 5 '+ve' points, $w^T x_i = 1$ and $y_i = 1$.

So for all the given 5 '+ve' points, $y_i^*(w^T x_i) = 1 * 1 = 1$

For all the given 5 '-ve' points, $w^T x_i = -1$ and $y_i = -1$

So for all the given 5 '-ve' points, $y_i^*(w^T x_i) = (-1) * (-1) = 1$

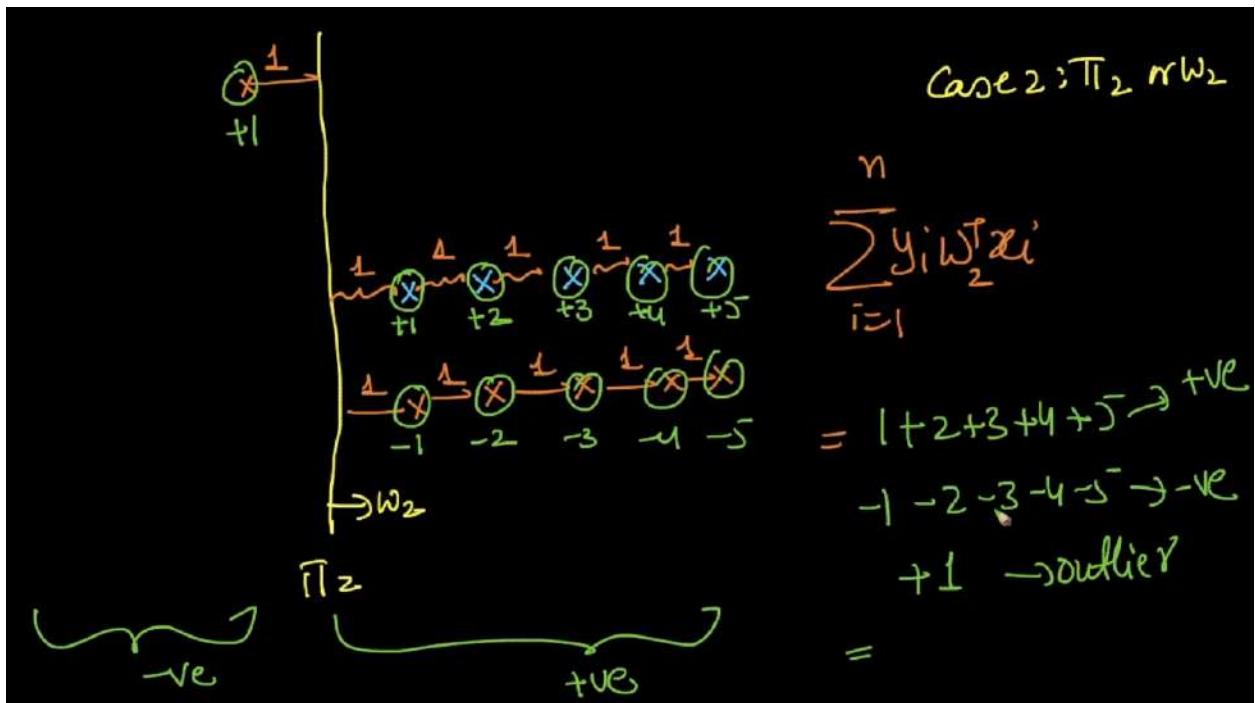
We have a '-ve' point present on the same side as vector 'w' and is located at a distance of 100 units from the plane ' π '.

So $y_i^*(w^T x_i) = -1 * 100 = -100$

So now,

$$\sum_{i=1}^n y_i^*(w^T x_i) = (1+1+1+1+1) + (1+1+1+1+1) + (-100) = -90$$

Case 2



So here,

$$\sum_{i=1}^n y_i^*(w^T x_i) = (1+2+3+4+5)+(-1-2-3-4-5)+1 = 1$$

As we know the objective of Logistic Regression is to maximize the value of $\sum_{i=1}^n y_i^*(w^T x_i)$, with the planes ' π_1 ' and ' π_2 ', we get the sum values as -90 and 1 respectively. So if we want to go with the maximum value of $\sum_{i=1}^n y_i^*(w^T x_i)$, we have to go for case 2.

But from the geometric intuition point, in case 1, out of 11 points, 10 of them are classified correctly, whereas in case 2, only 6 of them are classified correctly.

$$\text{Accuracy in case 1} = 10/11 = 0.909$$

$$\text{Accuracy in case 2} = 6/11 = 0.545$$

Intuitively we can say ' π_1 ' is better than ' π_2 '. We are getting a larger value of $\sum_{i=1}^n y_i^*(w^T x_i)$ in case 2, just because of the outlier point, which is very far away, and it makes us prefer the plane ' π_2 '. Here one single outlier is impacting our model drastically.

So the maximization of summation of signed distances is much prone to the outliers. It means the outliers can impact a lot.

So we have to modify this formulation, by modifying the function $\arg\max_w \sum_{i=1}^n y_i^*(w^T x_i)$ in such a way that the outliers do not impact our model. For this purpose, we use a technique called **Squashing**.

Squashing

The idea of squashing is instead of using the signed distances as they are for all the points,

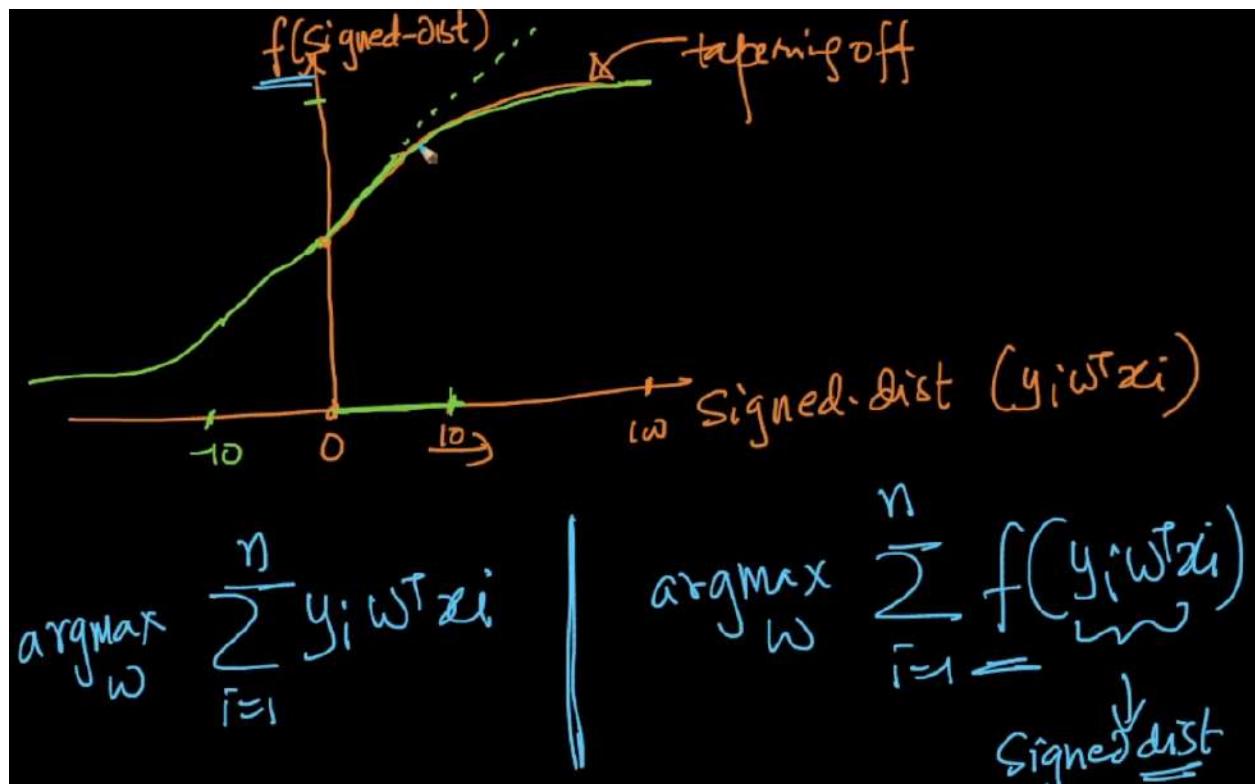
If signed distance is small → Use the signed distance as it is.

If signed distance is large → Make it a smaller value

So in our example,

If the signed distance of the normal point to the plane ' π_1 ' is 1 unit, then we can use it as it is. Whereas if the signed distance of an outlier point to the plane ' π_1 ' is 100 units, we have to reduce it to a smaller value.

How to reduce the large value to a smaller one?



Let us assume we have all the values of signed distances on X-axis (ie., the values of $y_i^*(w^T x_i)$). Let the 'Y' axis be a function of the signed distances (ie., function of $y_i^*(w^T x_i)$).

Now we should make sure that from $y_i^*(w^T x_i) = 0$, as the value of $y_i^*(w^T x_i)$ increases, we want $f(y_i^*(w^T x_i))$ to increase linearly.

This function $f(y_i^*(w^T x_i))$ needs to increase linearly for smaller positive values of $y_i^*(w^T x_i)$ and should taper-off for larger positive values of $y_i^*(w^T x_i)$. Here tapering-off means becoming constant.

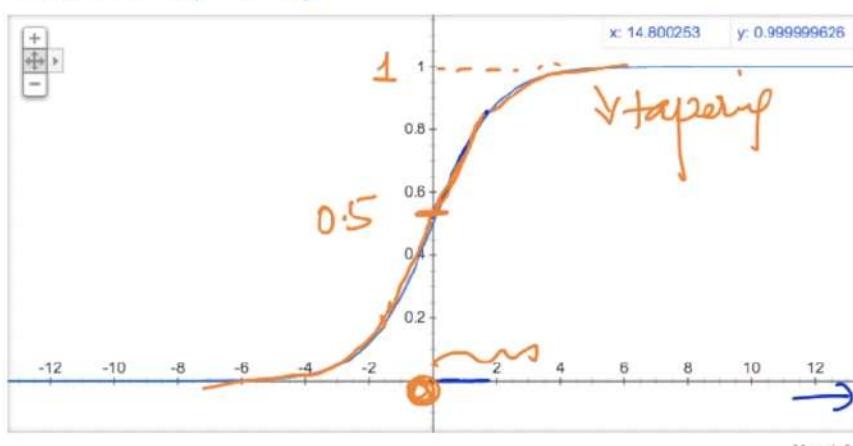
We should make sure this function increases linearly upto a certain threshold value of $y_i^*(w^T x_i)$ on the positive side of the 'X' axis and then gets tapered off. Similarly on the negative side of the 'X' axis, the function should decrease linearly upto a threshold value and then should get tapered off.

So here in order to get rid of the problem of outliers, we are converting the problem of $\arg\max_w \sum_{i=1}^n y_i^*(w^T x_i)$ to $\arg\max_w \sum_{i=1}^n f(y_i^*(w^T x_i))$.

Here we actually need a function $f(y_i^*(w^T x_i))$ in such a way that on the positive side of the axis of $y_i^*(w^T x_i)$, it increases linearly upto a certain threshold value and then gets tapered off. Similarly on the negative side of the axis of $y_i^*(w^T x_i)$, it decreases linearly upto a certain threshold value and then gets tapered off. Such a function satisfying our requirement is **Sigmoid Function**.

About 16,10,00,000 results (0.35 seconds)

Graph for $1/(1+e^{-x})$



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

✓ Max : 1
✓ Min : 0
 $\sigma(0) = 0.5$

$x = \text{Signed dist}$

Sigmoid function: $\sigma(x) = 1/(1+\exp(-x))$

So now we shall change the mathematical formulation to $\arg\max_w \sum_{i=1}^n \sigma(y_i^*(w^T x_i))$

Maximum value of $\sigma(x) = 1$

Minimum value of $\sigma(x) = 0$

$\sigma(0) = 0.5$

So in our problem, we have seen outlier present at a distance of 100 units. Here as the maximum value of the sigmoid function is 1, the distance of outlier instead of 100 units, gets tapered off to 1 unit.

If a point (x_i, y_i) lies on the hyperplane ' π ', then $w^T x_i = 0$.

$$P(y_i^*(w^T x_i)) = P(0) = 0.5$$

$$\text{So } P(y_i=1) = 0.5$$

If a point (x_i, y_i) lies on the same side as 'w' of the hyperplane ' π ', then $w^T x_i > 0$.

$$\sigma(y_i^*(w^T x_i)) > 0.5$$

$$\text{So } P(y_i=1) > 0.5$$

If a point (x_i, y_i) lies on the opposite side as 'w' of the hyperplane ' π ', then $w^T x_i < 0$.

$$\sigma(y_i^*(w^T x_i)) < 0.5$$

$$\text{So } P(y_i=1) < 0.5$$

Note: In all the above 3 cases, (x_i, y_i) is a '+ve' point.

So we have a good probabilistic interpretation with Sigmoid function. Coming back to our problem, our actual mathematical formulation was to maximize the sum of signed distances (ie., $\text{arg-max}_w \sum_{i=1}^n y_i^*(w^T x_i)$), but it was severely impacted by the outliers.

Hence we have gone for sigmoid function which has got the below properties required for our problem.

- 1) Linear Behaviour for small values of $y_i^*(w^T x_i)$.
- 2) Tapering Behaviour for large values of $y_i^*(w^T x_i)$.
- 3) Nice Probabilistic Interpretation

So now our mathematical formulation is to maximize the sum of transformed signed distances.

So the optimal value of 'w' is the value that maximizes

$$w^* = \arg\max_w (\sum_{i=1}^n \sigma(y_i^*(w^T x_i))) = \arg\max_w (\sum_{i=1}^n 1/(1+\exp(-y_i^*(w^T x_i))))$$

This above mathematical formulation is less impacted by the presence of outliers, when compared to the sum of signed distances. This function is also called Squashing and it helps us in getting rid of outliers. Here squashing reduces the $f(y_i^*(w^T x_i))$ values which lie in $(-\infty, \infty)$ to the values in $[0,1]$.

33.3 - Mathematical Formulation of Objective Function

The optimization problem is $w^* = \arg\max_w \sum_{i=1}^n 1/(1+\exp(-y_i^*(w^T x_i)))$
Here we shall use monotonic functions in this optimization problem.

A function $g(x)$ is said to be monotonic

- If $g(x)$ increases with an increase in ' x ', then $g(x)$ is called monotonically increasing function.
- If $g(x)$ decreases with an increase in ' x ', then $g(x)$ is called monotonically decreasing function.

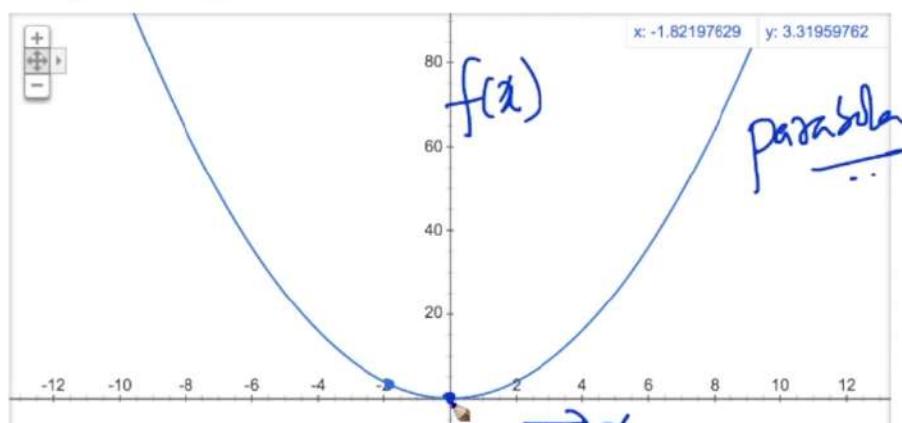
For all $x_1 > x_2$, if $g(x_1) > g(x_2)$, then we can say $g(x)$ is monotonically increasing function.

For all $x_1 > x_2$, if $g(x_1) < g(x_2)$, then we can say $g(x)$ is monotonically decreasing function.

For example, $\log(x)$ is defined only for $x > 0$ and is an example of monotonically increasing function. Let us assume we have an optimization problem as below

$$x^* = \arg \min_x (x^2)$$

Graph for x^2



$$f(x) = x^2$$

The curve of x^2 is a parabola $y=mx^2$ passing through origin. So here we see the minimum value of this curve occurs at $x=0$. So the optimal value of ' x ' in order to minimize this optimization problem is $x=0$.

The curve $y=x^2$ is

- Monotonically increasing for $x > 0$
- Monotonically decreasing for $x < 0$

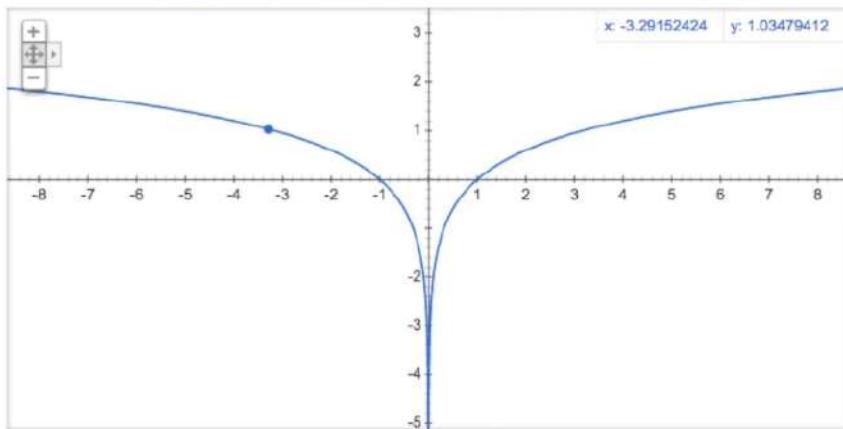
So we have assumed $g(x) = \log(x)$ which is increasing function. So far we have worked on the optimization problem of $\mathbf{x}^* = \arg\min_x f(x)$

$$f(x) = x^2$$

Now we shall look at the optimization problem of $\mathbf{x}^* = \arg\min_x g(f(x))$
As $g(x) = \log(x) \rightarrow g(f(x)) = g(x^2) = \log(x^2)$. The graph of $\log(x^2)$ is given as

About 24,90,00,000 results (0.42 seconds)

Graph for $\log(x^2)$



$$\begin{aligned} g(f(x)) \\ = \log(x^2) \end{aligned}$$

The minimum value of $\log(x^2)$ also occurs at $x=0$.

Let us assume we have a function $f(x)$ and a monotonic function $g(x)$, then

$$\arg\min_x f(x) = \arg\min_x g(f(x))$$

$$\arg\max_x f(x) = \arg\max_x g(f(x))$$

These above two properties are valid only if $g(x)$ is a monotonic function. Coming back to our original problem,

$$\mathbf{w}^* = \arg\max_w \sum_{i=1}^n 1/(1+\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i)))$$

Let us consider $g(x) = \log(x)$ which is a monotonic function. So from the property of monotonic function, we can write the optimization problem as

$$\mathbf{w}^* = \arg\max_w \sum_{i=1}^n \log(1/(1+\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i))))$$

We have a formula that $\log(1/x) = -\log(x)$

So the optimization problem now becomes

$$\mathbf{w}^* = \arg\max_w \sum_{i=1}^n -\log(1+\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i)))$$

Now $\arg\max(f(x)) = \arg\min(-f(x))$

So the optimization $\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{i=1}^n -\log(1+\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i)))$ changes to $\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n \log(1+\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i)))$

Finally this is the optimization problem of Logistic Regression with the class labels +1 and -1.

If we do not have the term '1' in the optimization problem, then the optimization problem becomes

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n \log(\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i)))$$

Here the 'log' and 'exp' get cancelled out because $\log_e e^x = x$.

So now the optimization problem becomes

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n (-y_i^*(\mathbf{w}^T \mathbf{x}_i))$$

So if we remove the '-ve' sign, then arg-min becomes arg-max, and the optimization problem becomes

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{i=1}^n (y_i^*(\mathbf{w}^T \mathbf{x}_i))$$

So we can say that the optimization of Logistic Regression is a slight change made to the optimization of summation of signed distances.

But the optimization of summation of the signed distances is more impacted by the outliers. Hence we go with the optimization of the sigmoid function, as it is not much affected by the outliers.

33.4 - Weight Vector

So far we have seen the optimization problem of

$$w^* = \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i(w^T x_i)))$$

The optimal value w^* is called the weight vector. This weight vector is d-dimensional same as our data point $x_i \in R^d$.

$$w^* = \langle w_1, w_2, w_3, \dots, w_d \rangle$$

Here as we have 'd' features, we have the weight component associated with each feature in w^* . That's the reason we call it the weight vector.

In Logistic Regression, the decision about the class label ' y_q ' for a data point ' x_q ' is made on the below criteria.

If $w^T x_q > 0$, then $y_q = 1$

If $w^T x_q < 0$, then $y_q = -1$

When it comes to the probabilistic interpretation, if we want to know $P(y_q=1)$, we have to compute the value of $\sigma(w^T x_q)$.

Interpretation of w^*

Let ' w_i ' → i^{th} component of the weight vector (ie., the weight component associated with the i^{th} feature in the dataset)

' x_{qi} ' → i^{th} feature value of the data point ' x_q '.

- 1) If w_i = '+ve' and if ' x_{qi} ' increases, then the value of the product ($w_i \cdot x_{qi}$) also increases. It means the value of ($\sum_{i=1}^d w_i * x_{qi}$) also increases. There by the $\sigma(w^T x_q)$ increases, and finally $P(y_q=1)$ increases.
- 2) If w_i = '-ve' and if ' x_{qi} ' increases, then the value of the product ($w_i \cdot x_{qi}$) decreases. It means the value of ($\sum_{i=1}^d w_i * x_{qi}$) decreases. There by the $\sigma(w^T x_q)$ decreases, and finally $P(y_q=1)$ also decreases. The value of $P(y_q = -1)$ increases.

- 3) If w_i = 've' and if ' x_{qi} ' decreases, then the value of the product $(w_i \cdot x_{qi})$ decreases. It means the value of $(\sum_{i=1}^d w_i * x_{qi})$ decreases. There by the $\sigma(w^T x_q)$ decreases, and finally $P(y_q=1)$ also decreases. The value of $P(y_q = -1)$ increases.
- 4) If w_i = '-ve' and if ' x_{qi} ' decreases, then the value of the product $(w_i \cdot x_{qi})$ increases. It means the value of $(\sum_{i=1}^d w_i * x_{qi})$ increases. There by the $\sigma(w^T x_q)$ increases, and finally $P(y_q=1)$ also increases. The value of $P(y_q = -1)$ decreases.

Note

In Logistic Regression formulation of the optimization problem, we have not strictly imposed the constraint that 'w' has to be a unit vector. The mathematical formulation will work even if 'w' is a non unit vector perpendicular to the hyperplane.

The only difference would be that the distance from any point ' x_i ' to the hyperplane would be $w \cdot x_i / \|w\|$ instead of just $w \cdot x_i$. So the resulting weight vector after solving the optimization problem while being perpendicular to the hyperplane need not be a unit vector always.

33.5 - L2 Regularization: Overfitting and Underfitting

The optimization problem we have been working on so far is

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n \log(1+\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i)))$$

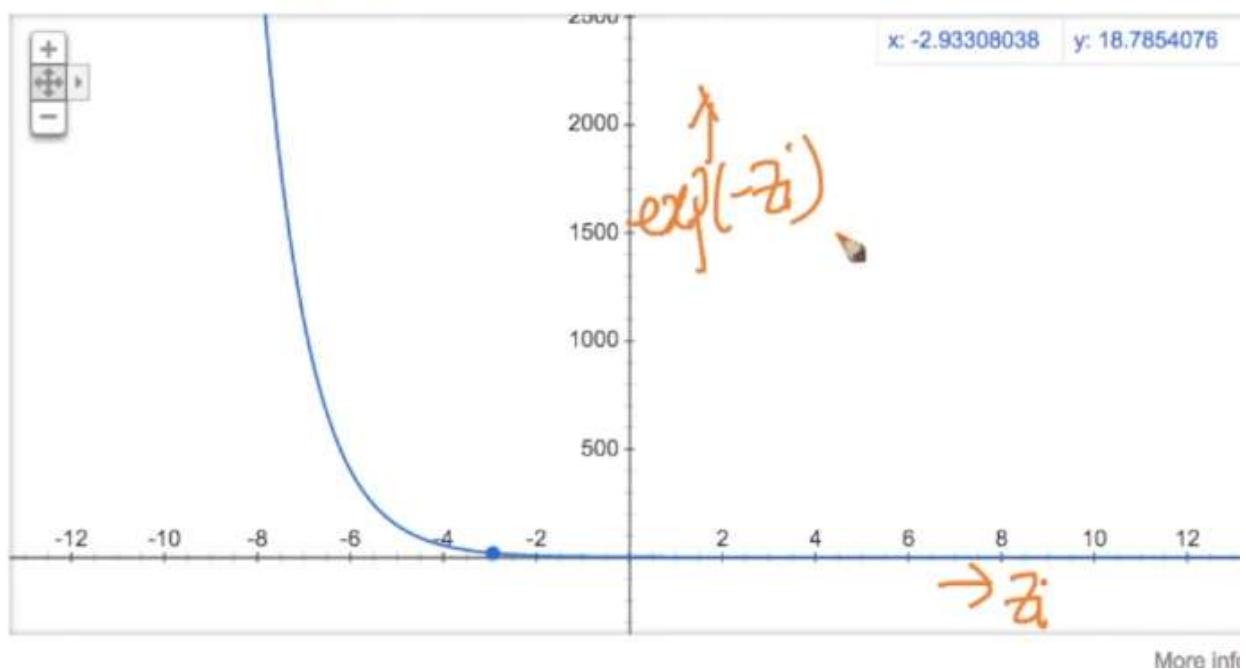
Let us denote $-y_i^*(\mathbf{w}^T \mathbf{x}_i) = z_i$

The optimization problem can now be written as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n \log(1+\exp(-z_i))$$

Let us have a look at the plot of $\exp(-z_i)$ below.

Graph for $\exp(-x)$



For whatever value of ' z_i ', the value of $\exp(-z_i)$ is always greater than or equal to 0.

$$\exp(-z_i) \geq 0$$

Now let us look into the term $\log(1+\exp(-z_i))$, we know that $\log(1)=0$

And if we add any term to '1' and apply logarithm, then the value of it would be greater than or equal to 0, provided if the newly added term is non-negative.

So if $\log(1+\delta) \geq 0$ (only if $\delta \geq 0$)

So ultimately $\log(1+\exp(-z_i)) \geq 0$

Coming back to our optimization problem

$$w^* = \arg\min_w \sum_{i=1}^n \log(1+\exp(-z_i))$$

Here the minimum value of $\sum_{i=1}^n \log(1+\exp(-z_i))$ is 0. This minimum value is obtained only if all the values in the sequence become 0.

The optimization problem is

$$w^* = \arg\min_w \sum_{i=1}^n \log(1+\exp(-z_i))$$

From the graph of $\exp(-z_i)$, we see

As ' z_i ' increases, $\exp(-z_i)$ decreases.

As ' z_i ' $\rightarrow \infty$, $\exp(-z_i) \rightarrow 0$.

If ' z_i ' = +ve and ' z_i ' $\rightarrow \infty$, then $\exp(-z_i) \rightarrow 0$.

So gradually the term $\log(1+\exp(-z_i)) \rightarrow 0$

So the minimum value of $\sum_{i=1}^n \log(1+\exp(-z_i))$ is '0' when ' $z_i \rightarrow \infty$ for all 'i'.

But we have assumed as $z_i = y_i^*(w^T x_i)$, $D = \{(x_i, y_i), i \rightarrow 1 \text{ to } n\}$

So here ' x_i ' and ' y_i ' values are constant as they are the given data points.

So the only varying term is 'w'.

We have to modify 'w' in such a way that each ' $z_i \rightarrow +\infty$

$z_i = y_i^*(w^T x_i)$ is positive, only if the given data point (x_i, y_i) is classified correctly by the plane correctly.

$z_i = y_i^*(w^T x_i)$ is negative, only if the given data point (x_i, y_i) is misclassified.

If ' $z_i \rightarrow +\infty$ ', then we reach our minima.

So the best optimal w^* can be obtained only if

- All the training data points are classified correctly
- ' $z_i \rightarrow +\infty$

But here is a problem when all the training data points are classified correctly. If all the training data points are classified correctly, then if there are any outliers in the training data, then model fits to the training data in such a way that even the outliers also are classified correctly, thereby

leading the model to overfit. In overfitting, classification might be perfect on the training data, but not on the test data.

But if ' $z_i \rightarrow \infty$ ', then ' w ' also has to be either $-\infty$ or $+\infty$, depending on the values of ' x_i ' and ' y_i '. So according to the optimization problem, as we know ' w ' is a vector, each component(w_j) of the vector ' w ' has to be made either $-\infty$ or $+\infty$ depending on the values of ' x_i ' and ' y_i ' in order to arrive at a perfect classification of training data points.

So in order to get rid of the problem of overfitting, we apply a technique called **Regularization**.

L2 Regularization

The optimization problem becomes

$$\begin{aligned} w^* &= \arg\min_w \sum_{i=1}^n \log(1+\exp(-y_i(w^T x_i))) + \lambda w^T w \\ w^T w &= \sum_{j=1}^d w_j^2 = \|w\|_2^2 \end{aligned}$$

The optimization problem now becomes

$$w^* = \arg\min_w \sum_{i=1}^n \log(1+\exp(-y_i(w^T x_i))) + \lambda \sum_{j=1}^d w_j^2$$

The first term is the loss term and the second term is the regularization. Our job is to minimize the whole sum of both these terms. Here if $w_j \rightarrow +\infty$ (or) $w_j \rightarrow -\infty$, then the whole sum value becomes infinity as $w_j^2 \rightarrow \infty$. This is going to be quite opposite of what we actually need as we're actually trying to minimize the sum value.

The parameter ' λ ' prevents ' w_j ' from becoming $+\infty$ (or) $-\infty$. So the regularization term with the help of ' λ ' is avoiding $w_j \rightarrow +\infty$ and $w_j \rightarrow -\infty$.

If $w_j \rightarrow +\infty$ (or) $w_j \rightarrow -\infty$, then $\log(1+\exp(-z_i)) = 0$, but $\lambda \sum_{j=1}^d w_j^2$ becomes very large. So the regularization term and the loss term are moving in the opposite directions and are avoiding z_i 's from going to infinity. But for this optimization problem, they both reach an equilibrium stage where both the terms have minimum value and here they converge to an optimal value w^* .

The optimization problem is

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n \log(1+\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i))) + \lambda \sum_{j=1}^d w_j^2$$

Here ' λ ' → Hyperparameter. Its optimal value is obtained through cross validation.

If $\lambda=0$, then the model overfits. The influence of the regularization term on the model is negligible.

If ' λ ' is large(say ∞), then the influence of the loss term is negligible when compared to the regularization term, and doesn't show any impact on the model, leading the model to underfitting.

33.6 - L1 Regularization and Sparsity

The optimization problem we have been solving so far is

$$w^* = \arg\min_w \sum_{i=1}^n \log(1+\exp(-y_i^*(w^T x_i))) + \lambda \|w\|_2^2$$

We are using L2 Regularizer to ensure that ' z_i ' don't tend to ' $+\infty$ '. Otherwise ' w_i ' tend to be either $+\infty$ or $-\infty$. Also in order to prevent the model from overfitting, we use L2 regularizer.

Here arises a question of why to use L2 regularizer and are there any alternatives for L2 regularization.

There is a popular alternative called L1 Regularization. In L1 regularization, instead of using $\|w\|_2^2$ for regularization, we use $\|w\|_1$.

So now the optimization problem becomes

$$w^* = \arg\min_w \sum_{i=1}^n \log(1+\exp(-y_i^*(w^T x_i))) + \lambda \|w\|_1$$

As we are using absolute value of ' w_i ' in $\|w\|_1$, though ' w_i ' is '-ve' or '+ve', after converting into absolute format it becomes '+ve'.

Even L1 regularizer avoids $w_i \rightarrow \infty$ and $w_i \rightarrow -\infty$. L1 regularizer functions same as L2 regularizer. But it has got one advantage over L2 regularizer and it is the **Sparsity**.

Sparsity

Let us assume the optimal vector ' w ' = $\langle w_1, w_2, w_3, \dots, w_d \rangle$.

A solution for Logistic Regression is said to be sparse if many w_i 's are zero.

If we use L1 regularizer in Logistic Regression, all the unimportant (or) less important features in ' w ' become zero.

If we have the features $f_1, f_2, f_3, \dots, f_d$, then the corresponding components in ' w ' are $w_1, w_2, w_3, \dots, w_d$. If a feature ' f_j ' is least important, then

- If we apply L1 regularization, ' w_j ' becomes zero.
- If we apply L2 regularization, ' w_j ' becomes a small value, but not necessarily zero.

If we want all the unimportant features to become zero in our problem, we have to apply L1 regularization.

Elastic Net

Elastic Net regularization is a combination of both L1 and L2 regularization.

The optimization problem we have been working so far will become as below if we apply Elastic Net regularization.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^n \log(1+\exp(-y_i^*(\mathbf{w}^T \mathbf{x}_i))) + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2$$

(loss term + L1 regularization + L2 regularization)

Here we have 2 hyperparameters ' λ_1 ' and ' λ_2 '. These two hyperparameters can be found using cross-validation.

Elastic-Net results in sparse solutions as it contains 'L1' regularization while avoiding some of the disadvantages of 'L1' regularization.

On the other hand, 'L2' regularization tends to typically result in better performance, but no sparsity. Now the elastic net tries to combine the advantages of both these methods while minimizing their disadvantages.

It is preferred to use ElasticNet, but it takes more time to train with ElasticNet regularization as compared to 'L1' and 'L2'.

If $\lambda_1=0$ → ElasticNet reduces to L2 regularization.

If $\lambda_2=0$ → ElasticNet reduces to L1 regularization.

Note

L1 regularization can be used in any ML model to introduce sparsity. We use L1 regularization to zero out the weights of the least important features.

Both L1 and L2 regularizations are used to avoid overfitting. But along with avoiding overfitting, if we have a requirement of low latency, we have to prefer L1 regularization.

33.7 - Probabilistic Interpretation: Gaussian Naive Bayes

In Naive Bayes,

- a) If features are real valued, then we can assume that the real valued features have Gaussian Distribution.
- b) If $y_i = +1$ or 0 , then we can think of our class label as a Bernoulli Random variable.

If these two conditions are satisfied, then we can easily derive the whole Logistic Regression.

In probabilistic interpretation, using Gaussian Naive Bayes,

Logistic Regression = Gaussian Naive Bayes + Bernoulli Distribution

Assumptions made in Probabilistic Interpretation of Logistic Regression

- a) The response variable ' Y ' is boolean, governed by a Bernoulli distribution, with parameter $\pi = P(Y=1)$
- b) All the features ' X_i ' ($i = 1$ to d) are continuous random variables.
- c) For each ' X_i ', $P(X_i|Y=y_k)$ is a Gaussian distribution of form $N(\mu_{ik}, \sigma_i)$.
- d) For all $i \neq j$, ' X_i ' and ' X_j ' are conditionally independent given ' Y '.

In the geometric interpretation of Logistic Regression, the optimization problem is

$$w^* = \arg\min_w \sum_{i=1}^n \log(1+\exp(-y_i * (w^T x_i) / ||w||)) + \text{Regularization}$$

Here $y_i \in \{-1, +1\}$

In the probabilistic interpretation of Logistic Regression, the optimization problem is

$$w^* = \arg\min_w \sum_{i=1}^n -y_i \log P_i - (1-y_i) \log(1-P_i) + \text{Regularization}$$

Case 1

When $y_i = +ve$, then

In geometric interpretation, $y_i = +1$

In probabilistic interpretation, $y_i = +1$

In geometric interpretation, if 'w' is not a unit vector, then the optimization problem becomes

$$w^* = \arg\min_w \sum_{i=1}^n \log(1+\exp(-(w^T x_i)/\|w\|))$$

In probabilistic interpretation, if 'w' is not a unit vector, then the optimization problem becomes

$$\begin{aligned} w^* &= \arg\min_w \sum_{i=1}^n -\log(1/(1+\exp(-(w^T x_i)/\|w\|))) \\ &= \arg\min_w \sum_{i=1}^n \log(1+\exp(-(w^T x_i)/\|w\|)) \end{aligned}$$

Case 2

When $y_i = -ve$, then

In geometric interpretation, $y_i = -1$

In probabilistic interpretation, $y_i = 0$

In geometric interpretation, if 'w' is not a unit vector, then the optimization problem becomes

$$w^* = \arg\min_w \sum_{i=1}^n \log(1+\exp((w^T x_i)/\|w\|)) \text{ (because } y_i = -1)$$

In probabilistic interpretation, if 'w' is not a unit vector, then the optimization problem becomes

$$\begin{aligned} w^* &= \arg\min_w \sum_{i=1}^n -\log(1-(1/(1+\exp((w^T x_i)/\|w\|)))) \\ &= \arg\min_w \sum_{i=1}^n -\log(\exp((w^T x_i)/\|w\|)/(1+\exp((w^T x_i)/\|w\|))) \\ &= \arg\min_w \sum_{i=1}^n \log(1+\exp((w^T x_i)/\|w\|)) \end{aligned}$$

Note

Once if we have the same optimization which can be arrived at using either the probabilistic approach or geometric approach, we can solve the optimization problem using stochastic gradient descent.

33.8 - Loss Minimization Interpretation

The optimization problem is

$$w^* = \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i^* (w^T x_i) / \|w\|))$$

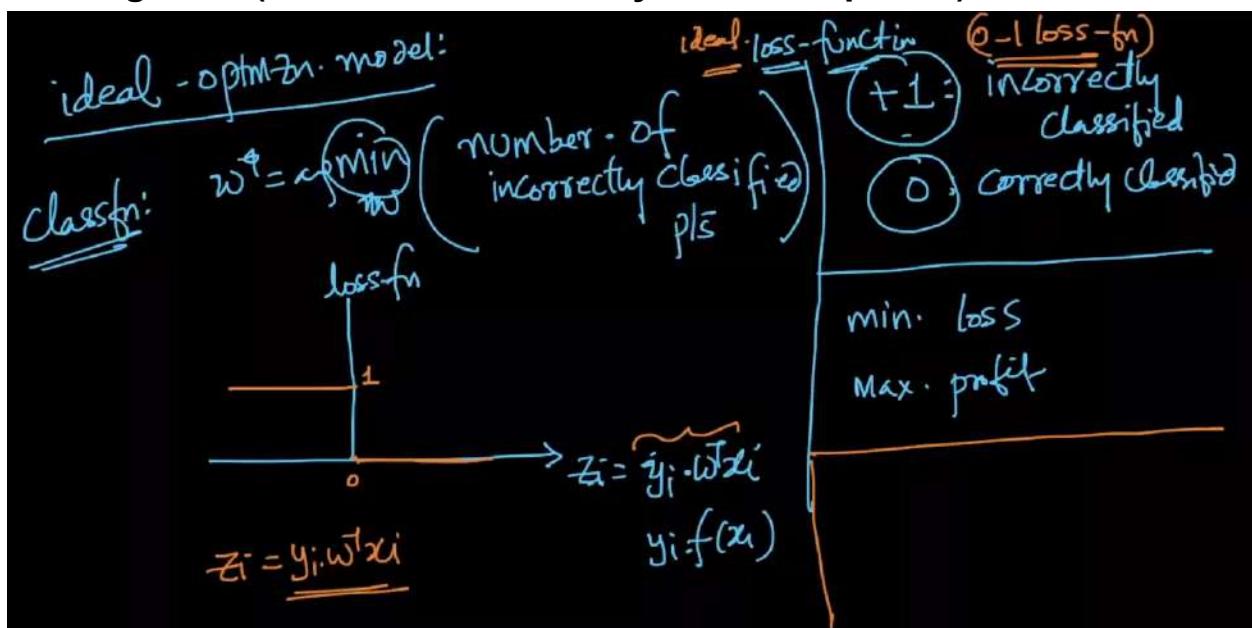
If 'w' is a unit vector, then $\|w\| = 1$.

$$z_i = y_i^* (w^T x_i) / \|w\| = y_i^* f(x_i) \text{ (where } f(x_i) = (w^T x_i) / \|w\|)$$

Let us go with building an ideal optimization model. Let us assume a function which returns '+1' if a data point passed through it is incorrectly classified and '0' if a data point passed through it is correctly classified. This function is called **Loss Function**.

So now the optimization problem would be

$$w^* = \arg\min_w (\text{Number of Incorrectly classified points})$$



As we know that

If ' z_i ' is +ve, then the data point (x_i, y_i) is classified correctly.

If ' z_i ' is -ve, then the data point (x_i, y_i) is classified incorrectly.

So we need a loss function such that

$$\begin{aligned} \text{Loss-function} &= 0, \text{ if } z_i > 0 \\ &= 1, \text{ if } z_i < 0 \end{aligned}$$

Such a loss function is called **0-1 loss function**.

$$\begin{aligned}
 \text{0-1 loss}(z_i) &= 0, \text{ if } z_i > 0 \\
 &= 1, \text{ if } z_i < 0
 \end{aligned}$$

From the loss minimization perspective, the ideal 'w' is obtained by

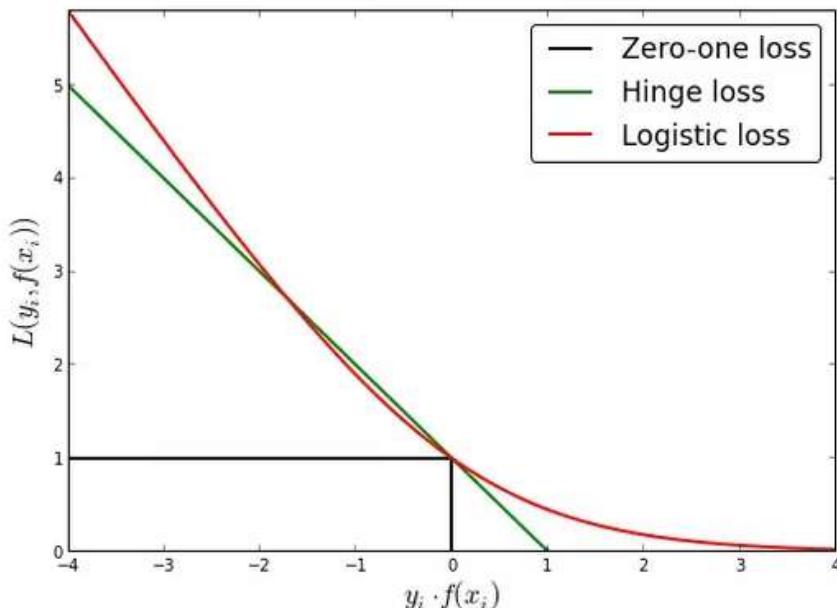
$$w^* = \arg\min_w \sum_{i=1}^n \text{0-1 loss}(z_i)$$

To solve optimization problems in ML, we'll use differentiation in calculus.

But a function is differentiable only if it is continuous. Similarly, a function is non-differentiable if it is not continuous. From this problem, we have the 0-1 loss function being defined in the intervals $(-\infty, 0)$ and $(0, \infty)$.

But it is not defined for $z_i = 0$. Hence we couldn't say it is differentiable, as it is not continuous at $z_i = 0$. Even though the 0-1 loss function is ideal, we could not solve the problem.

In optimization problems, if we are not able to solve them, we can approximate them. One such approximation for this problem is **Logistic Loss**.



From the above graph, we see that

If z_i is '+ve', then (0-1) loss = 0

As $z_i \rightarrow \infty$, Logistic Loss $\rightarrow 0$.

If z_i is '-ve', then (0-1) loss = 1

As $z_i \rightarrow -\infty$, Logistic Loss \rightarrow very large value.

In this problem, we are using Logistic Loss as an approximation over 0-1 loss. As we are using Logistic Loss as an approximation, we get the Logistic Regression model.

Similarly, if we use Hinge Loss as an approximation, we get the Support Vector Machine model.

33.9 - Hyperparameter Search: Grid Search and Random Search

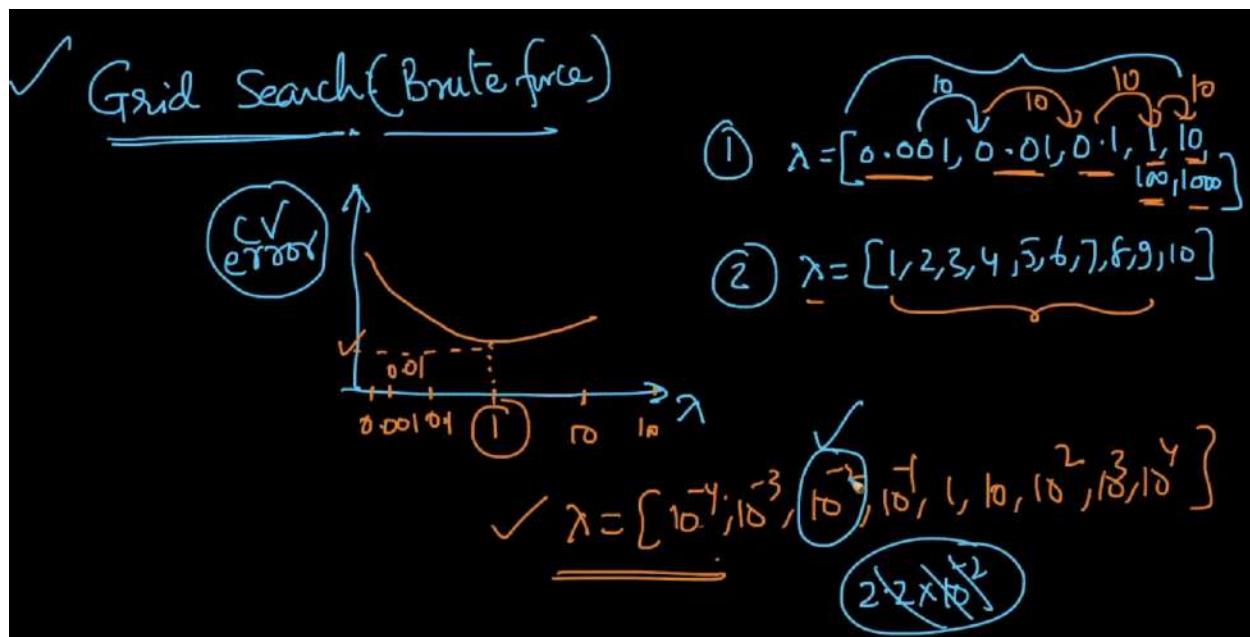
So far we have seen the hyperparameter ' λ ' in the formulation of Logistic Regression.

If $\lambda = 0 \rightarrow$ Overfitting

If $\lambda = \infty \rightarrow$ Underfitting

One job here is to find the optimal value of ' λ '.

Grid Search Cross-Validation

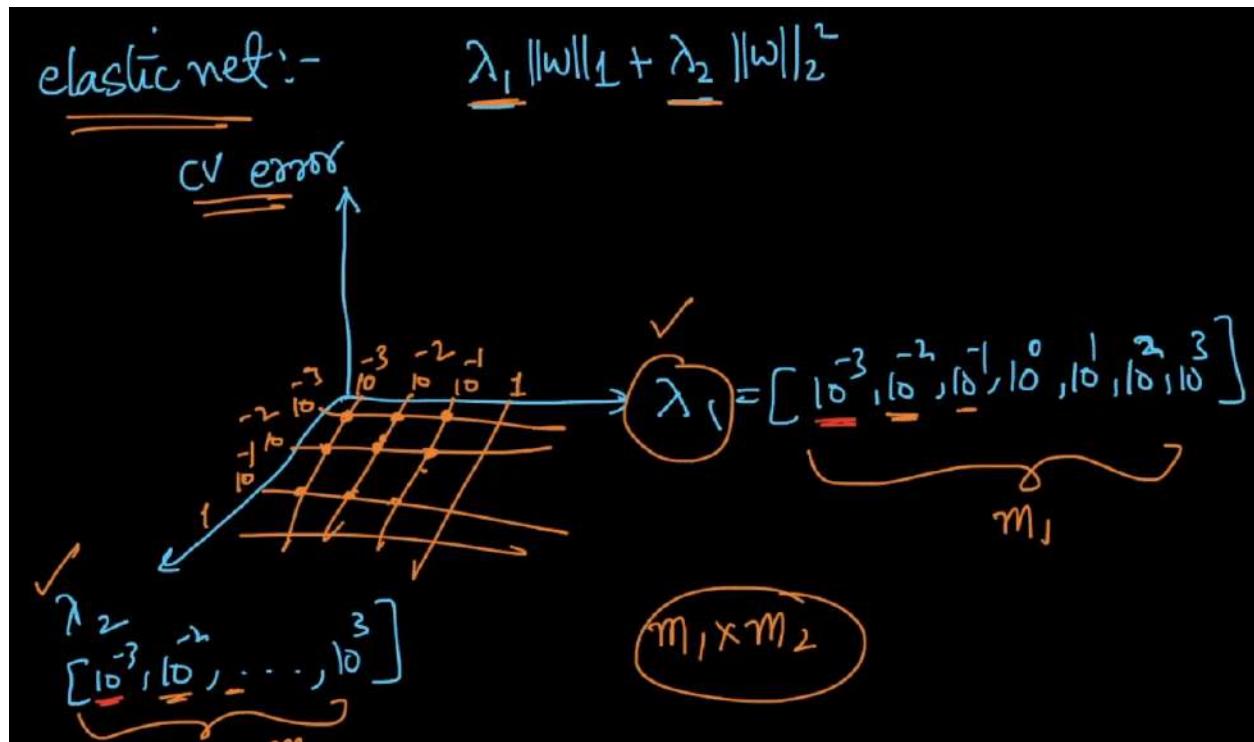


In Logistic Regression with either ' L_1 ' or ' L_2 ' regularization, ' λ ' is the only hyperparameter we have to tune. In K-NN, the hyperparameter 'K' takes only integers. Similarly in Logistic Regression, ' λ ' takes only the real values.

So we have to try different values of ' λ ' and compute the cross validation error for each value of ' λ ', and whichever value of ' λ ' yields the least cross validation error, that will be the optimal value of ' λ '.

Let us assume if we are using around 'm' different values for tuning of ' λ ', then we have to train the model 'm' times.

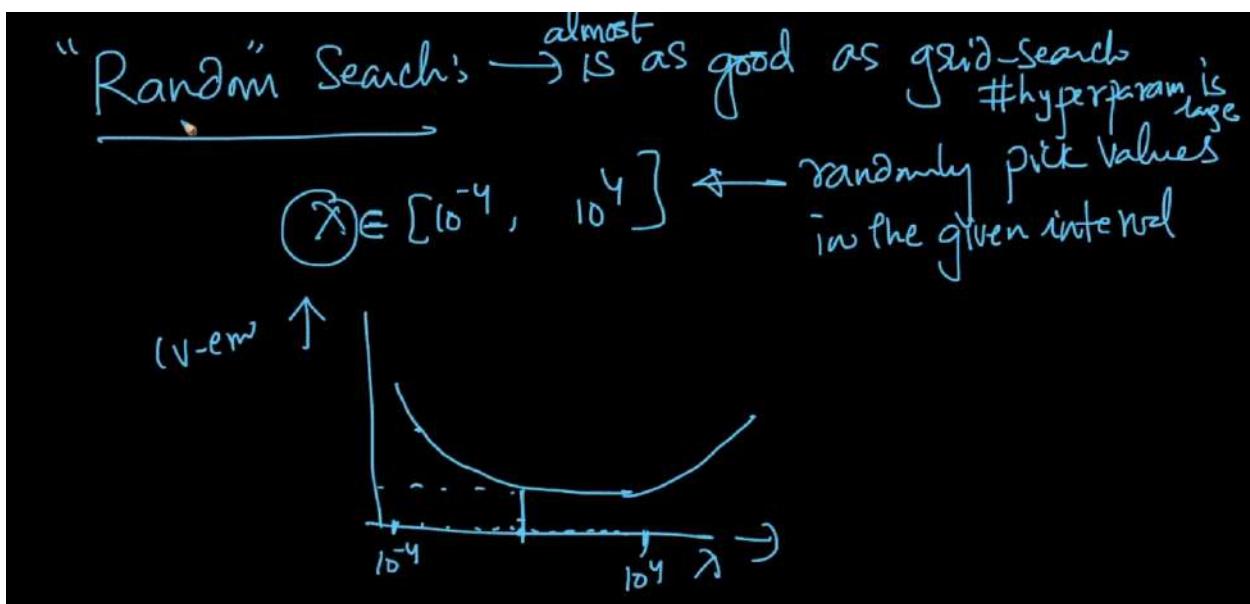
If we apply Elasticnet Regularization(ie.,both ' L_1 ' & ' L_2 ' regularization), then we have to find out the optimal values for the two hyperparameters ' λ_1 ' and ' λ_2 '.



Let the count of values used for tuning ' λ_1 ' be ' m_1 ', and the count of values used for tuning ' λ_2 ' be ' m_2 ', then the total number of combinations is $m_1 \times m_2$.

As the number of hyperparameters increase, the number of times the model needs to get trained increases exponentially. But in Logistic Regression, at the max, we could have 2 hyperparameters, whereas in neural networks, we have multiple hyperparameters to tune, and Grid Search Cross-Validation is not a good choice. In such a case, we have an alternative technique to go with and it is the **Random Search Cross Validation**.

Random Search Cross Validation



If we have a hyperparameter ' λ ' and if an interval is given for ' λ ', then this technique picks random values in the given interval for ' λ ' and plots the cross validation errors associated with those random values of ' λ '. Whichever value of ' λ ' gives the least cross validation error, that is chosen as the optimal value.

Random Search cross validation is as good as Grid Search cross validation, and is faster than Grid Search especially when the number of hyperparameters is large.

These two techniques can not only be used for Logistic Regression, but also can be used for any other ML algorithm like K-NN, Naive Bayes, etc.

Scikit-Learn Documentation Links

Grid Search CV :

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Random Search CV:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

33.10 - Column Standardization

Let us assume we have a dataset with 'd' features and 'n' data points. Here $x_i \in \mathbb{R}^d$

If we apply Feature/Column Standardization, then each value ' x_{ij} ' becomes

$$x_{ij}^l = (x_{ij} - \mu_j) / \sigma_j$$

where $i \rightarrow 1$ to 'n', $j \rightarrow 1$ to 'd'

Same as K-NN, even in Logistic Regression, it is important and mandatory to perform column/feature standardization. It is because in K-NN, we compute the distances between the data points. Similarly, in Logistic Regression, we compute the distances of the data points from the hyperplane separating different classes.

As the distances could easily get impacted by the features being present on different scales, the feature/column standardization is mandatory. Moreover standardization of features lead to faster convergence.

33.11 - Feature Importance and Model Interpretability

Let us assume we have features $f_1, f_2, f_3, \dots, f_d$ in our dataset. We shall compute the weight vector and let the components be $w_1, w_2, w_3, \dots, w_d$.

Let us make an assumption that all the features in our dataset are conditionally independent of each other(same like in Naive Bayes because Logistic Regression = Gaussian Naive Bayes + Bernoulli Distribution)

If we consider this assumption to be True, then we can make an interesting conclusion. That is the feature importance could be obtained using the values of w_j 's only if the assumption is True. If the assumption is False, then the conclusion also becomes False.

In K-NN, the feature importance is obtained using the feature selection techniques like Forward Feature Selection, Backward Feature Elimination, etc.

In Naive Bayes, the feature importance is obtained using the values $P(w_i|y=1)$.

In Logistic Regression, the feature importance could be obtained by w_j 's.

How to get the feature importance using ' w_j '?

For a feature ' f_j ', its corresponding weight component is ' w_j '.

$|w_j| \rightarrow$ Absolute value of weight corresponding to ' f_j '.

If $|w_j|$ is large(whether it is +ve or -ve), its contribution to $(w^T x)$ is large.

Case 1

If ' w_j ' is +ve and large, it shows a huge positive impact on $\sum_{i=1}^d w_j \cdot x_{qj} = w^T x_q$, and the value of $P(y_q=+1)$ increases.

Case 2

If ' w_j ' is -ve and large, it shows a huge negative impact on $\sum_{i=1}^d w_j \cdot x_{qj} = w^T x_q$, and the value of $P(y_q=-1)$ increases.

So we can determine the important features in Logistic Regression model, on the basis of the $|w_j|$ values.

Example

Let us assume we have to predict the gender of a given person, and out of all the features we have in our dataset, ‘hair-length’ is one among them. Let us denote the coefficient of ‘Hair-Length’ as ‘ w_{HL} ’. Let us denote ‘Male’ gender as class label ‘1’, and the female gender as the class label ‘-1’.

If $|w_{HL}|$ is large, as we know from common sense point that women tend to have longer hair than men. So here as women have longer hair, ‘ f_{HL} ’ tend to be more and more negative(it means large value, but negative), and the class label for this data point will be predicted as ‘-1’, as the value of $P(y_q = -1)$ increases with large negative value of ‘ w_{HL} ’.

Model Interpretability

In case of prediction, it would be sufficient if we just predict the value of ‘ y_q ’ either as -1 or +1, for a given data point ‘ x_q ’.

But in the case of interpretability, we also should be able to give the reasons for the predictions.

It is hard to interpret the feature importance geometrically, but is easier and more intuitive to understand it from an algebraic perspective.

33.12 - Collinearity of Features

So far we have seen that $|w_j|$ is used in determining the feature importance only if all the features are independent of each other.

If we have two features ' f_i ' and ' f_j ', and if one feature can be expressed as a linear function of the other feature (say $f_i = \alpha f_j + \beta$), then we can say that both ' f_i ' and ' f_j ' are collinear.

If we have the features $f_1, f_2, f_3, \dots, f_d$ and if they all are linearly related such as $f_1 = \alpha_1 + \alpha_2 f_2 + \alpha_3 f_3 + \dots + \alpha_d f_d$, then we can say that $f_1, f_2, f_3, \dots, f_d$ are multicollinear.

Why is $|w_j|$ not useful in determining feature importance if features are collinear?

Let us assume we have a dataset $D = \{x_i, y_i\}_{i=1}^n$

Let the optimal $w^* = <1,2,3>$

The corresponding $x_q = <x_{q1}, x_{q2}, x_{q3}>$

So if we compute ' $w^{*T}x_q$ ',

$$w^{*T}x_q = x_{q1} + 2x_{q2} + 3x_{q3} \quad (1)$$

Let's say $f_2 = 1.5f_1$, then

$$x_{q2} = 1.5x_{q1} \quad (2)$$

We can say ' f_1 ' and ' f_2 ' are collinear. If we substitute (2) in (1), we get

$$w^{*T}x_q = x_{q1} + 2(1.5x_{q1}) + 3x_{q3} = x_{q1} + 3x_{q1} + 3x_{q3} = 4x_{q1} + 3x_{q3} \quad (3)$$

So now the component coefficients have become $<4,0,3>$.

But the original vector coefficients of $w^{*T}x_q$ in (1) are $<1,2,3>$, but due to collinearity, we got the coefficients as $<4,0,3>$ in (3).

Here these two are entire different weight vectors, but still they represent the same classifier. Also these two vectors ultimately give the same value of ' $w^{*T}x_q$ ' for a data point ' x_q ', because the two features ' f_1 ' and ' f_2 ' are collinear.

Let us take those two vectors again.

$$w^* = <1,2,3> \quad (f_1 = 1, f_2 = 2, f_3 = 3)$$

$$\hat{w} = <4,0,3> \quad (f_1 = 4, f_2 = 0, f_3 = 3)$$

From ' w^* ', we come to a conclusion that ' f_3 ' is the most important feature, whereas from ' \hat{w} ', we come to a conclusion that ' f_1 ' is the most important feature, and ' f_2 ' is useless.

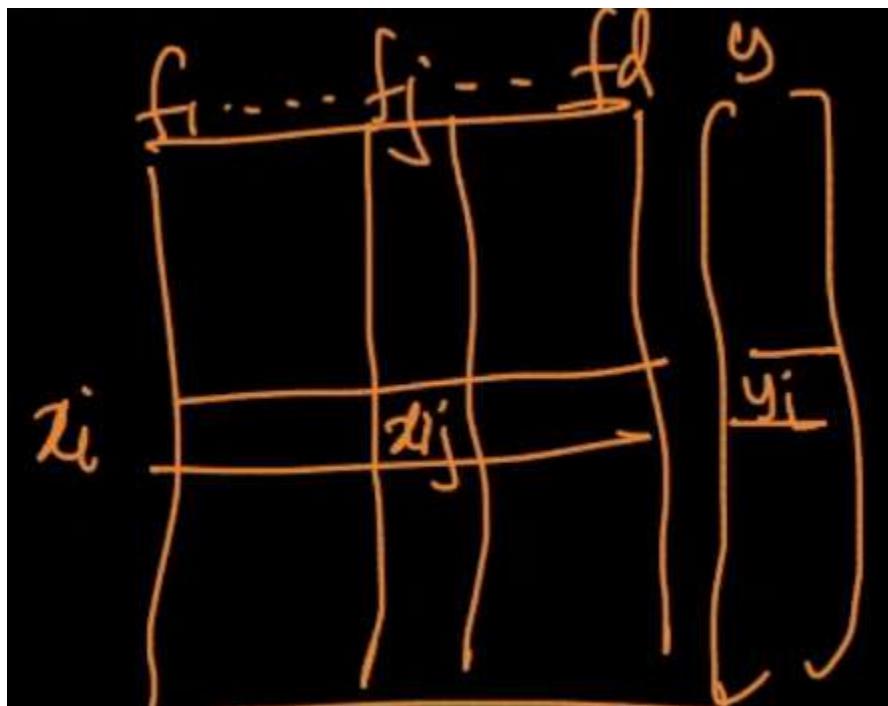
The conclusion drawn from both these vectors are quite opposite. There are more chances of misinterpretations because of collinearity among the features.

So if the features are collinear, the weight vector can change arbitrarily. Hence $|w_j|$ cannot be used in determining feature importance when any of the given features are collinear.

Before we use $|w_j|$ for determining the feature importance, we have to determine if there exists multi-collinearity between the features. The best technique to determine if the features are multi-collinear is **Perturbation** technique.

Perturbation

Let us assume we have a dataset 'X' of 'n' data points and each data point has 'd' dimensions.



We have apply Standardization/Normlization (as per our problem requirement) on the features.

We have to train the Logistic Regression model on the data matrix and obtain the weight vector ‘w-bar’.

Now we have to take each value ‘ x_{ij} ’ and perturbate them. Here perturbation in the sense, adding a small noise. Let the noise be denoted as ‘ ϵ ’. So train the logistic regression model using the perturbation data, and obtain the weight vector ‘w-tilda’.

$$w\text{-bar} = \langle w\text{-bar}_1, w\text{-bar}_2, w\text{-bar}_3, \dots, w\text{-bar}_d \rangle$$

$$w\text{-tilda} = \langle w\text{-tilda}_1, w\text{-tilda}_2, w\text{-tilda}_3, \dots, w\text{-tilda}_d \rangle$$

We now have to check how the components of these two vectors differ. If the differences between the components of ‘w-bar’ and ‘w-tilda’ are huge, then we can say that our features are collinear. Then we can’t use $|w_j|$ to determine the feature importance.

If the differences between the components of ‘w-bar’ and ‘w-tilda’ are small, then we can neglect the collinearity issue.

If we have the problem of collinearity, then as we could not use $|w_j|$ to determine feature importance, we have to go for generic techniques like Forward Feature Selection to determine the feature importance.

But before using $|w_j|$ for determining feature importance, we always have to perform collinearity check.

33.13 - Train and Runtime Space and Train Complexities

The problem of optimization that has to be solved in Logistic Regression is

$$w^* = \arg\min_w (\text{Logistic Loss} + \text{Regularization})$$

This problem is going to be solved using Stochastic Gradient Descent algorithm.

For this problem, let 'D' be the given training dataset.

$n \rightarrow$ number of points in the training data

$d \rightarrow$ number of dimensions

For Training Phase,

$$\text{Time Complexity} = O(nd)$$

For Runtime,

Space Complexity = $O(d)$ (It is because, we compute the vector ' w^* ' and store each component of ' w ' which is d -dimensional)

Time Complexity = $O(d)$ (It is because, we compute $\sum_{j=1}^d w_j x_{qj}$ which has ' d ' terms in the sequence)

Note

- If ' d ' is small, then Logistic Regression is good for low latency applications.
- If ' d ' is large, then we have to apply L1 regularization that introduces sparsity (as it makes w_j 's corresponding to least important features equal to 0).
- In case, if we still have more number of computations that could not be performed by the system, then even after having sparsity, we have to keep increasing the value of ' λ ', which increases the sparsity and makes more number of weight components equal to 0 and thereby reducing the number of computations.
- But increasing/decreasing ' λ ' by a large amount leads to underfitting/overfitting. Hence we need to modify ' λ ' in such a way that it keeps a balance between Bias, Variance and Latency time.

33.14 - Real-World Cases

1) Decision Surface

We have a linear decision surface in Logistic Regression. In 2-D space, it is called a line, in 3-D it is called a plane and in n-D space it is called a hyperplane.

2) Assumption

Logistic Regression works on the assumption that the given data is perfectly/almost linearly separable.

3) Feature Importance

The feature importance in Logistic Regression is obtained by the values of $|w_j|$, if the features are not collinear. If collinearity exists among the features, then we have to choose techniques like Forward Feature Selection, Backward Feature Elimination, etc.

4) Imbalanced Data

We can balance the dataset either using upsampling (or) downsampling (or) on the basis of class weights.

5) Impact of Outliers

The impact of the outliers is very low as we have the sigmoid function which reduces the magnitude of signed distance. The impact of the outliers is very low, but not completely vanished.

Procedure for outlier removal that can be followed

- a) For every point ' x_i ' in D_{Train} , compute the distance of it to the hyperplane. (ie., you have to compute $w^T x_i$)
- b) Remove those points from D_{Train} , which are far away from the hyperplane, and keep the remaining points. Let the dataset with the remaining points be D_{Train}^1
- c) Compute the optimal 'w' for D_{Train}^1 as the final solution.

6) Missing Values

We can use the standard imputation techniques(that were discussed in the course) to handle the missing values in the data.

7) Multi-class problems

We can go with the one-vs-rest approach in case of a multi-class classification problem. There are also extensions of Logistic Regression to multi-class classification. They are Maximum Entropy model, Softmax classifier and Multinomial Logistic Regression.

8) Similarity/Distance Matrix

There is an extension of Logistic Regression called Kernel Logistic Regression. It takes similarity matrices as input, whereas the normal Logistic Regression, doesn't accept the similarity matrices as input.

9) Best and Worst Cases of Logistic Regression

Best Case

- a) If our data is almost/perfectly linearly separable.
- b) If we have low latency requirement. (ie., especially when L1 regularization is applied)
- c) It is very fast to train.

Worst Case

When our data is not linearly separable.

10) High Dimensionality

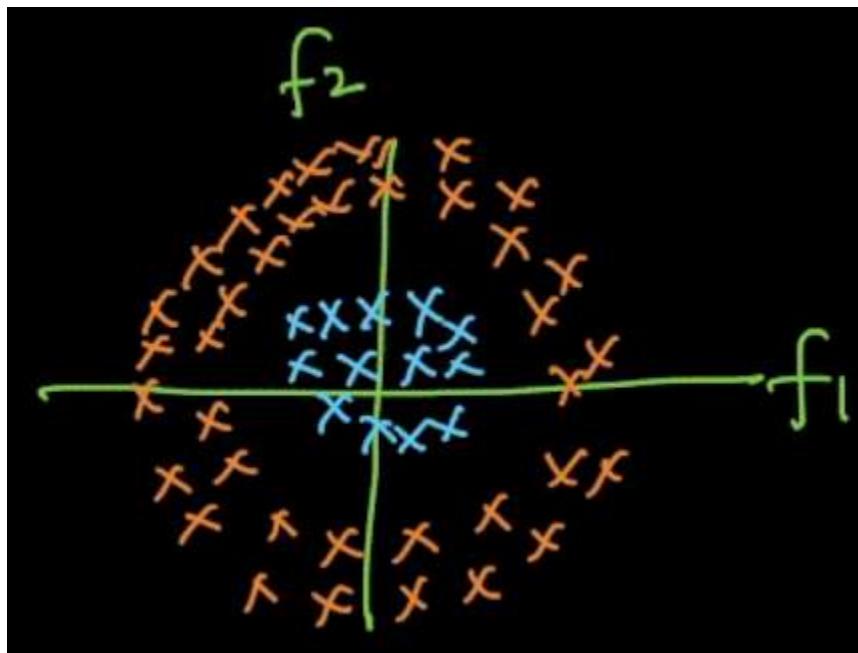
- a) When the dimensionality is high, the chances for our data to be linearly separable are also high.
- b) Logistic Regression works well when the dimensionality is large.
- c) If we want a low latency system in presence of high dimensionality, then we have to apply L1 regularization.

33.15 Non Linearly Separable Data and Feature Engineering

Logistic Regression in general works on the assumption that the given data is linearly separable.

Example 1

Let us assume that have the given data as shown below.

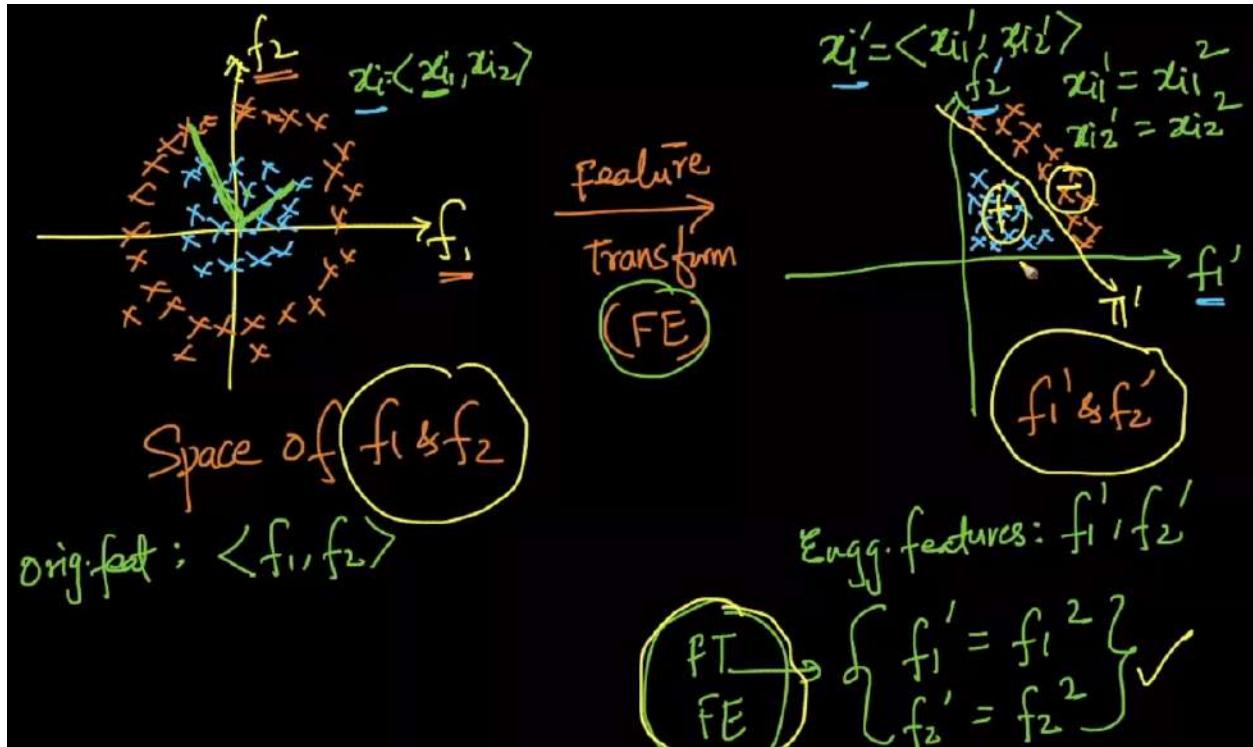


Here the data isn't linearly separable. In such cases, we cannot draw a line/plane/hyperplane in the space of ' f_1 ' and ' f_2 ' to separate the positive and the negative points, as it doesn't work.

Moreover, at the same time, we also couldn't conclude that Logistic Regression is not going to work in this context. We can apply Logistic Regression by following some ways.

In order to make the data suitable for applying Logistic Regression, we have to perform Feature Engineering (ie., applying some transformations to the existing features).

Let us have a look at the below geometric representation of the same data.



Through Feature Engineering we are converting the features ' f_1 ' and ' f_2 ' as $f_1' = f_1^2$ and $f_2' = f_2^2$. After feature engineering, the data has become linearly separable. So we can now apply Logistic Regression.

How do we know which transform to apply?

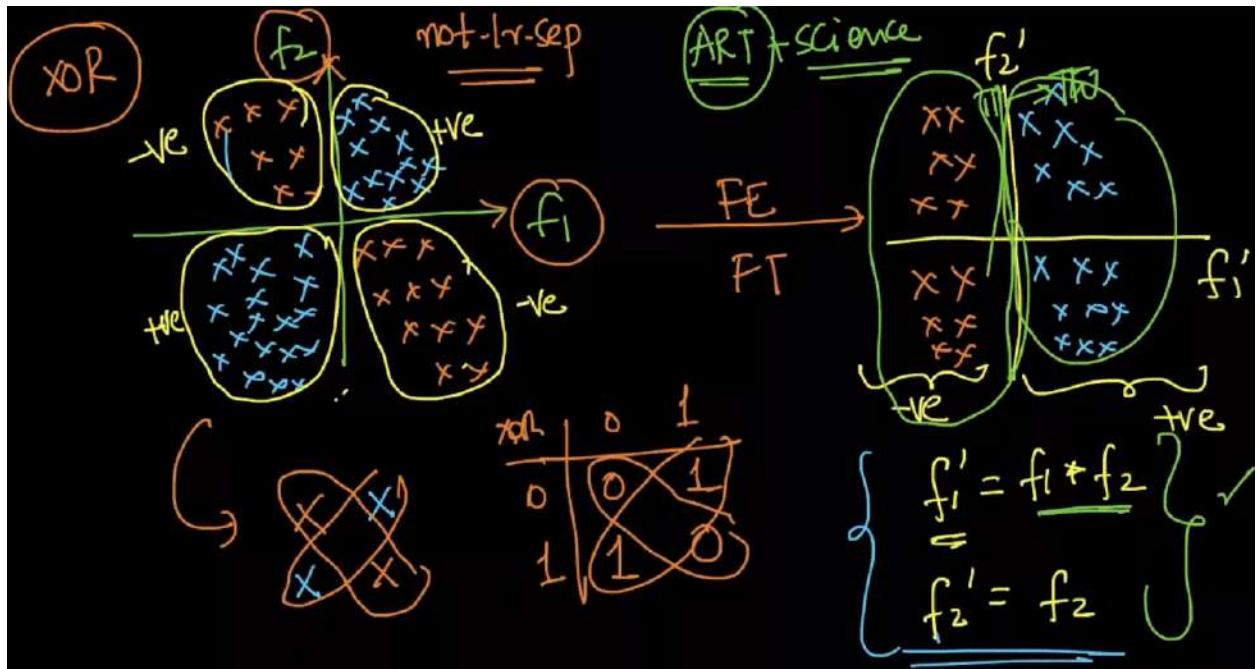
We see that the given data is in the form of a circle. As we have ' f_1 ' and ' f_2 ' as features, we should consider

$$f_1^2 + f_2^2 = r^2 \quad \dots \quad (1)$$

Here equation (1) is not linear in ' f_1 ' and ' f_2 ' but is linear in ' f_1^2 ' and ' f_2^2 '. So if $f_1' = f_1^2$ and $f_2' = f_2^2$, then we can say equation (1) is linear in ' f_1' and ' f_2' .

Example 2

Let us look at another dataset whose points are present in the 2D space as shown below



Here left side figure shows us the actual position of the points in the 2D space. We need to apply feature transforms such that we could separate the classes using a line.

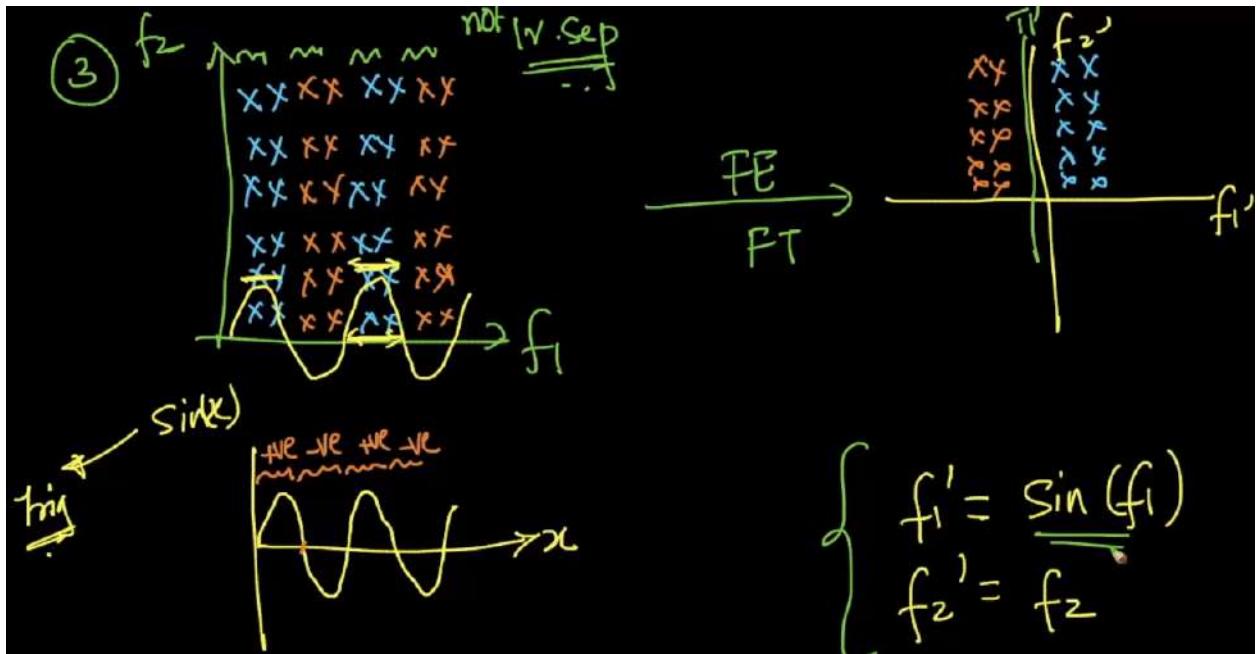
When we look at the points in the left side figure, we can see the figure is in XOR form. So the appropriate transformation that we can apply to make the data linearly separable as shown in the right side figure is

$$f_1' = f_1 * f_2$$

$$f_2' = f_2$$

Example 3

Let us look at another example in the below figure.



Here the input looks in the form of a sinusoidal wave. So the appropriate transformation that has to be applied in order to transform the data into a linearly separable format is

$$f_1' = \sin(f_1)$$

$$f_2' = f_2$$

Typical Transforms for Real-Valued Functions

1) Polynomial Feature Transforms

Examples: $f_1 * f_2$, f_1^2 , f_2^2 , f_1^3 , f_1^4 , ..., $f_1^2 f_2$, ...

2) Trigonometric Feature Transforms

Examples: $\sin(f_1)$, $\cos(f_1)$, $\sin(f_1) * \cos(f_2)$, $(\sin(f_1))^2$, ...

3) Boolean Feature Transforms

Examples: OR, AND, XOR

4) Mathematical Feature Transforms

Examples: $\log(f_1)$, $\exp(f_1)$, ...

Note: Regarding the video lecture 33.16, we aren't giving any notes, as it is purely a discussion on the code samples. We are providing you with the link to download the ipython notebook, and if you still have any queries, please feel free to post them in the comments section.

Link: <https://drive.google.com/file/d/1WcVTkIMZBMu9VTCIWeupOK0r2aYbHk8p/view>

33.17 - Extensions to Logistic Regression: Generalized Linear Models (GLM)

There is an extension to Logistic Regression called Generalized Linear Model (GLM).

In probabilistic interpretation,

Logistic Regression = Gaussian Naive Bayes + Bernoulli Distribution

- a) In Logistic Regression, if we change the distribution of random variable from Bernoulli to Multinomial, then the resulting Logistic Regression model is called **Multinomial Logistic Regression**. This is useful when we have a multi-class classification problem to solve. It is also known as the soft-max classifier.
- b) In Logistic regression, if we assume that $P(y|X) \sim N(\mu, \sigma^2)$, then we call the resulting algorithm a **Linear Regression**, and it is a regression technique. (Here $y_i \in R$)
- c) If the output variables y_i 's are poison distributed, then we get Poisson Regression. Poisson Regression is used to measure the counts.

Reference Link

Generalized Linear Models:

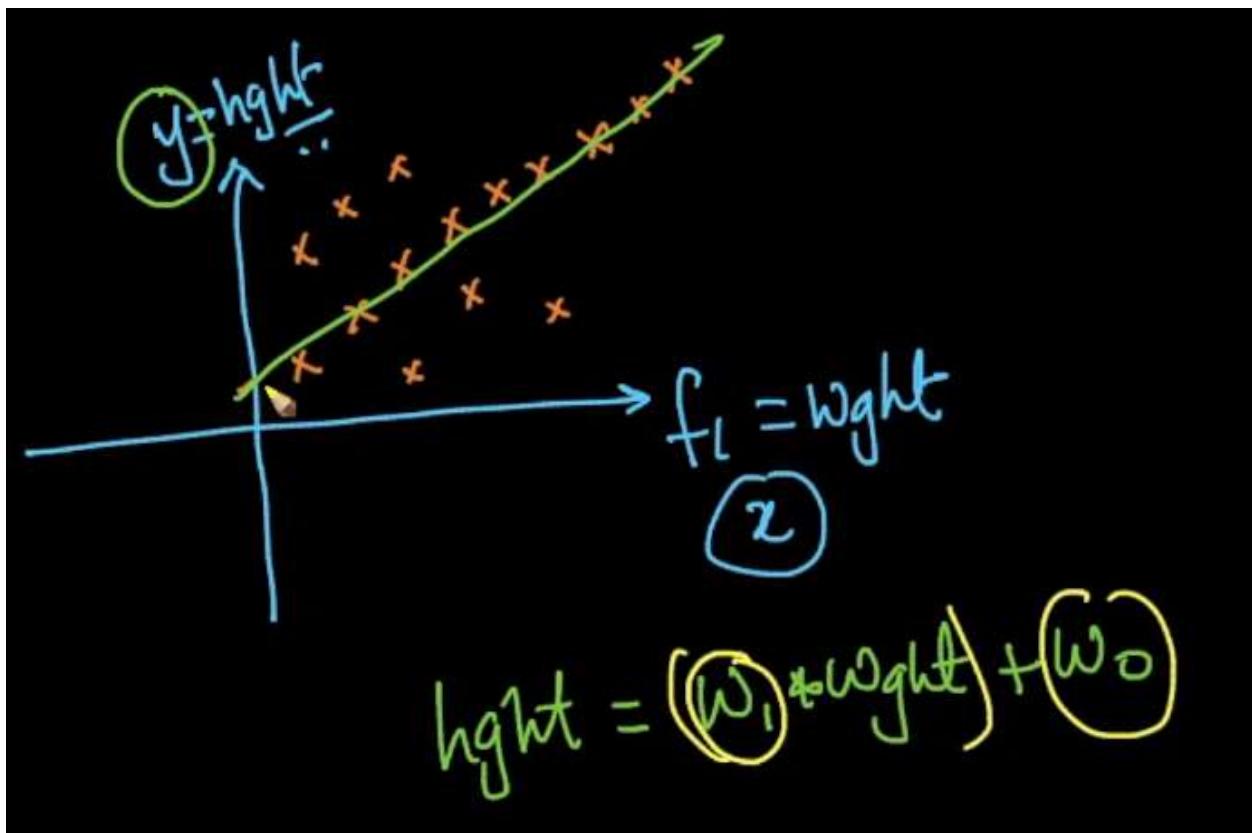
<http://cs229.stanford.edu/notes2020spring/cs229-notes1.pdf> (Part 3)

34.1 Geometric Intuition of Linear Regression

Linear Regression is an actual regression technique. Given the dataset, $D = \{x_i, y_i\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$.

Geometry behind Linear Regression

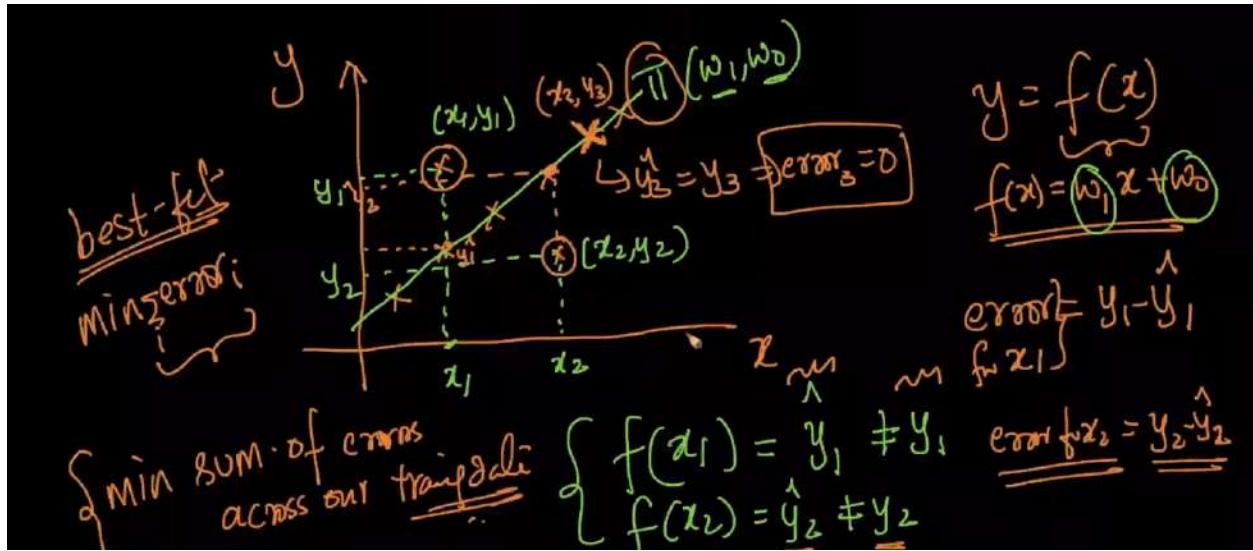
Let us consider the problem of predicting the height given weight, gender, ethnicity, hair color, etc.



The main objective in linear regression is to find a line/plane that fits the given data. In 2-D space, $\text{height} = (w_1 * \text{weight}) + w_0$.

If we want to take another feature 'hair-color' also into consideration, then the formulation becomes $\text{height} = (w_1 * \text{weight}) + (w_2 * \text{hair-color}) + w_0$.

Linear Regression works on the assumption that most of the points lie on a linear surface. We have to find a line/plane that best fits the data points.



If any point (x_i, y_i) doesn't lie on the plane $\pi(w_1, w_0)$, then there occurs an error with that point. (ie., $y_i - y_i^*$).

In the figure, we see the points (x_1, y_1) and (x_2, y_2) are not present on the plane. So

For $(x_1, y_1) \rightarrow f(x_1) = y_1^*$ (where $y_1^* \neq y_1$)

For $(x_2, y_2) \rightarrow f(x_2) = y_2^*$ (where $y_2^* \neq y_2$)

Error for $x_1 \rightarrow e_1 = y_1 - y_1^*$

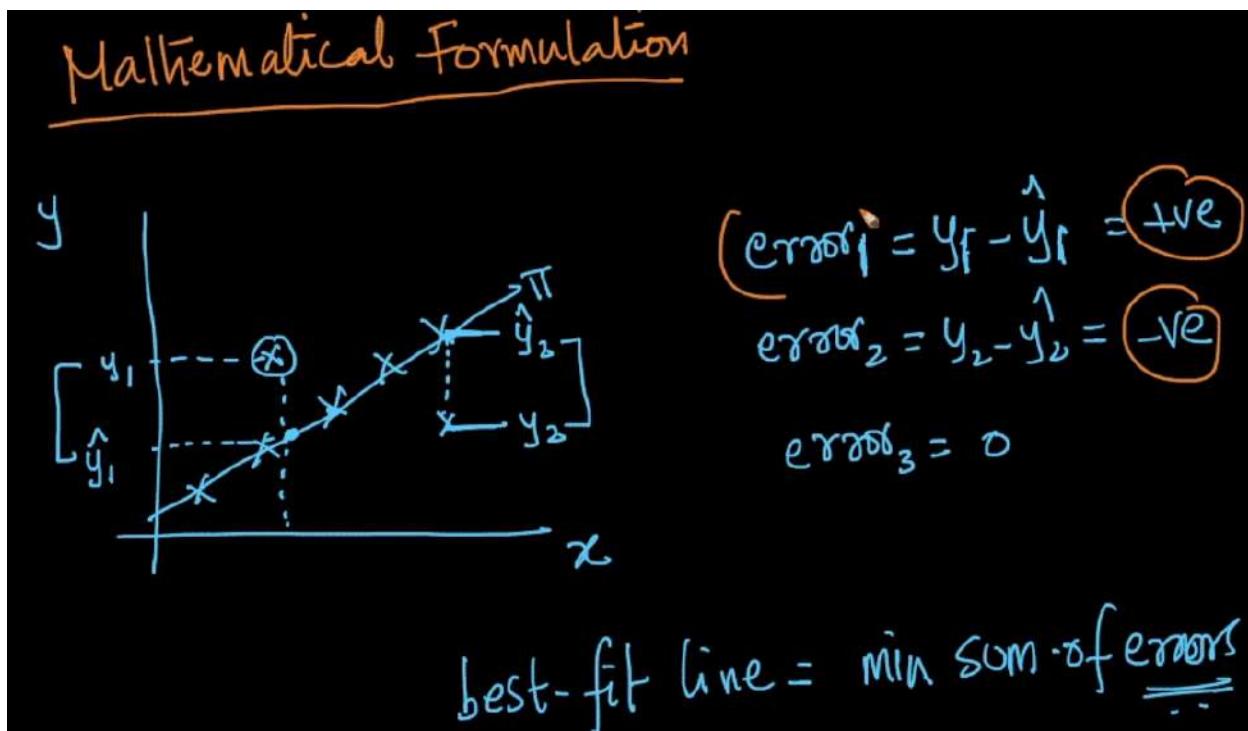
Error for $x_2 \rightarrow e_2 = y_2 - y_2^*$

We have to best fit the plane. It means we have to minimize the sum of errors for each point across the training data. So the term that has to be minimized is $\sum_{i=1}^n (y_i - y_i^*)$.

Difference between Logistic Regression and Linear Regression

- While both Linear Regression and Logistic Regression are linear models, their objective functions are slightly different.
- Logistic Regression minimizes the Log-Loss whereas Linear Regression minimizes the Squared loss.
- In Logistic Regression, we are trying to find a plane that separates the data points belonging to different classes, whereas in Linear Regression, we find a plane that passes through most of the data points.

34.2 Mathematical Formulation



For a point (x_i, y_i) , $\text{error}(e_i) = y_i - \hat{y}_i$

In the above figure, we can see

For the point $(x_1, y_1) \rightarrow \text{error}(e_1) = +ve$

For the point $(x_2, y_2) \rightarrow \text{error}(e_2) = -ve$

If we have positive and negative values for the errors, they might get cancelled out and may not give the proper interpretation. Hence we take squares of the errors.

So the mathematical formulation for Linear Regression is we have to find the optimal (w, w_0) that maximizes the sum of squares of errors.

$$(w^*, w_0^*) = \arg\min_{w, w_0} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$\text{Where } \hat{y}_i = f(x_i) = w^T x_i + x_0$$

So now the optimization problem becomes

$$w^*, w_0^* = \arg\min_{w, w_0} \sum_{i=1}^n [y_i - (w^T x_i + x_0)]^2$$

Linear Regression is also called **Ordinary Least Squares** (or) **Linear Least Squares**.

Regularization

The optimization problem with regularization becomes

$$(\mathbf{w}^*, \mathbf{w}_0^*) = \arg\min_{\mathbf{w}, w_0} \sum_{i=1}^n [y_i - (\mathbf{w}^T \mathbf{x}_i + w_0)]^2 + \lambda \|\mathbf{w}\|_2^2$$

Same like in Logistic Regression, we can apply L1 regularization (or) L2 regularization (or) ElasticNet regularization.

Note

In the probabilistic interpretation of Linear Regression,
If $P(y_i|x_i) = N(\mu, \sigma^2)$, then using the basic tools of probability, we can easily derive the Linear Regression.

All the three interpretations of Linear Regression(ie., Geometric, Probabilistic, and Loss Minimization) leads us to the same point.

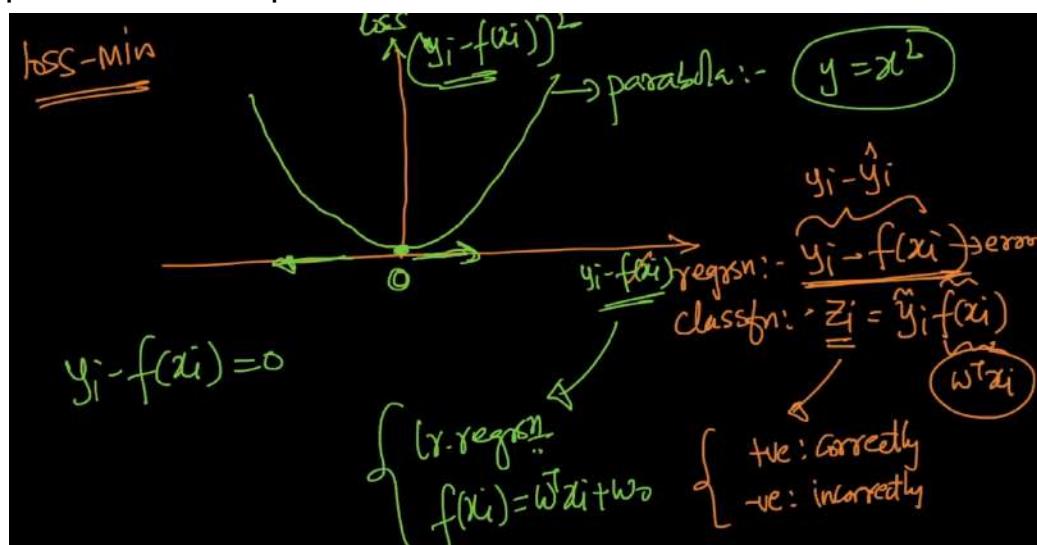
Loss Minimization Interpretation

In classification using Logistic Regression, we have seen

If $z_i = y_i^*(\mathbf{w}^T \mathbf{x}_i) > 0$, then ' \mathbf{x}_i ' is correctly classified.

If $z_i = y_i^*(\mathbf{w}^T \mathbf{x}_i) < 0$, then ' \mathbf{x}_i ' is misclassified.

In the case of Linear Regression, we see a plot with the error on the 'X' axis and loss on the 'Y' axis. Here we do not have right or wrong, as we have for classification in Logistic Regression. We only have the error. The plot looks like a parabola as shown below.



Using the loss minimization approach for regression, if we assume square loss, what we get is the Linear Regression.

The loss is penalized equally for both positive and negative points. The loss is always positive for both positive and negative errors.

The derivative of a squared function is easier when compared to the absolute value. Hence we take the square of the errors in the optimization problem.

Need for Regularization

Video Lecture Link: <https://www.youtube.com/watch?v=iUm2Z1SKuGk>

In Logistic Regression, we have seen that if $w \rightarrow \infty$, then the whole optimization fails. Hence we have introduced regularization to overcome this problem.

But in Linear Regression, the optimization problem is given as

$$\mathbf{w}^* = \arg\min_w \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda \|w\|_2^2$$

(We also can use L1 regularization in place of L2)

Let us consider an example. The relation we have in Linear Regression is

$$y_i = w^T x_i$$

Let us assume we have 3 features in our problem.

$$x_i \in \mathbb{R}^3, w \in \mathbb{R}^3, y_i \in \mathbb{R}$$

So now the relationship becomes

$$y_i = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3}$$

Let us consider now a simulated dataset 'D' such that

$$y_i = 2x_{i1} + 3x_{i2} + 0x_{i3}$$

This is the underlying function and the output for every data point is computed using this function. If this dataset is given, and we are asked to fit a Linear Regression model, the ideal solution would be

$$w = [2, 3, 0]$$

But if we want to simulate the real-world datasets, then there will be minor errors associated with each feature. So for the real world simulated datasets, the relationship becomes

$$y_i = 2(x_{i1} + \epsilon_1) + 3(x_{i2} + \epsilon_2) + 0(x_{i3} + \epsilon_3)$$

The original data points in the training dataset are slightly corrupted by the noise. So now let this new simulated dataset be denoted as 'D'.

If we are not informed about the underlying function, then

Case 1: Applying Linear Regression without Regularization

Here as our dataset was corrupted with noise, and as we aren't applying the regularization, we'll have only the loss term in the optimization. As there is no regularization, the model tries to fit the training data as best as possible. So instead of [2,3,0], we get the weight vector as [2.1, 3.4, 0.2].

Case 2: Applying Linear Regression with Regularization

Here we get the weight vector somewhere like [2.01, 3.003, 0.003], but not perfectly as [2,3,0]. Here the regularization is trying to control the vector 'w' from reaching ' ∞ ', and the whole optimization problem is to reduce the squared error.

The reason why we do not get perfect values for the weight vector 'w' is, the main aim of 'w' here is to minimize the squared loss and the regularizer forces the vector 'w' to be as small as possible. The moment we introduce the regularizer, it pulls down the vector 'w' just to create a balance, as the squared loss tried to decrease, the regularizer tries to increase the loss (ie., the domination of loss is more than 'w').

Also in case 1, when there is no control on 'w', it tries to overfit the data. Hence in order to put some control on 'w', regularization is used.

These types of weights that are close to the underlying weights tend to generalize well on the unseen data. This is the reason why regularization occupies the first place.

34.3 Real-world cases

1) Imbalanced Data

As this is a regression problem, we do not have the problem of the dataset being imbalanced, as all the output values are the real values.

2) Feature Importance and Model Interpretability

If the features are not collinear, we can easily use $|w_i|$ to find out the feature importance.

3) Multi-class

As the output values are the real numbers, there are no chances for the existence of multiple classes in the dataset.

4) Feature Engineering/Feature Transform

Whatever Feature engineering techniques are applicable in Logistic Regression, the same can be applied here as well.

5) Outliers

In Logistic Regression, the sigmoid function has reduced the impact of the outliers on the model. But in Linear Regression, the procedure to remove the outliers is

- a) Find w^* , w_0^*
- b) Find the points very far away from the hyperplane ' π ' based on the error values(ie., $y_i - \hat{y}_i$).
- c) Remove these points as the outliers.

(ie., $D_{Train}^l = D_{Train} - \text{Outliers}$). Go to step 'a' and repeat this process.

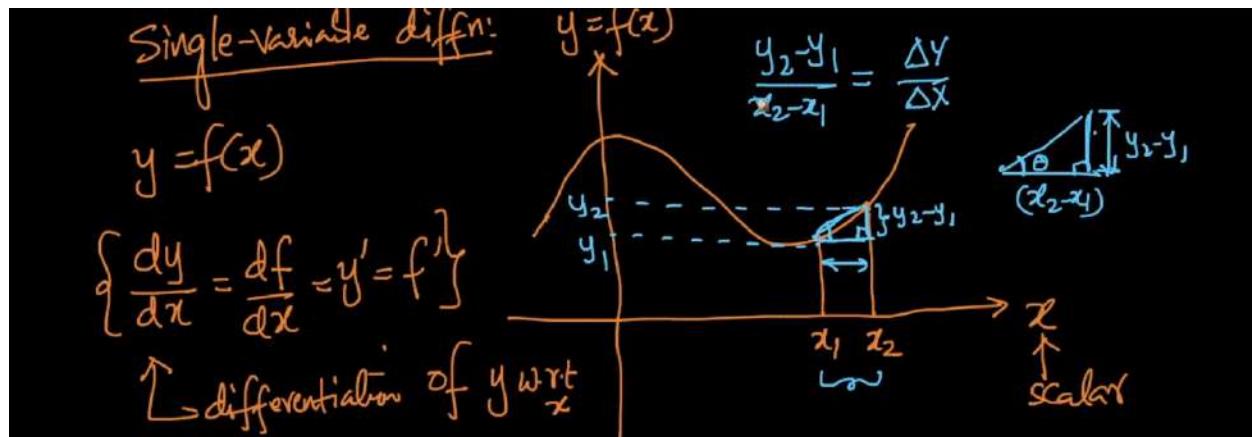
Note: Regarding the video lecture 34.4, we aren't giving any notes, as it is purely a discussion on the code samples. We are providing you with the link to download the ipython notebook, and if you still have any queries, please feel free to post them in the comments section.

Link:<https://drive.google.com/file/d/16yIAGm-A61vEgcnmNY5rWCWFCDQ6Y-k/view>

35.1 - Differentiation

Single Variable Differentiation

Let us consider a function $y = f(x)$ as shown in the figure below.



The derivative of this function is denoted as dy/dx (or) df/dx (or) y' (or) f' . It is pronounced as differentiation of 'y' with respect to 'x'.

$dy/dx \rightarrow$ rate of change of 'y' with respect to 'x'. (i.e., it indicates how much 'y' changes, as 'x' changes)

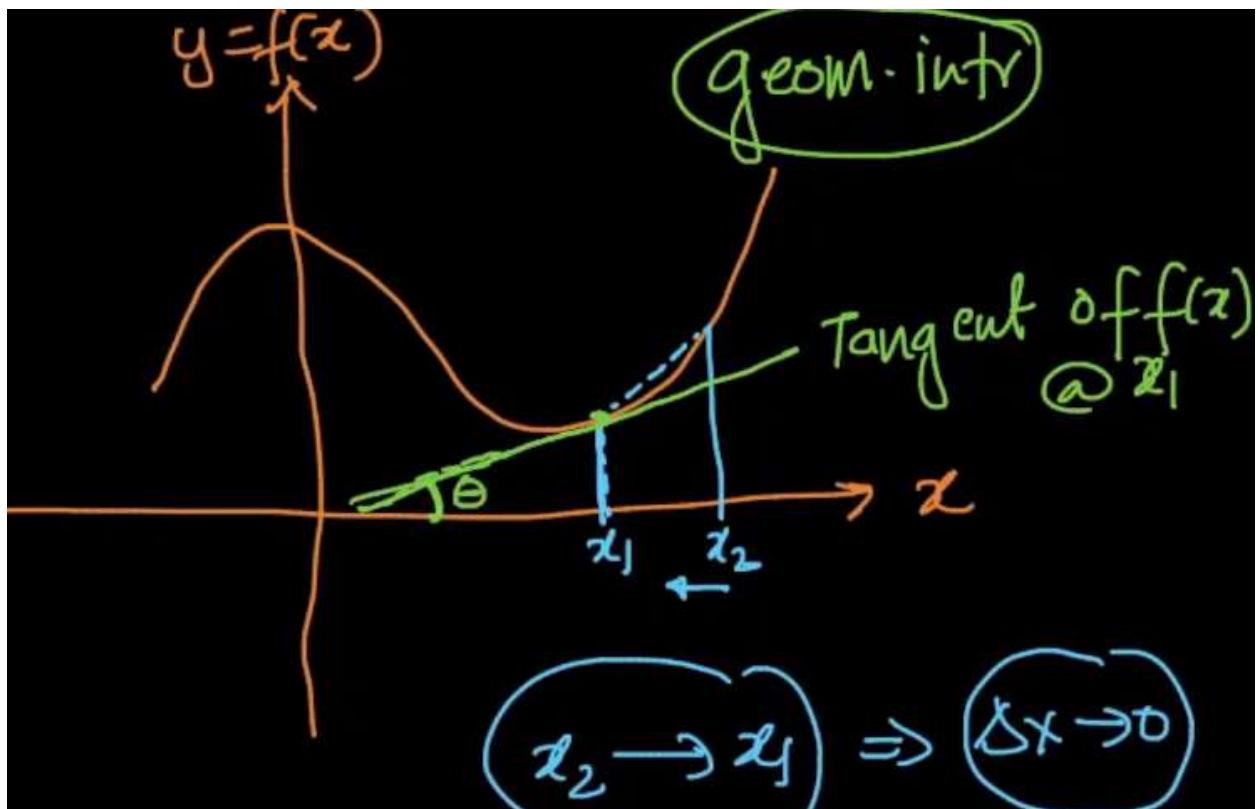
$$dy/dx = (y_2 - y_1)/(x_2 - x_1) = \Delta y / \Delta x = \tan \theta$$

$$dy/dx = \lim_{\Delta x \rightarrow 0} \Delta y / \Delta x \quad (\text{Here } \Delta x \rightarrow 0 \text{ means change in 'x' is very small})$$

Here 'θ' is the angle made by the tangent to the function, with the 'X' axis. A tangent is the hypotenuse that we obtain as $\Delta x \rightarrow 0$.

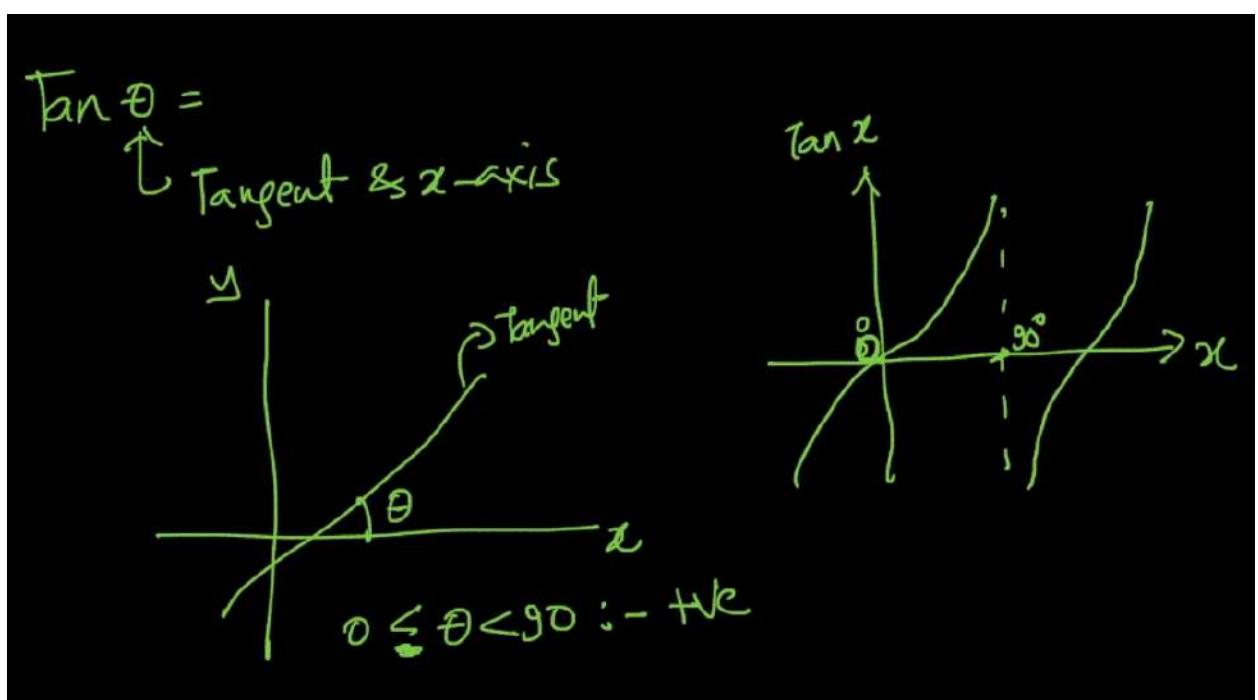
$dy/dx =$ slope of the tangent to $f(x)$

$[dy/dx]_{x=x_1} =$ slope of the tangent to $f(x)$ at $x=x_1$



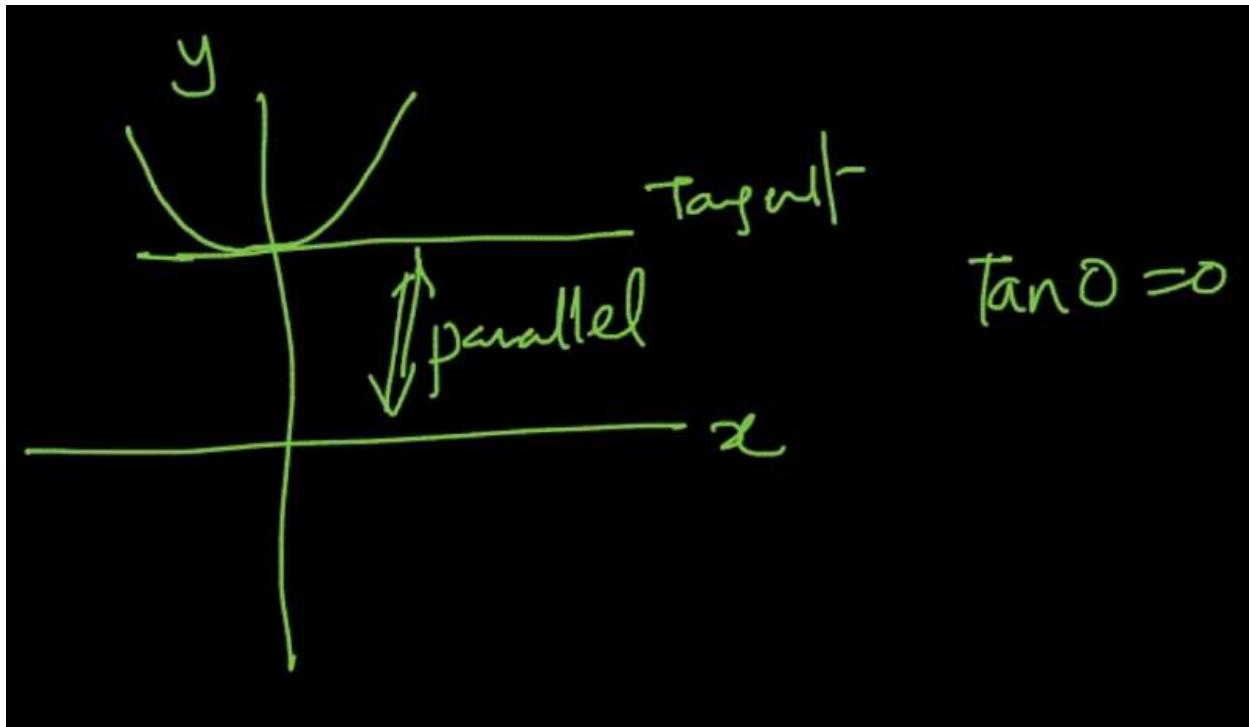
Slopes at different values of 'θ'

Case 1



If the value of ' θ ' lies in between 0 and 90° , then the value of $\tan\theta$ is positive.

Case 2

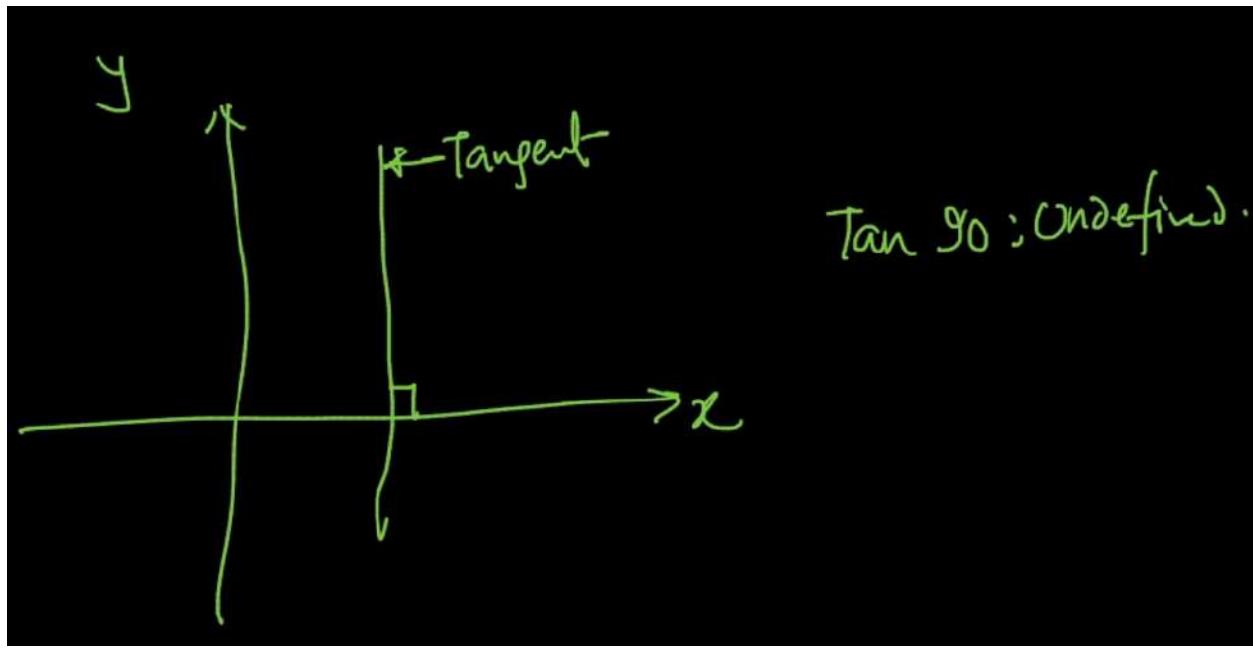


If the tangent is parallel to the 'X' axis, then the angle made by the tangent with the 'X' axis is 0. In such a case, the slope is always equal to 0.
Slope = $\tan\theta = \tan(0) = 0$.

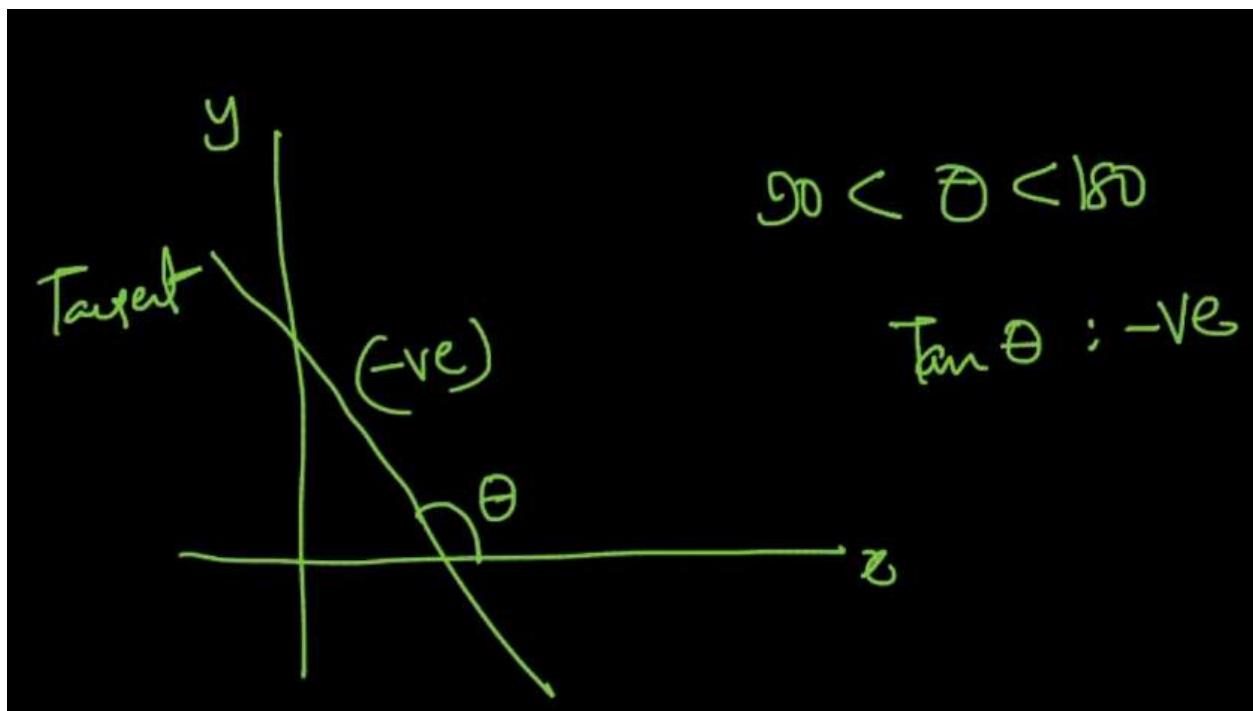
Case 3

If the tangent is parallel to the 'Y' axis, then the angle made by the tangent with the 'X' axis is 90° . In such a case, the slope is always undefined.

Slope = $\tan\theta = \tan(90) = \text{undefined}$.



Case 4



If the angle ' θ ' lies in between 90° and 180° , then the slope is always negative.

Slope = $\tan\theta$ is negative.

Basics of Differentiation

- 1) $d/dx(x^n) = n \cdot x^{n-1}$
- 2) $d/dx(c) = 0$
- 3) $d/dx(cx^n) = c \cdot n \cdot x^{n-1}$
- 4) $d/dx(\log(x)) = 1/x$
- 5) $d/dx(e^x) = e^x$
- 6) $d/dx(f(x)+g(x)) = d/dx(f(x)) + d/dx(g(x))$
- 7) $d/dx(f(g(x))) = df/dg \cdot dg/dx$

Example

$$f(g(x)) = (a-bx)^2$$

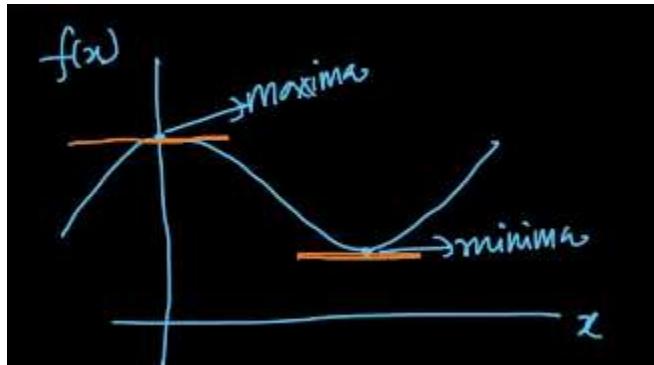
$$\text{Here } g(x) = a-bx, f(x) = x^2$$

$$\begin{aligned}d/dx f(g(x)) &= df/dg \cdot dg/dx \\&= -2b(a-bx)\end{aligned}$$

35.3 - Maxima and Minima

The largest value a function/variable can give is called Maxima and the smallest value is called Minima.

Let us look at the function 'f' given below.



The maxima and minima values of $f(x)$ are clearly shown in the figure above.

Both at maxima and minima, the slope of the function = 0.

For example, let us consider a function $f(x) = x^2 - 3x + 2$

$$df/dx = 0 \Rightarrow \text{slope} = 0$$

As we are differentiating with respect to 'x', the slope is equal to 0, for both maxima and minima.

$$\text{So } df/dx = 2x - 3 = 0 \Rightarrow x = 3/2$$

Now we have to find whether $f(x)$ is having a minima or maxima at $x=3/2$.

The function is $f(x) = x^2 - 3x + 2$

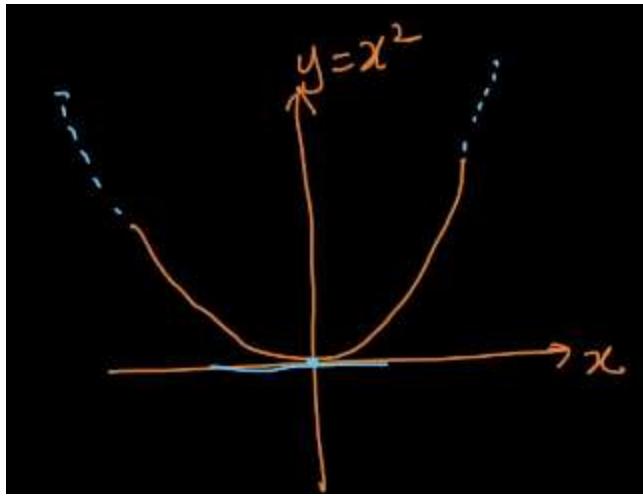
$$\text{Let us compute the value of } f(3/2) = -0.25$$

Let us now compute $f(1)$, we get $f(1) = 0$ (here just for the sake of an example, we are computing $f(1)$)

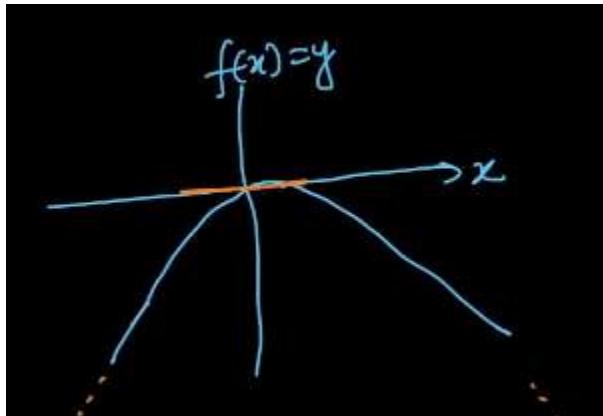
As $f(1.5) < f(1)$, $f(1.5)$ cannot be maxima.

So we can conclude that the minimum of $f(x)$ occurs at $x=3/2$.

Now let us consider the example of a parabola.

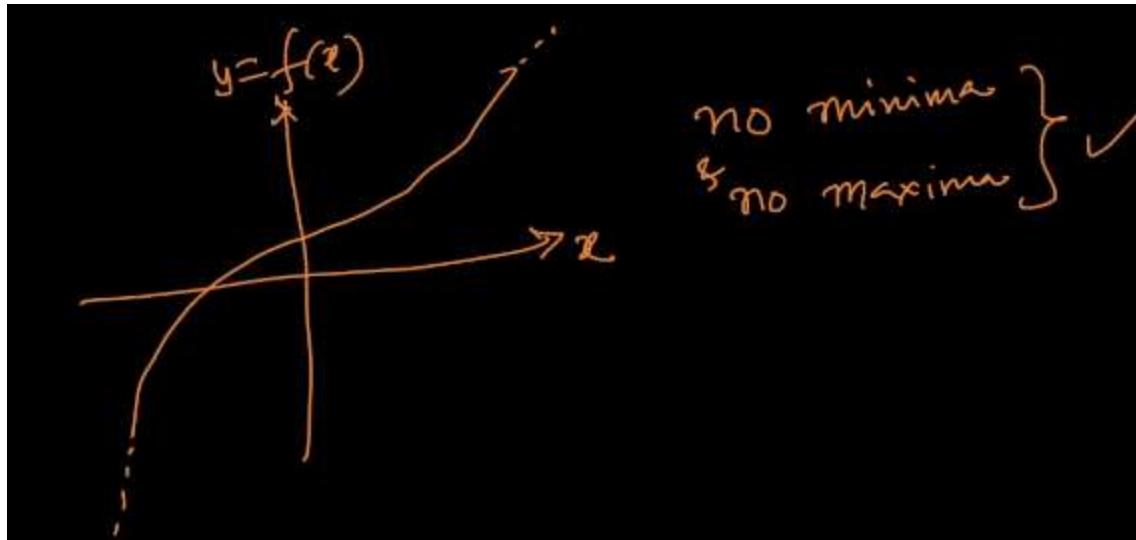


In this parabolic function, we can see the minimum value occurs at $x=0$, and there is no maximum value at all. Hence we can say the function $y = x^2$ has a minima, but not maxima.



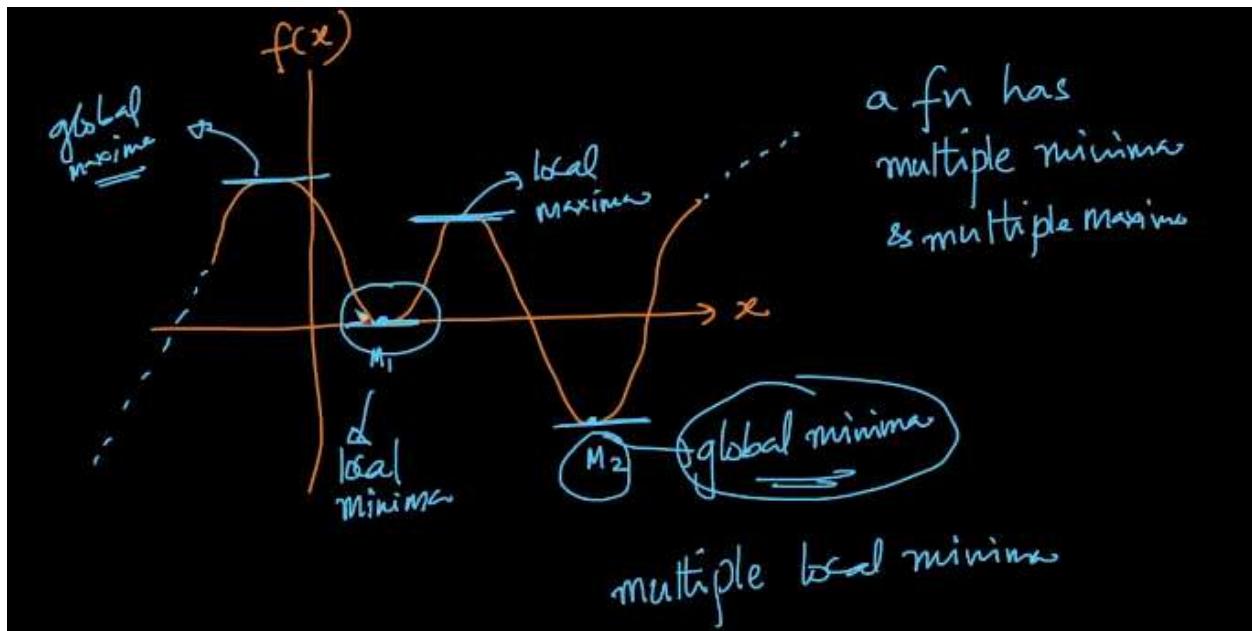
In this parabolic function, we can see the maximum value occurring at $x=0$, and there is no minimum value at all. Hence we can say the function $y=-x^2$ has a maxima, but not minima.

There are a few functions which neither have a maxima nor a minima. The best examples for such a category of functions are the monotonic functions.



In this example, we can see there is no maxima and no minima. Such functions do not converge at all in the optimization problems.

There is another category of functions which have multiple maxima and minima. The best examples for such a category are the trigonometric functions.



Here we can see multiple minima and maxima. A function can have multiple local minima/maxima, but can have only one global minima/maxima.

For simple functions with only one variable, we were able to find out the maxima/minima values using the differentiation approach.

In case, if we come across complex functions (say $\log(1+\exp(ax))$ which is of the form of logistic loss), then it is really difficult to find the maxima/minima using the single differentiation.

For such complex functions, we need to repeat the differentiation process for a certain number of times, in order to arrive at the maxima/minima. For such a process, we use an algorithm called **Gradient Descent**.

35.4 - Vector Calculus: Grad

So far we have worked with 'x' as a scalar, and $f(x)$ as a function of the scalar 'x'.

Let us now assume 'x' as a vector because in order to operate in high dimensional space, 'x' needs to be a vector.

Example:

$$\text{Let } y = \mathbf{a}^T \mathbf{x}$$

$$\mathbf{x} = \langle x_1, x_2, x_3, \dots, x_d \rangle$$

$$\mathbf{a} = \langle a_1, a_2, a_3, \dots, a_d \rangle$$

$$y = \sum_{i=1}^d a_i x_i$$

Here $\frac{\partial f}{\partial \mathbf{x}} = \nabla_{\mathbf{x}} f$ (This notation indicates that 'x' is a vector and 'f' is a function on vector 'x'. The ' ∇ ' symbol is pronounced as **Del** (or) **Grad.**)

$$\nabla_{\mathbf{x}} f = [\frac{\partial f}{\partial x_1}$$

$$\frac{\partial f}{\partial x_2}$$

$$\frac{\partial f}{\partial x_3}$$

.

.

$$\dots$$

$$\frac{\partial f}{\partial x_d}]$$

If we have a function $f(\mathbf{x})$ as

$$f(\mathbf{x}) = a_1 x_1 + a_2 x_2 + a_3 x_3 + \dots + a_d x_d \text{ then}$$

$$\nabla_{\mathbf{x}} f = [\frac{\partial f}{\partial x_1}$$

$$[a_1$$

$$\frac{\partial f}{\partial x_2}$$

$$a_2$$

$$\frac{\partial f}{\partial x_3} = a_3$$

.

.

.

.

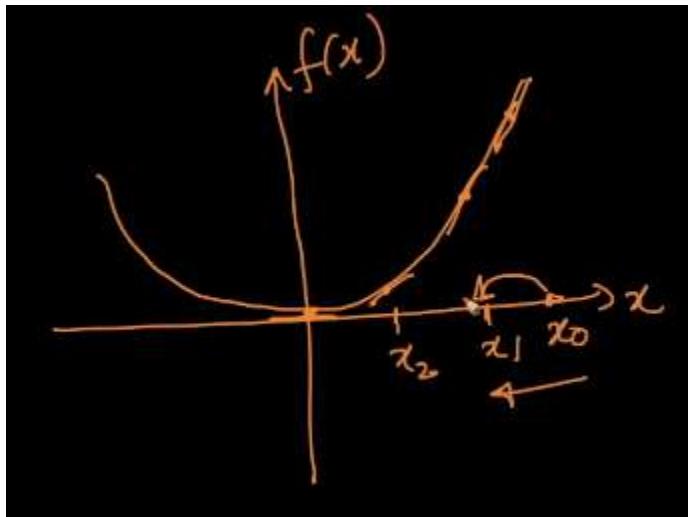
$$\frac{\partial f}{\partial x_d}]$$

$$a_d]$$

35.5 - Gradient Descent: Geometric Intuition

Gradient Descent is an iterative algorithm used to find maxima/minima points of a function which is highly complex.

In some problems like Logistic Regression optimization problem, in order to compute the maxima/minima, solving $df/dx = 0$ (using scalars) or $\nabla_x f = 0$ (using vectors) is not straight to find the optimal maxima/minima values. For solving such problems, we use Gradient Descent.



Pick some value ' x_0 ' as the first guess of x^* .

In iteration 1, we get ' x_1 '.

In iteration 2, we get ' x_2 '.

In iteration 3, we get ' x_3 '.

.

In iteration ' k ', we get ' x_k '.

The objective here is as the iteration number increases, the value of ' x ' becomes closer and closer to the optimal ' x^* '.

Let us consider the optimization problem

$$x^* = \arg\min_x f(x)$$

$$\text{Here } \min(f(x)) = \max(-f(x))$$

$$\max(f(x)) = \min(-f(x))$$

The slope changes its sign from +ve to -ve at minima. As we approach our minima, the slope decreases. As we move away from minima, the slope increases.

Procedure for Gradient Descent

- 1) Pick an initial point ' x_0 ' at random.
- 2) So now pick ' x_1 ' such that $x_1 = x_0 - r * [df/dx]_{x_0}$ (where $r \rightarrow$ learning rate (or) step size)
Let $r=1$, then $x_1 = x_0 - [df/dx]_{x_0}$
Initially as the slope is positive, $[df/dx]_{x_0} > 0$
So $x_1 = x_0 - (+\text{ve value}) \Rightarrow x_1 = x_0 - \text{slope} \Rightarrow x_1 < x_0$
Let us assume ' x_0 ' is on the other side. So now,
 $x_1 = x_0 - (-\text{ve value}) \Rightarrow x_1 = x_0 + \text{slope} \Rightarrow x_1 > x_0$

- 3) $x_2 = x_1 - r * [df/dx]_{x_1}$.

So at any iteration, $x_{i+1} = x_i - r * [df/dx]_{x_i}$

Similarly we have to compute $x_0, x_1, x_2, \dots, x_k, x_{k+1}$ where

$$x_k = x_{k-1} - r * [df/dx]_{x_{k-1}}$$

If $(x_k - x_{k-1})$ is very very small, then we have to terminate the loop and declare $x^* = x_k$.

So incase, if we are working on multi-dimensional data using vectors, then

$$x_k = x_{k-1} - r[\nabla_x f]$$

At every iteration, the slope decreases, as we move towards the minima.

i.e., $[df/dx]_{x_0} > [df/dx]_{x_1} > [df/dx]_{x_2} > \dots > [df/dx]_{x_k}$

35.6 - Learning Rate

In Gradient Descent, the update equation is given as

$$x_i = x_{i-1} - r[\frac{df}{dx}]_{x(i-1)}$$

Here 'r' → Learning rate (or) step size

So far we have assumed 'r' to be constant.

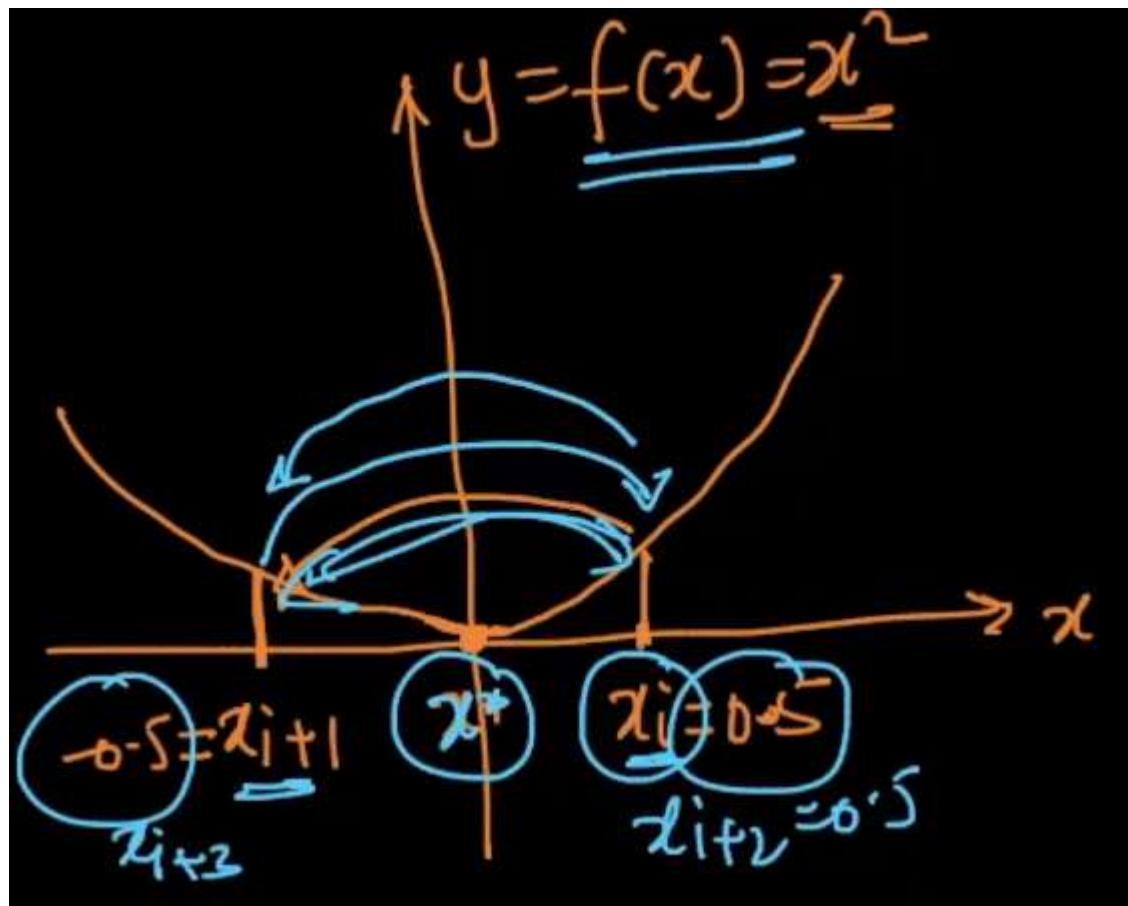
Let us assume a function $f(x) = x^2$. Let us assume after a few iterations, we reach at $x_i=0.5$.

$$x_{i+1} = x_i - r * [\frac{df}{dx}]_{x_i}$$

$$f(x) = x^2$$

$$\frac{df}{dx} = 2x$$

$$\text{Let } r = 1, \text{ then } x_{i+1} = 0.5 - 1 * [2 * 0.5] = -0.5$$



Here in the step of moving from $x_i = 0.5$ to $x_{i+1} = -0.5$, we are not moving towards minima, but have crossed the minima and have reached the other side. We simply jumped over x^* on the other side.

$$x_{i+2} = -0.5 - 1*(2*(-0.5)) = 0.5$$

In this way, we have alternate values on each side. We are oscillating between both sides. We can never converge to x^* . This is called an oscillation problem.

The remedy for the oscillation problem is to change 'r' with each iteration. When we reduce the learning rate 'r', the size of the oscillation(jump) decreases and there are more chances to converge.

So now 'r' becomes a function of the number of iterations.

$$r = f(\text{number of iterations})$$

So we can create some function $r = h(i)$, (where $i \rightarrow$ number of iterations) such that as 'i' increases, 'r' should reduce.

35.7 - Gradient Descent for Linear Regression

The mathematical formulation of Linear Regression with the intercept term (w_0) and without regularization is

$$L(w) = w^* = \arg\min_w \sum_{i=1}^n (y_i - w^T x_i)^2$$

Here $\langle x_i, y_i \rangle$ belongs to the dataset and are constants. The only thing that keeps changing in every iteration is 'w'.

So now we have to solve $\nabla_w L$.

$$\nabla_w L = \sum_{i=1}^n \{2 * (y_i - w^T x_i)(-x_i)\}$$

- 1) We have to pick a /vector $w_0 = \langle \dots \rangle$
- 2) $w_1 = w_0 - r * \sum_{i=1}^n \{2 * (y_i - w_0^T x_i)(-x_i)\}$
- 3) $w_2 = w_1 - r * \sum_{i=1}^n \{2 * (y_i - w_1^T x_i)(-x_i)\}$
- 4) $w_3 = w_2 - r * \sum_{i=1}^n \{2 * (y_i - w_2^T x_i)(-x_i)\}$
- 5) .
- 6) .
- 7) .

Like this, we have to repeat the steps from 2 to the end, and compute the values of $w_1, w_2, w_3, \dots, w_k, w_{k+1}$, as long as our vector doesn't change much.

If $w_{k+1} - w_k$ is a vector of very small values, then we declare $w^* = w_k$.

So for any iteration, $w_j = w_{j-1} - r * \sum_{i=1}^n \{2 * (y_i - w_{j-1}^T x_i)(-x_i)\}$

Here as we move from ' w_{j-1} ' to ' w_j ', for every iteration, if 'n' is very large, then it is computationally expensive.

Gradient Descent becomes very slow if the number of computations is large. In order to get rid of these kinds of problems, we have an alternative optimizer called **Stochastic Gradient Descent**.

35.8 - SGD Algorithm

So far we have seen the Gradient Descent algorithm in training a linear regression model. The update equation is given as

$$\mathbf{w}_{j+1} = \mathbf{w}_j - r^* \sum_{i=1}^n (-2x_i)(y_i - \mathbf{w}_j^T \mathbf{x}_i)$$

When 'n' is extremely large, this solution becomes computationally expensive and takes a lot of time, as the number of computations is more.

There is an algorithm to overcome this issue. It is called Stochastic Gradient Descent which is a modification done over the Gradient Descent.

Stochastic Gradient Descent says instead of choosing all the data points and perform summation over them in Gradient Descent, here we just have to pick a random set of 'K' points and perform the same update step for multiple iterations. After some iterations, the value w^* obtained by SGD is the same as the w^* obtained by GD.

In SGD,

$$\mathbf{w}_{j+1} = \mathbf{w}_j - r^* \sum_{i=1}^K (-2x_i)(y_i - \mathbf{w}_j^T \mathbf{x}_i) \quad (\text{where } 1 < K < n)$$
$$w_{GD}^* = w_{SGD}^*$$

Here we use a random set of points. Hence we call this variant of Gradient Descent as **Stochastic Gradient Descent**.

For example,

In Gradient Descent, if $n = 1$ million and if our problem requires 100 iterations to converge to x^* . In Stochastic Gradient Descent, if $K=1000$, then for the same problem, we need more than 100(say 500) iterations to converge to x^* .

$K \rightarrow$ Number of points that are taken at each iteration for the update formula

The set of the random points chosen in each iteration are different from the sets of random points chosen in successive iterations.

Here in SGD, 'K' is called Batch Size, as we choose a batch of random points.

If $K=1 \rightarrow$ we call it simply SGD.

If $K>1 \rightarrow$ we call it Batch SGD with batch size = 100 (say)

If $K=n \rightarrow$ then SGD becomes GD.

35.9 - Constrained Optimization and PCA

So far we have seen optimization problems like $\max_x f(x)$ (or) $\min_x f(x)$.

But the formulation of PCA was

$$\max_u \left(\frac{1}{n} \sum_{i=1}^n (u^T x_i)^2 \right) \text{ such that } u^T u = 1$$

Here $\left(\frac{1}{n} \sum_{i=1}^n (u^T x_i)^2 \right) \rightarrow$ Objective Function, $u^T u = 1 \rightarrow$ Constraint.

This formulation is similar to
 $\max_x f(x)$ such that $g(x)=c$.

Such an optimization problems with constraints is called **Constrained Optimization Problem**.

General Constrained Optimization Problem

It looks of the form $\max_x f(x) \rightarrow$ objective function
Such that $g(x)=c \rightarrow$ equality constraint
 $h(x) \geq d \rightarrow$ inequality constraint

If an inequality constraint of the form $K(x) \leq e$ is given, we immediately have to convert it into \geq form by multiplying with '-' and making it $-K(x) \geq -e$.

How to find Maxima and Minima when constraints are given

The optimization problem is of the form
 $x^* = \max_x f(x)$ such that $g(x)=c, h(x) \geq d$

We shall modify this problem using Lagrangian Multipliers. Here using the given objective function, we shall construct a new function called **Lagrangian**, denoted by 'L'.

So now,

Lagrangian $L(x, \lambda, \mu) = f(x) - \lambda\{g(x)-c\} - \mu\{d-h(x)\}$
where $\lambda, \mu \rightarrow$ Lagrangian Multipliers. Here $\lambda \geq 0, \mu \geq 0$.

We should compute parallel derivatives of 'L' with respect to 'x', ' λ ' and ' μ ' separately and equate them to 0.

$$\partial L/\partial x = 0, \partial L/\partial \lambda = 0, \partial L/\partial \mu = 0$$

Here we get 3 equations and if we solve them, we get x_{tilde} , λ_{tilde} and μ_{tilde} . We can ignore λ_{tilde} and μ_{tilde} for the time being, the value of x_{tilde} obtained here is equal to x^* for the problem.

Let us now come back to the optimization problem of PCA.

$$\max_u (1/n) \sum_{i=1}^n (u^T x_i)^2 \text{ such that } u^T u = 1.$$

The same above constrained optimization problem can be written as
 $\max_u u^T S u \text{ such that } u^T u = 1$

where 'S' is the covariance matrix of 'X' with mean centred variance '1' and is given as $S = (1/n) \sum_{i=1}^n x_i^T x_i$

The Lagrangian form for this function is written as

$$L(u, \lambda) = u^T S u - \lambda(u^T u - 1)$$

$$\partial L/\partial u = 0 \Rightarrow \partial/\partial u (u^T S u - \lambda u^T u + \lambda) = 0 \quad \dots (1)$$

$$u^T u = u^2 \quad \dots (2)$$

So now, if we substitute (2) in (1), then we get

$$\partial/\partial u (S u^2 - \lambda u^2 + \lambda) = 0$$

The result obtained after derivation is

$$2S u - 2\lambda u = 0$$

We are cancelling out the number 2. So the equation becomes

$$S u = \lambda u$$

Here $u \rightarrow$ eigen vector of 'S'

$\lambda \rightarrow$ eigen value of 'S'

'S' \rightarrow Covariance Matrix

The best 'u' corresponds to the highest eigen value of 'S'.

35.10 - Logistics Regression Formulation Revisited

The optimization problem of Logistic Regression looks like

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} (\text{Logistic Loss}) + \lambda \mathbf{w}^T \mathbf{w} \text{ such that } \mathbf{w}^T \mathbf{w} = 1$$

The goal here is to minimize the sum of Logistic Loss and the regularization term. Here ' w ' is a vector normal to the hyperplane.

The above given optimization problem is a constrained optimization problem.

So we can convert this constrained optimization problem into non-constrained optimization problem using Lagrangian multipliers.

$$\text{So } L = \text{Logistic Loss} - \lambda(1 - \mathbf{w}^T \mathbf{w})$$

$$L = \text{Logistic Loss} - \lambda + \lambda \mathbf{w}^T \mathbf{w}$$

35.11 - Why L1 Regularization creates sparsity?

Optimization problem with L2 regularization $\Rightarrow \min_w (\text{loss} + \lambda ||w||_2^2)$,
 Optimization problem with L1 regularization $\Rightarrow \min_w (\text{loss} + \lambda ||w||_1)$.

In both these optimizations, ' λ ' and the loss term are the same. So we are left with minimizing only the terms $||w||_1$ and $||w||_2^2$. So now the above two optimizations become $\min_w (||w||_2^2)$ and $\min_w (||w||_1)$ respectively.

$$\min_w (||w||_2^2) = \min_{w_1, w_2, w_3, \dots, w_d} (w_1^2 + w_2^2 + w_3^2 + \dots + w_d^2)$$

$$\min_w (||w||_1) = \min_{w_1, w_2, w_3, \dots, w_d} (|w_1| + |w_2| + |w_3| + \dots + |w_d|)$$

Now as the above two optimizations are having each coefficient as a separate term (ie., we do not see multiple coefficients in the same term like $w_1 w_2$, $w_1 w_2 w_3, w_1/w_2$, etc), let us consider the two optimizations only from ' w_1 ' point of view. So now the optimizations become

$$\min_w (||w||_2^2) = \min_{w_1} (w_1^2)$$

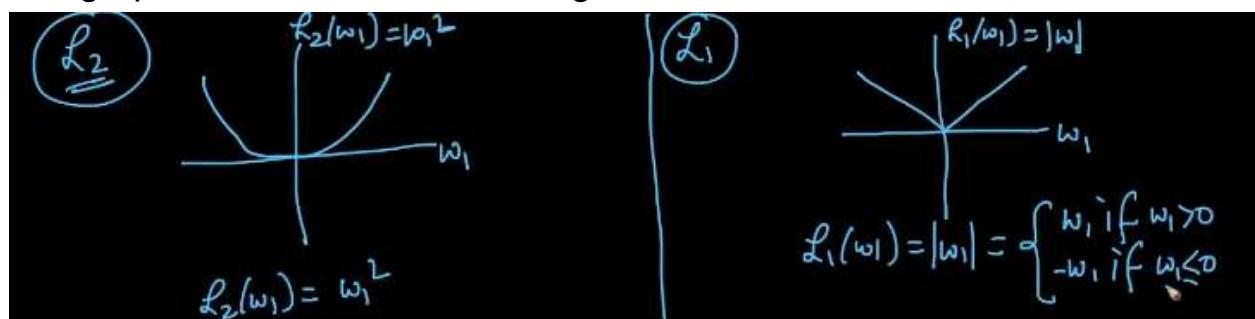
$$\min_w (||w||_1) = \min_{w_1} (|w_1|)$$

Let us represent these functions as

$$L_2(w_1) = w_1^2$$

$$L_1(w_1) = |w_1|$$

The graphs of these functions are given below



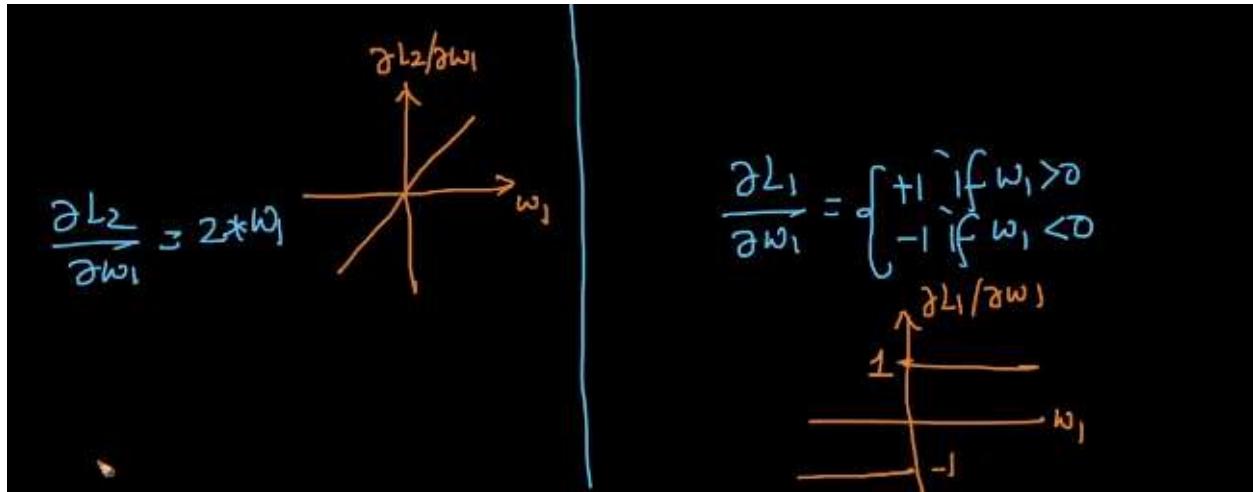
Now let us compute the derivatives of the functions $L_2(w_1)$ and $L_1(w_1)$

$$dL_2/dw_1 = 2 * w_1$$

$$dL_1/dw_1 = \{1, \text{ if } w_1 > 0$$

$$-1, \text{ if } w_1 < 0\}$$

The graphs of the derivatives of these functions look like below



Now let us look at the update step

$$(w_1)_{(j+1)} = (w_1)_j - r * [df/dx]_{(w1)_j}$$

For the timebeing, we shall assume ' w_1 ' value is only positive. Here when we look at the slopes of both the functions $L_1(w_1)$ and $L_2(w_1)$,

$$\partial L_2 / \partial w_1 = 2 * w_1$$

$$\partial L_1 / \partial w_1 = \begin{cases} 1 & \text{if } w_1 > 0 \\ -1 & \text{if } w_1 < 0 \end{cases}$$

We can clearly observe that the slope of $L_1(w_1)$ is independent of the ' w_1 ' value, whereas the slope of $L_2(w_1)$ is dependent on ' w_1 ' value.

So when we apply the regularization, the update step becomes

$$(w_1)_{(j+1)} = (w_1)_j - r * [2 * w_1]_{(w1)_j} \quad (\text{In case of L2 regularization})$$

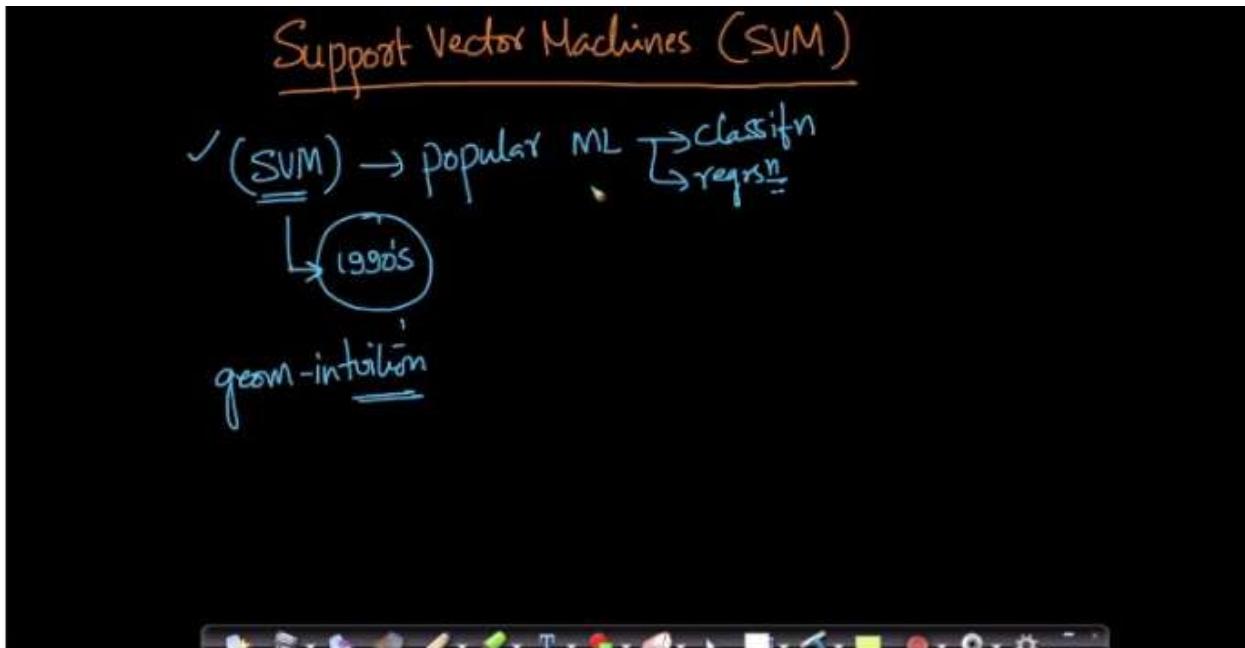
$$(w_1)_{(j+1)} = (w_1)_j - r * [1]_{(w1)_j} \quad (\text{In case of L1 regularization})$$

So we see the slope of $L_2(w_1)$ is dependent on ' w_1 ', if the value of ' w_1 ' is very small, then the slope will be a very small value, and the product of ' r ' with the slope will become much more smaller, taking smaller steps to move towards the minima, there by taking a huge number of iterations.

Whereas in the case of $L_1(w_1)$, as it's slope is independent of ' w_1 ', it takes a constant step size in the weight update step and thereby converging faster. In the process of converging faster, the weights are reduced to the optimal values(ie., most of them become 0) quickly when compared to that of $L_2(w_1)$. Hence we can say L1 regularization introduces sparsity.

In this chapter we will look at an algorithm called Support Vector Machines(SVM).

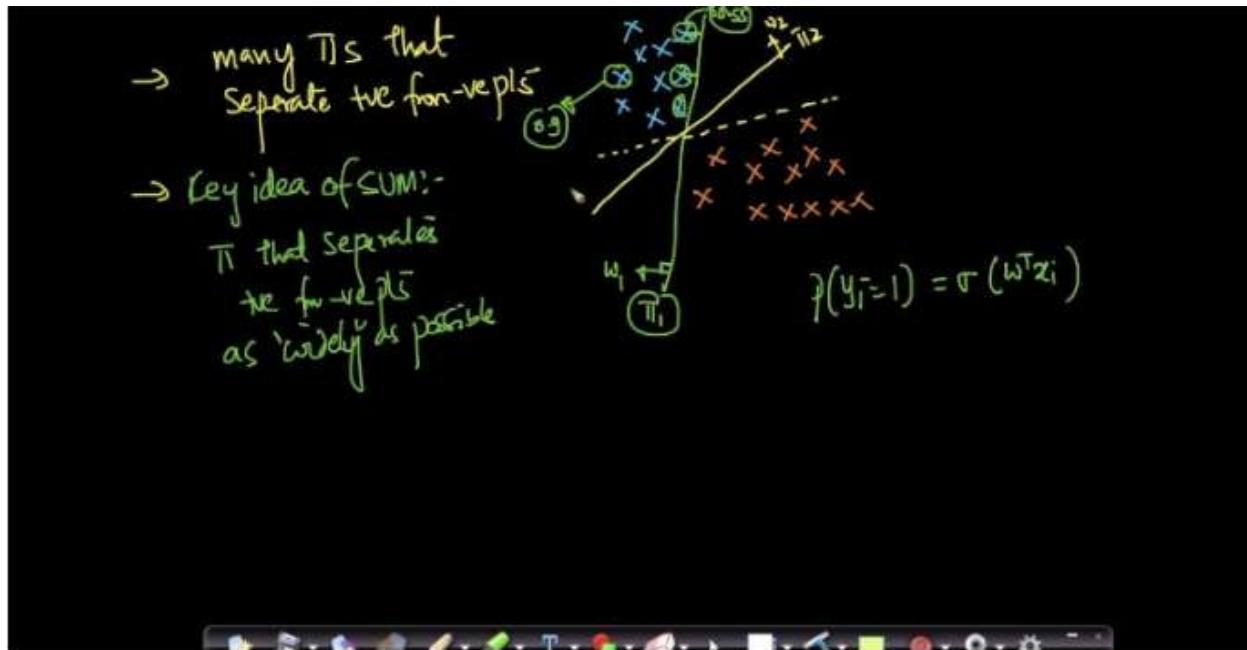
36.1 Geometric Intuition



Timestamp 1:01

Support Vector Machines(SVM) is a very popular algorithm. It was developed during the 1990's and it is still very popular. It can handle both regression and classification problems.

Let's first talk about the geometric intuition behind this algorithm.

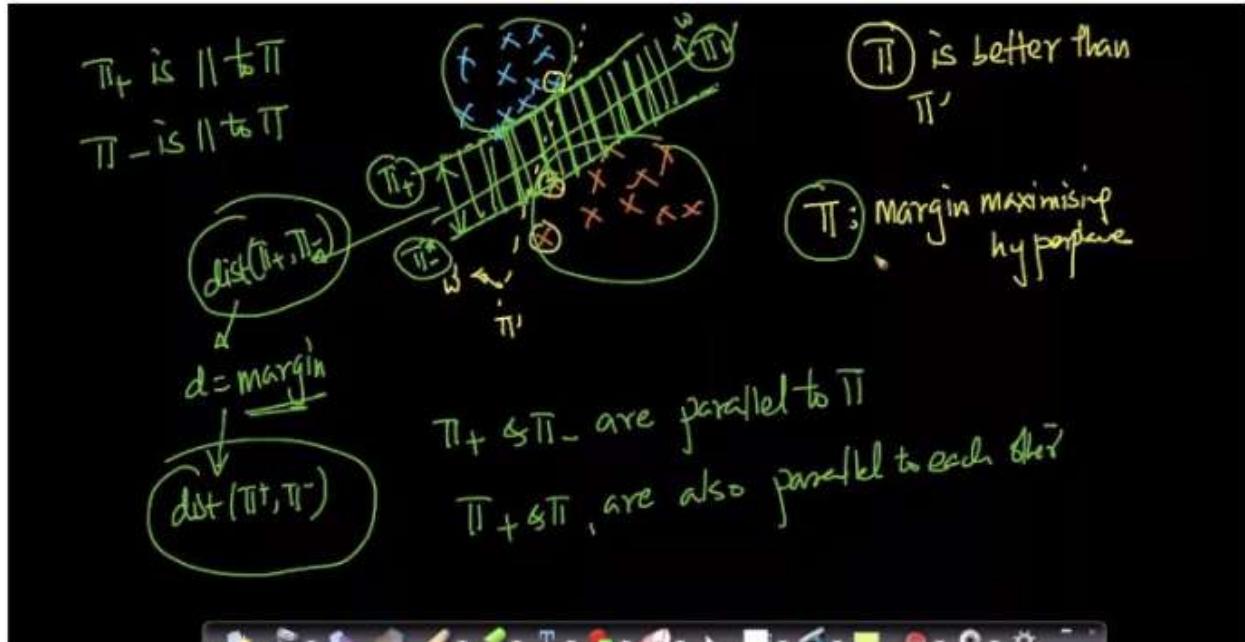


Timestamp 4:10

Suppose we have a dataset containing positive and negative points as shown in the image. Say the red crosses are negative points and the blue ones are positive. Now we want to find a hyperplane such that it separates these points. We will represent our hyperplane by π , each hyperplane has a normal vector, say w .

Now there can be multiple hyperplanes which can separate these points like π_1 , π_2 .etc as shown in the image above. Say we select π_1 as our separating hyperplane. If we do so we can see that some of the positive points are very close to the hyperplane π_1 as shown in the image. Now for points closer to the hyperplane, algorithms like logistic regression will have very low confidence in its prediction whereas for points farther away from the hyperplane it will have a greater confidence in its prediction. So following this argument, we can conclude that in our dataset π_2 is a better hyperplane than π_1 .

So the key idea of SVM is to find a hyperplane such that it separates the dataset as "widely" as possible.



Timestamp 8:45

Now the question is how to define “widely”.

Here in the above image we have again taken the previous dataset and redrawn the hyperplanes. So from the intuition that we developed in the previous slide we know that π is better than π' . The hyperplane π is called the margin maximizing hyperplane.

Say we have drawn hyperplanes parallel to π towards the positive data points. The hyperplane which first intersects with a positive data point can be termed as positive hyperplane π_+ .

Similarly we can find a negative hyperplane π_- by going towards the negative data points. The distance between these two hyperplanes is called margin. Also note that π_+ , π , π_- are all parallel to each other. We want to find a hyperplane such that it maximizes this margin.

If this distance is maximized then our positive and negative points are as far away from the separating hyperplane as possible.

SUM :- Try to find a π that maximizes the margin
margin = $\text{dist}(\pi^+, \pi^-)$

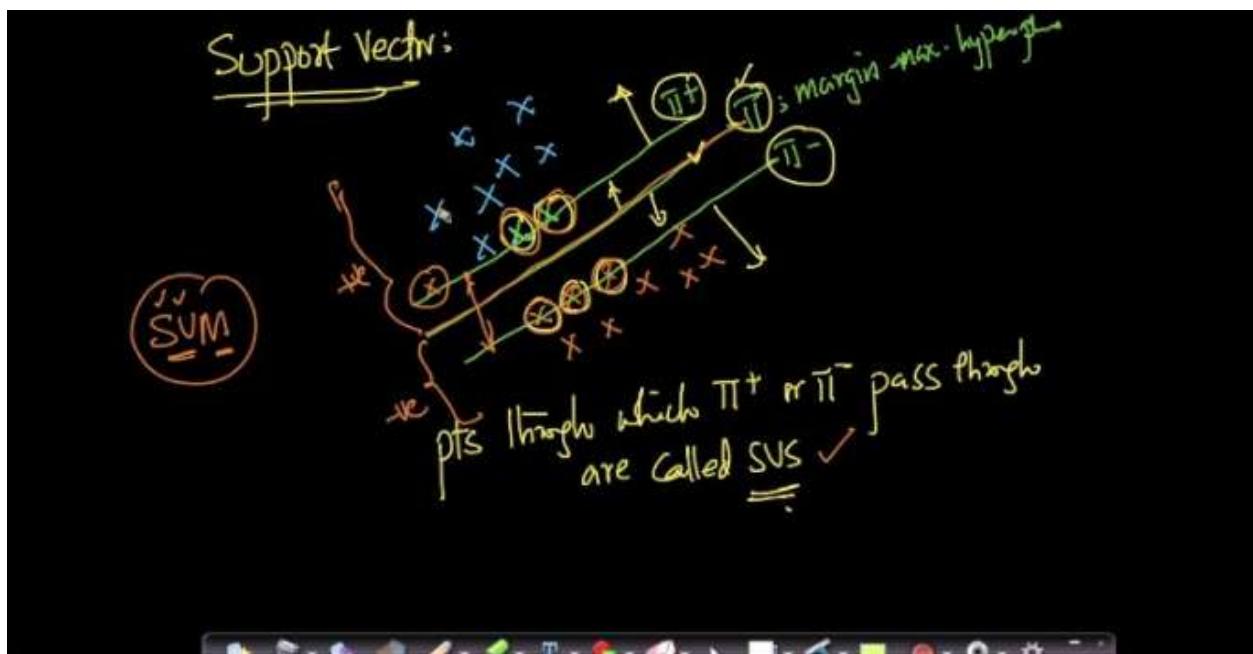
Margin ↑ generalization acc ↑

Timestamp 10:15

So in SVM's we essentially try to find a hyperplane π such that it maximizes the

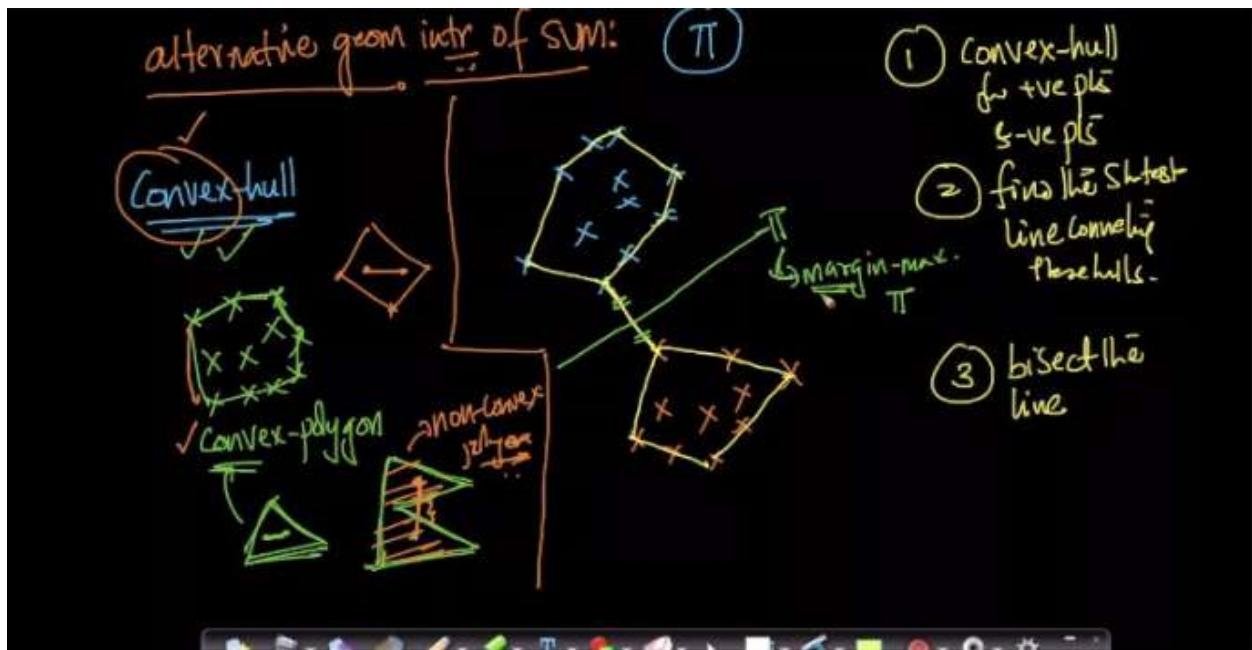
$$\text{margin} = \text{distance}(\pi^+, \pi^-)$$

Also there is a nice theory which says that as the margin increases, accuracy on unseen data i.e. generalization accuracy increases.



Timestamp 13:08

The points through which our positive and negative hyperplanes pass are called support vectors. We can see the points which are passing through the hyperplanes in the above image, we have circled them. These points or support vectors are very important as we will see in later lectures.



Timestamp 18:55

There is a nice alternative geometric interpretation of SVM. For this we need to define a few terms.

Convex Polygon - A convex polygon is a simple polygon in which no line segment connecting two points can ever go outside the polygon. For example, triangles as we have shown in the image above.

Convex Hull - A convex hull is the smallest convex polygon that contains all the data points. In order to draw a convex hull we need to draw a convex polygon such that it contains all the data points. It should also be the smallest so it may pass through some of the data points as shown in the above image.

Now coming back to SVM's. In order to find the margin maximizing hyperplane π :

1. Draw separate convex hulls for both positive and negative points as shown in the image above.
2. Find the smallest line connecting these convex hulls.
3. Bisect the smallest line that we found joining the convex hulls. This particular bisection is our margin maximizing hyperplane π .

36.2 Mathematical Derivation

Mathematical formulation of SVM:

π^+ = margin maximization

Let $\pi: w^T x + b = 0$

if $\pi^+: w^T x + b = 1$

$\pi^-: w^T x + b = -1$

Margin: $d = \frac{2}{\|w\|}$

amar@appliedroots.com, 103

Timestamp 3:17

Here we will try to build the mathematical formulation of SVM.

Suppose we have a margin maximizing hyperplane π with a normal w . Also let, π^+ and π^- be positive and negative hyperplanes respectively. The normal w will also be perpendicular to both π^+ and π^- as they are both parallel to π .

$$\text{Let } \pi \Rightarrow w^T x + b = 0$$

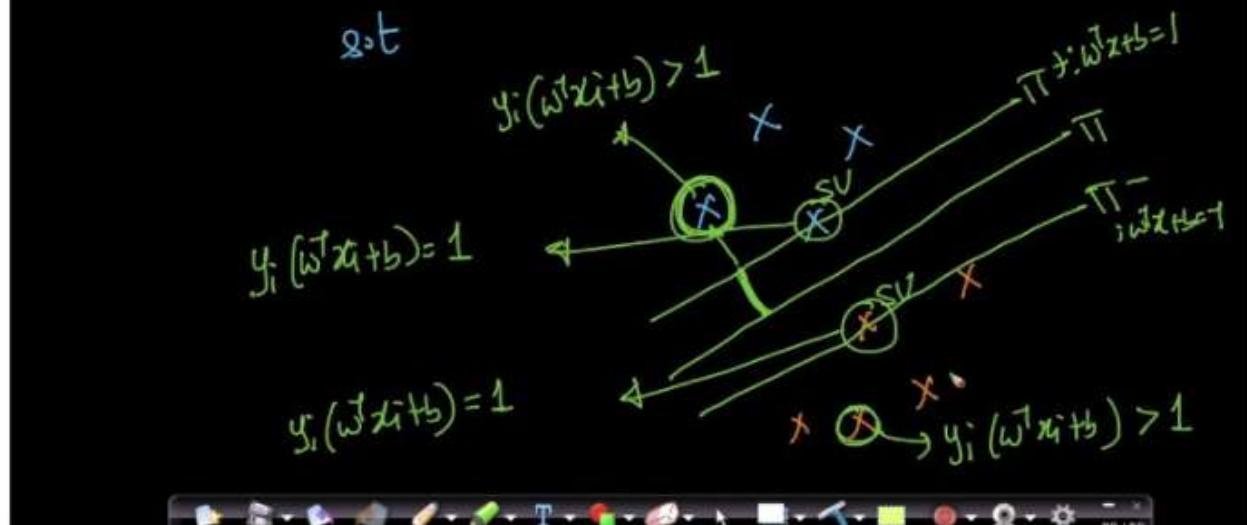
$$\text{For now assume that } \pi^+ \Rightarrow w^T x + b = 1 \text{ and } \pi^- \Rightarrow w^T x + b = -1.$$

Now the distance between these hyperplanes will be

$$= 2 / \|w\|$$

Also note that w is a normal vector not a unit vector.

$$(\omega^*, b^*) = \arg \max_{\omega, b} \frac{2}{\|\omega\|} = \text{Margin}$$



Timestamp 7:44

So we want to maximize this margin. We need to find a (w^*, b^*) such that,

$$(w^*, b^*) = \operatorname{argmax}_{(w, b)} 2 / \|w\|$$

The above equation will have some constraints. Let's talk about those, say we represent our positive points by +1 and negative points by -1. So $y_i \in \{+1, -1\}$

Now for any positive support vector we have $y_i = 1$ and also the quantity $w^T x_i + b = 1$. So the term $y_i^*(w^T x_i + b)$ becomes 1.

Also for any negative support vector we have $y_i = -1$ and also the quantity $w^T x_i + b = -1$. So again the term $y_i^*(w^T x_i + b)$ becomes 1.

Also for positive points above the positive hyperplane i.e. in the direction of w we have $y_i^*(w^T x_i + b) > 1$ and similarly for negative points below the negative hyperplane i.e. opposite to the direction of w we have $y_i^*(w^T x_i + b) > 1$. These points can be referred to as non support vectors.

CONSTN.
optimization
prob.,
of SVM

$$\left\{ \begin{array}{l} w^*, b^* = \underset{w, b}{\operatorname{argmax}} \frac{2}{\|w\|} \\ \text{s.t. } \forall i, y_i(w^T x_i + b) \geq 1 \end{array} \right.$$

Timestamp 9:44

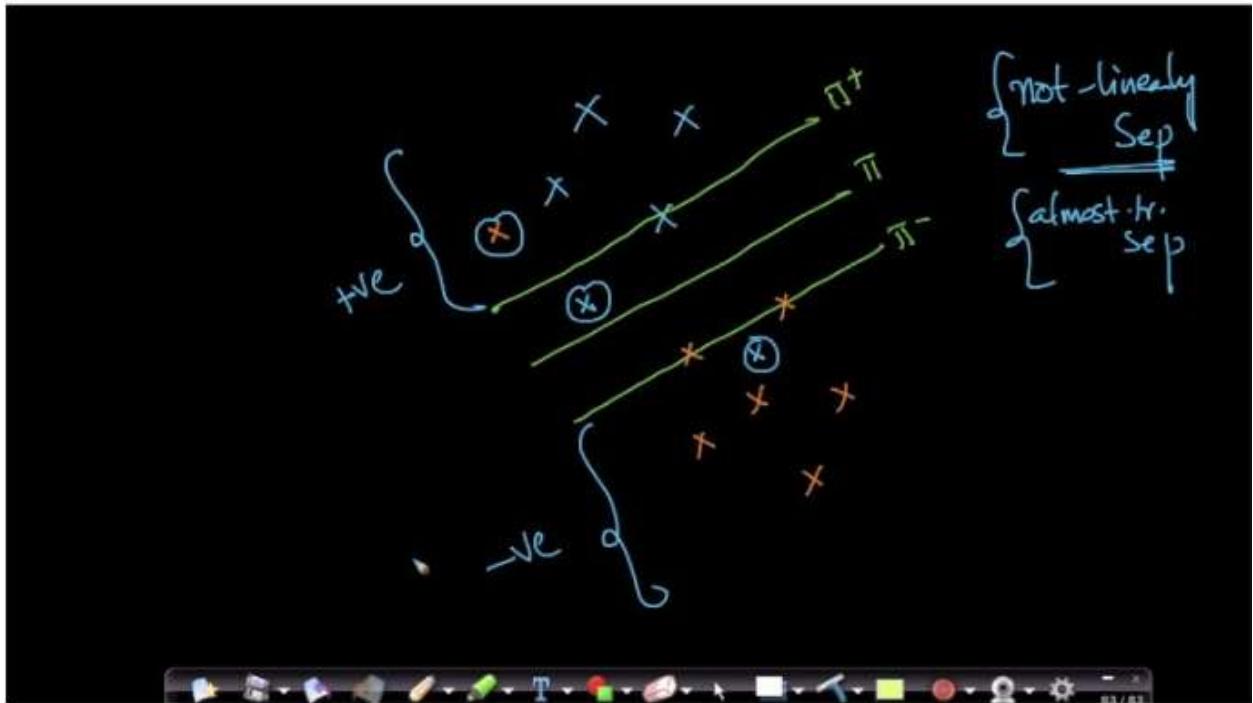
So, our final optimization problem is

$$(w^*, b^*) = \underset{(w, b)}{\operatorname{argmax}} \frac{2}{\|w\|},$$

s.t. $\forall i, y_i^*(w^T x_i + b) \geq 1$

Note that we have a total of n constraints for n data points, because we want the constraint to hold true for all data points. The equalities in the constraint are for the support vectors and the greater than part are for the non support vectors.

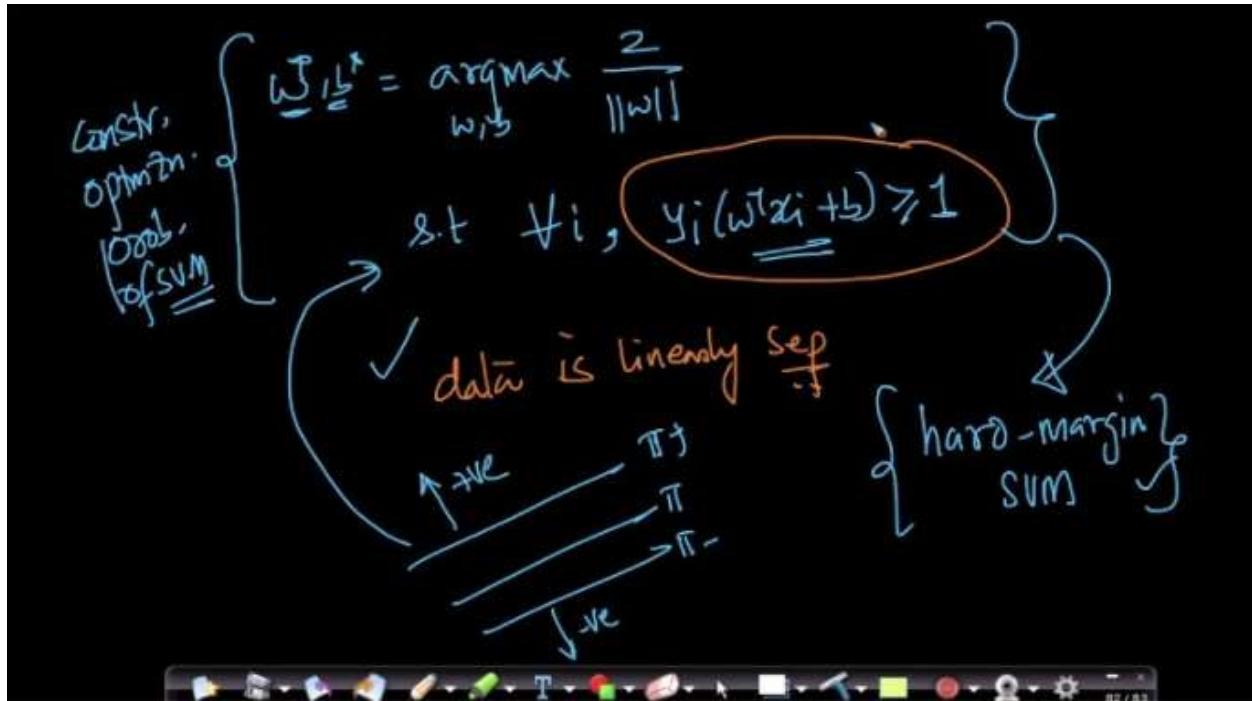
This type of optimization problems are called constraint optimization problems. We typically solve them using techniques like lagrangian multipliers.



Timestamp 12:32

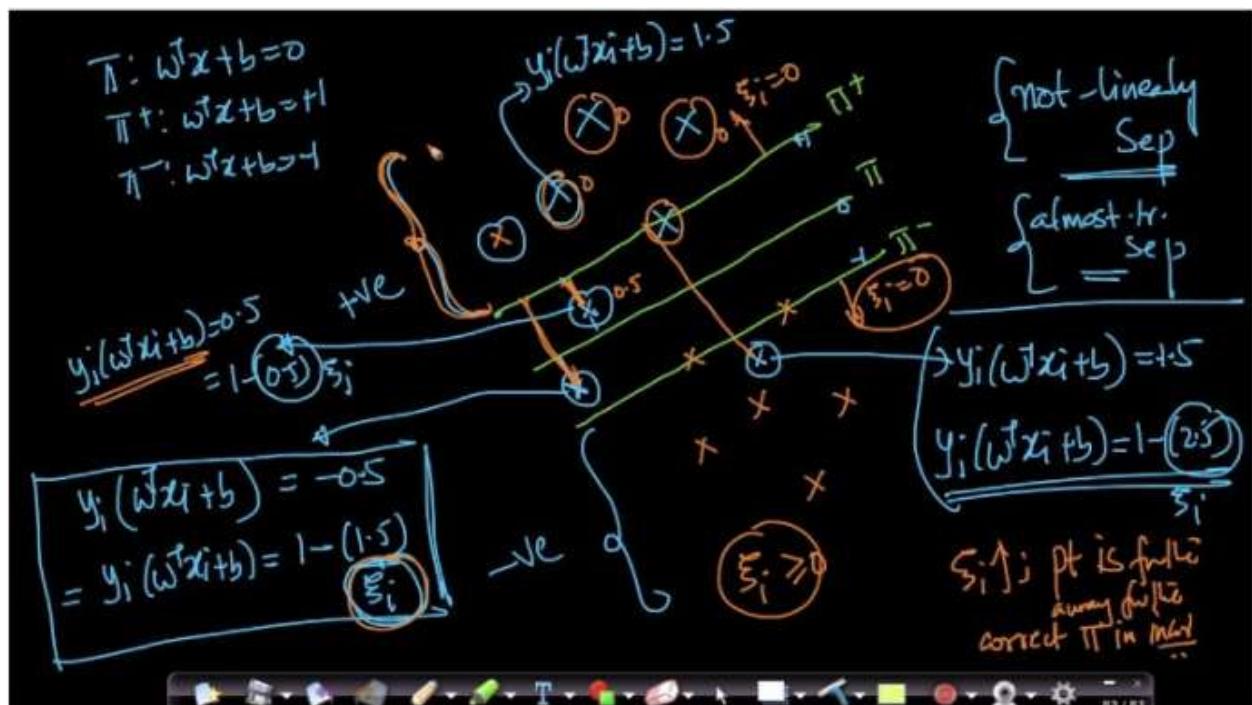
There is one issue with this optimization problem. Consider a dataset where the dataset is not linearly separable. As you can see from the image above there are some positive points which lie below the negative hyperplane and some negative points lie above the positive hyperplane. Such a dataset isn't linearly separable. It is almost linearly separable; only some points lie on the wrong side of the hyperplane.

So the constraints in our optimization problem i.e. $y_i^*(w^T x_i + b) \geq 1$ may not be satisfied for some of the data points.



Timestamp 13:48

This is the reason that this formulation is called hard margin SVM because it strictly requires the data points to be linearly separable i.e. all the data points should satisfy the constraint.



Timestamp 19:20

Now we need to handle the case where the points are not completely linearly separable. Consider a positive data point which lies on the negative side of the hyperplane as shown in the image above, say this point is -1.5 units away from the correct(positive) side of π . So in this case we can write

$$y_i^*(w^T x_i + b) = -1.5$$

$$\text{or, } y_i^*(w^T x_i + b) = 1 - 2.5$$

We are subtracting it from 1 because ideally we want $y_i^*(w^T x_i + b) = 1$.

Similarly, say a positive data point lies between positive and negative hyperplane. Say it lies -0.5 units away from the correct(positive) side of π . So in this case we can write

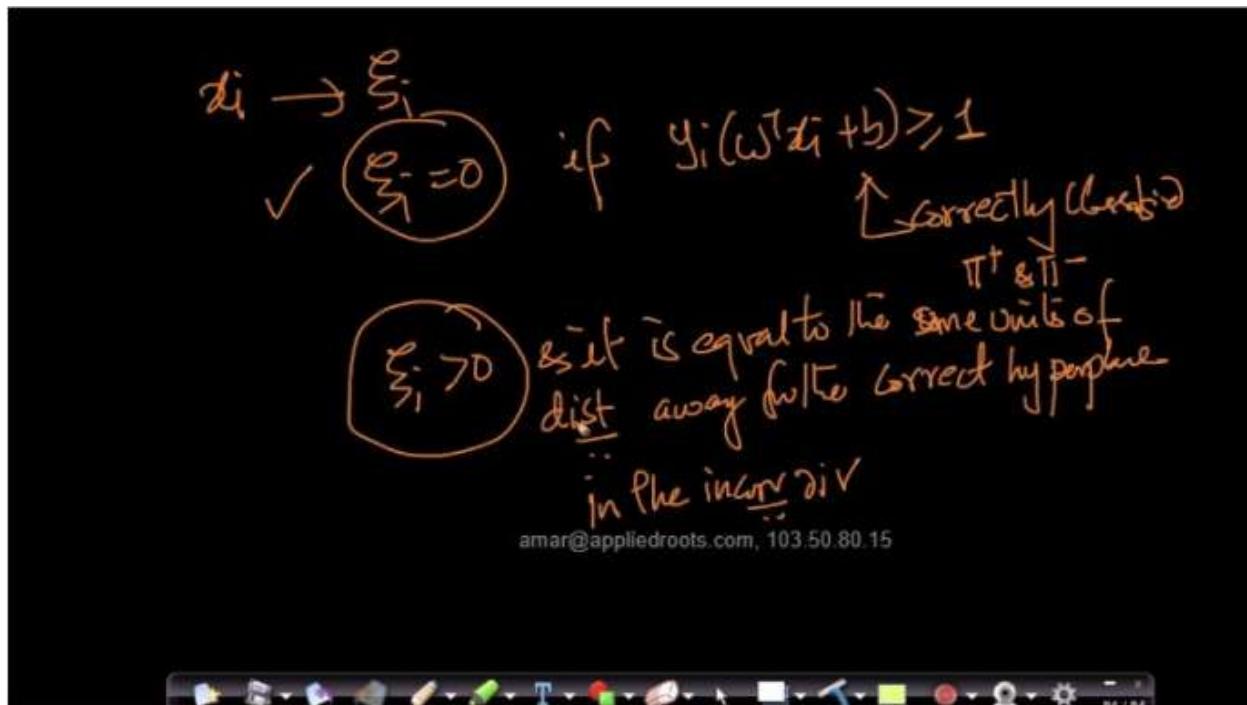
$$y_i^*(w^T x_i + b) = -0.5$$

$$\text{or, } y_i^*(w^T x_i + b) = 1 - 1.5.$$

We can continue like this for all the misclassified points as shown in the image above and calculate the misclassified distance. In general, we write the misclassified distance as ξ (zeta). For correctly classified points ξ is 0. For incorrectly classified points $\xi > 0$.

So in general $\xi \geq 0$.

A higher value of ξ indicates that the point is farther away from the correct hyperplane.



Timestamp 21:20

So to summarize,

For every datapoint x_i , we are getting a ξ_i , $x_i \rightarrow \xi_i$.

Now this $\xi_i = 0$, if $y_i^*(w^T x_i + b) \geq 1$ i.e. point is correctly classified w.r.t π_+ and π_- ,

and $\xi_i > 0$ if the point is incorrectly classified and the magnitude of ξ_i represents the units by which it is away from the correct hyperplane.

$$(\hat{w}, \hat{b}) = \underset{(w, b)}{\operatorname{argmax}} \frac{2}{\|w\|} = \underset{(w, b)}{\operatorname{argmin}} \frac{\|w\|}{2}$$

$$\max_x f(x) = \min_x \frac{1}{f(x)}$$

Timestamp 22:11

Now we will reformulate our optimization problem.

Earlier we had,

$$(w^*, b^*) = \underset{(w, b)}{\operatorname{argmax}} \frac{2}{\|w\|},$$

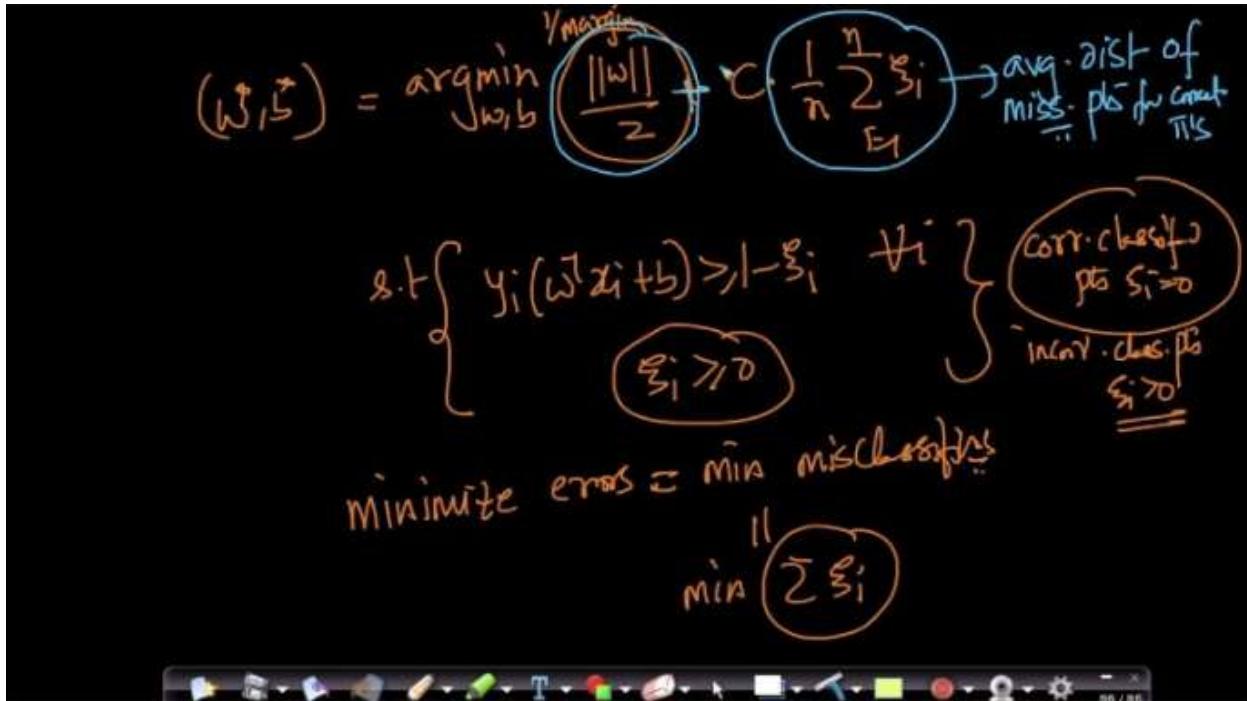
s.t. $\forall i, y_i^*(w^T x_i + b) \geq 1$

The above equation can also be written as

$$(w^*, b^*) = \underset{(w, b)}{\operatorname{argmin}} \frac{\|w\|^2}{2},$$

s.t. $\forall i, y_i^*(w^T x_i + b) \geq 1$

This is because the $\max f(x) = \min 1/f(x)$, when $f(x) \neq 0$



Timestamp 25:14

So, our optimization problem is

$$(w^*, b^*) = \operatorname{argmin}_{(w, b)} \|w\|/2 + C * \frac{1}{n} \sum_{i=1}^n \xi_i$$

$$\text{s.t. } \forall i, y_i^*(w^T x_i + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0$$

where C is the hyperparameter and n is the no of data points.

$\frac{1}{n} \sum_{i=1}^n \xi_i \Rightarrow$ this term basically represents the average distance of the misclassified points, so

we want to reduce it.

This formulation is called soft margin SVM. It allows errors unlike hard margin SVM, but asks to minimize them.

Timestamp 27:55

We can think of this term $\Rightarrow 1/n \sum_{i=1}^n \xi_i$ as our loss function because we want to reduce it. This loss is called hinge loss.

Also, $\|w\|^2 / 2$ can be thought of as a regularizer. We have already seen a similar formulation while learning logistic regression as you can see in the image above.

So we can say that our formulation is something like this

$\text{argmin}_{(w,b)} C * \text{hinge loss} + \lambda (\text{reg})$, where C is the hyperparameter

$C \uparrow$; tendency to make mistakes ↓
on D_{train}
 \Rightarrow overfit \Rightarrow high-variance

$C \downarrow$; underfit \Rightarrow high-bias

Timestamp 29:38

C is always ≥ 0 .

If the value of C is high it means we cannot have a very high hinge loss. Meaning less misclassification. This results in overfitting \Rightarrow high variance.

If the value of C is low it means we can have a high hinge loss. Meaning more misclassification. This results in underfitting \Rightarrow high bias.

36.3 Why we take values +1 and -1 for Support vector planes

SVM:

(Q) Why +1 & -1 on the RHS of $\Pi^+ \& \Pi^-$

margin: $\frac{2}{\|w\|}$

$w^T b^P = \max_{w, b} \frac{2}{\|w\|}$ (any vecr)
constraints

$\Pi^+: w^T x + b = +1$
 $\Pi^0: w^T x + b = 0$
 $\Pi^-: w^T x + b = -1$
 $w \perp \Pi$

$\left\{ \begin{array}{l} \|w\| \neq 1 \\ \text{(any vecr)} \\ \text{need not be} \\ \text{unit vecr} \end{array} \right.$

Timestamp 2:43

Here we will discuss a question that was asked by one of our students.

Why are we taking -1 and +1 in the RHS of the negative and positive hyperplanes respectively?

Note that in previous lectures we have said w is not a unit vector. Also the margin between the hyperplanes was $2 / \|w\|$ and we wanted to maximize this based on some constraints.

$\pi^+ :- w^T x + b = K$
 $\pi^- :- w^T x + b = -K$
 $K > 0$ (Note: x_1 and x_2 are labeled with minus signs)
 K : Constant
 $K=4$
Margin: $\frac{2K}{\|w\|}$
 $\underset{w,b}{\operatorname{argmax}} \frac{2}{\|w\|} = \underset{w,b}{\operatorname{argmax}} \frac{2K}{\|w\|} = \frac{8}{\|w\|}$

Timestamp 4:49

We will try to answer this question from multiple approaches.

1. Consider a case where we would have taken a constant k instead of 1 in the RHS. In that case our equations would have been

$$\pi^+ \Rightarrow w^T x + b = k \text{ and } \pi^- \Rightarrow w^T x + b = -k.$$

Keep in mind that the hyperplane π needs to be in equal distance from both positive and negative hyperplanes, that's why we have taken the same RHS in both cases. k can take any positive value i.e. $k > 0$ and it's a constant.

So, our margin becomes $2k / \|w\|$.

Then, our equation becomes

$$(w^*, b^*) = \underset{(w, b)}{\operatorname{argmax}} 2k / \|w\|, \text{ which is equivalent to}$$

$$(w^*, b^*) = \underset{(w, b)}{\operatorname{argmax}} 2 / \|w\|$$

because k is just a constant and both these equations will yield the same results.

So, for ease of calculation we have taken the value of $k = 1$.

② $\pi^+ : w^T x + b = k$

$\left(\frac{w}{k} \right)^T x + \left(\frac{b}{k} \right) = 1$

$w \perp \pi$ $\|w\|$ need not be 1

$\boxed{\left(\frac{w}{k} \right)^T x + \frac{b}{k} = 1}$

Timestamp 6:47

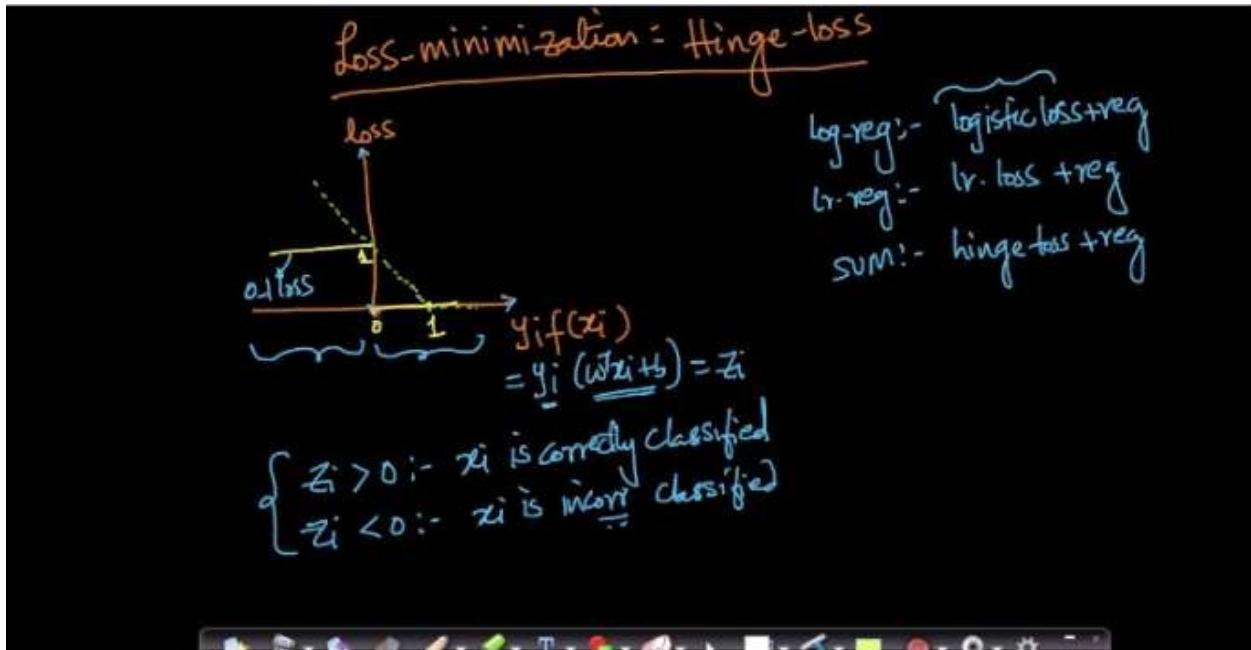
2. Say we taken our positive hyperplane as $\pi^+ \Rightarrow w^T x + b = k$.

Now we can divide both sides by k, then, $\pi^+ \Rightarrow \left(\frac{w}{k} \right)^T x + \left(\frac{b}{k} \right) = 1$.

We can rewrite this as $\pi^+ \Rightarrow \left(\frac{w}{k} \right)^T x + \frac{b}{k} = 1$

Dividing w by k which is a constant won't change anything because our only requirement is that w should be perpendicular to π^+ , which won't change if we divide it by a constant.

36.4 Loss function (Hinge Loss) based interpretation



Timestamp 2:03

Here we will try to understand SVM from the perspective of hinge loss.

Remember when we studied logistic regression, it is logistic loss + regularizer. Similarly linear regression is linear loss + regularizer.

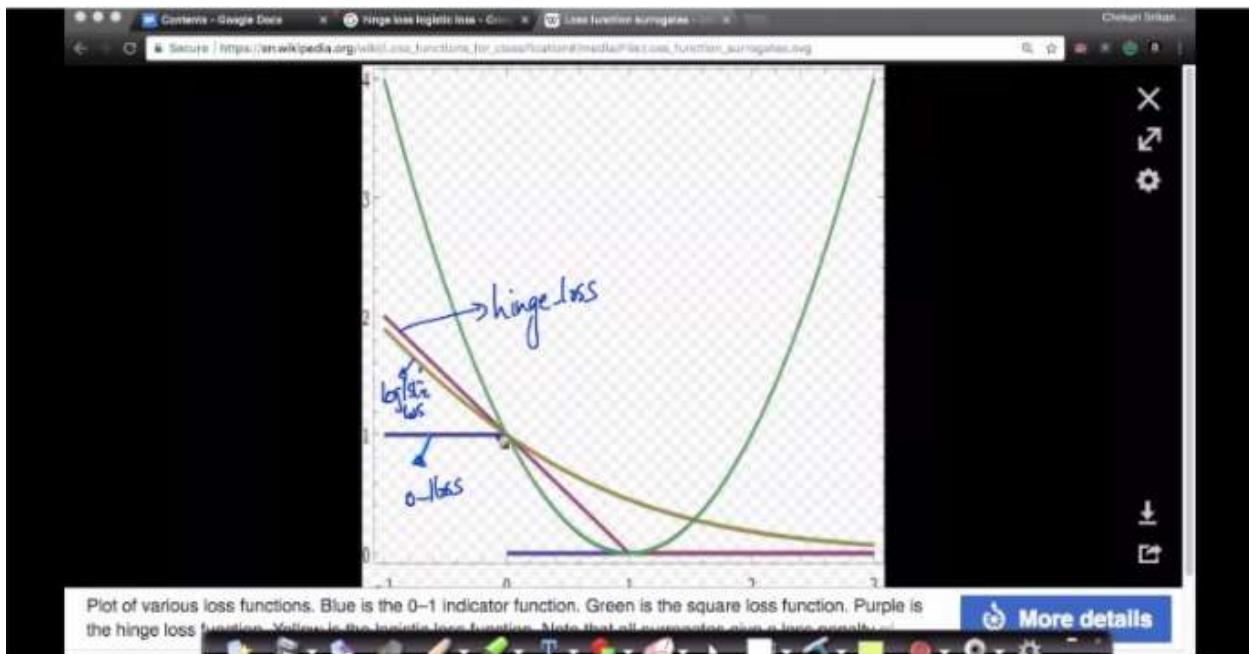
Similarly we saw that SVM is hinge loss + regularizer.

In the above image we can see a graph where on the y-axis we have loss and on the x-axis we have our prediction $y_i^* f(x_i)$. In case of SVM, $f(x_i) = y_i^* (w^T x_i + b)$. Let's call this z_i .

We can draw a 0-1 loss for this as shown by the yellow curve on the graph. Here for 0-1 loss we have loss = 1 for $z_i < 0$ and the loss = 0 for $z_i > 0$.

But as we know that 0-1 loss is not continuous at 0, hence it is not differentiable.

The green dotted line in the graph is the hinge loss.



Timestamp 2:47

We can see in the image above that the blue line represents the 0-1 loss. The brown line represents the logistic loss and the purple line represents the hinge loss. Both hinge loss and logistic loss are approximations of 0-1 loss.

$$\text{hinge-loss} := \begin{cases} z_i \geq 1; \text{ hinge-loss} = 0 \\ z_i < 1; \text{ hinge-loss} = 1 - z_i \end{cases}$$

$\hookrightarrow \max(0, 1 - z_i)$

$$\begin{cases} \text{Case 1: } z_i \geq 1 \Rightarrow 1 - z_i \text{ is -ve value} \Rightarrow \max(0, 1 - z_i) = 0 \\ \text{Case 2: } z_i < 1; 1 - z_i > 0 \Rightarrow \max(0, 1 - z_i) = 1 - z_i \end{cases}$$

Timestamp 7:08

Hinge loss is 0 if $z_i \geq 1$, and hinge loss is $1 - z_i$ if $z_i < 1$.

We can verify this using the graph that we have drawn earlier.

Hinge loss can also be written as $\Rightarrow \max(0, 1 - z_i)$

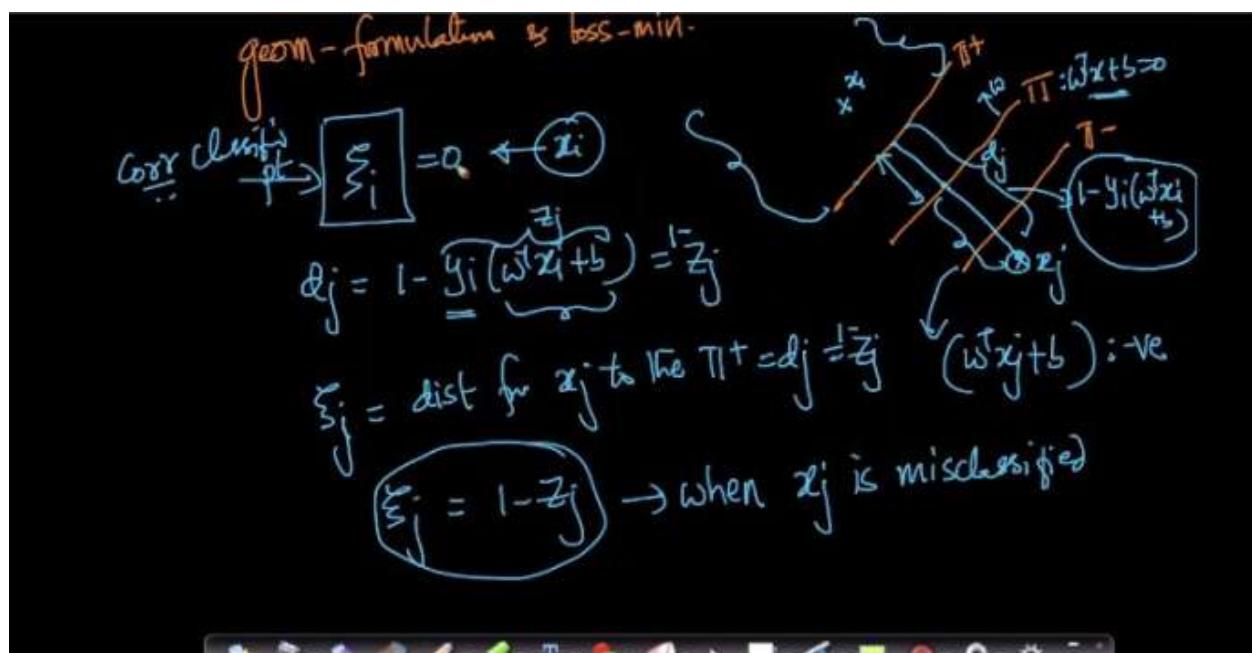
Let's check why this is true,

Case 1: if $z_i \geq 1$, then $1 - z_i$ is negative $\Rightarrow \max(0, 1 - z_i) = 0$

Case 2 : if $z_i < 1$, then $1 - z_i > 0 \Rightarrow \max(0, 1 - z_i) = 1 - z_i$

These cases match our formulation of hinge loss. This is a more compact way to write hinge loss.

Also note that hinge loss is not differentiable, so in practice we use some hacks to get around this problem.



Timestamp 11:36

Let's try to connect the geometric formulation and loss minimization.

Consider a positive point x_i which is present on the correct side of the hyperplane. ξ_i for this data point will be 0.

Consider another positive data point x_j which is present on the side of the negative hyperplane.

The distance d_j of this point from the positive hyperplane is $= 1 - y_j * (w^T x_j + b) = 1 - z_j$.

We also know that ξ_j for this point will represent the distance from the correct in our case positive hyperplane.

So, for misclassified points $\xi_j = 1 - z_j$

$$\max(0, 1 - z_i) = \xi_i$$

Timestamp 11:51

So we can see that $\max(0, 1 - z_i) = \xi_i$. We saw the verification in our previous slide.

Soft sum:

$$\min_{w, b} \frac{\|w\|^2}{2} + C \sum_{i=1}^n \xi_i$$

s.t. $(1 - y_i(\omega^\top x_i + b)) \geq \xi_i \quad \forall i$

$\xi_i > 0$

loss-Min:

$$\min_{w, b} \sum_{i=1}^n \max(0, 1 - y_i(\omega^\top x_i + b)) + \lambda \|\omega\|^2$$

$\lambda \uparrow \Rightarrow \text{Underfit}$

$\lambda \downarrow \Rightarrow \text{Overfit}$

$\|\omega\| \geq 0 \Rightarrow \min \frac{\|\omega\|}{2}$ is same as $\min \|\omega\|^2$

Timestamp 17:15

We have already seen that ξ_i is equivalent to $\max(0, 1 - z_i)$ under the constraints of ξ_i .

So, the loss minimization interpretation of soft margin SVM is

$$(w^*, b^*) = \operatorname{argmin}_{(w,b)} \sum_{i=1}^n \max(0, 1 - z_i) + \lambda \|w\|^2$$

Here, $\sum_{i=1}^n \max(0, 1 - z_i)$ => loss part

$\lambda \|w\|^2$ => It's a regularizer with λ as the hyperparameter and $\|w\|^2$ is L_2 norm of w.

The above formulation is equivalent to the one we saw during geometric interpretation i.e.

$$\begin{aligned} (w^*, b^*) &= \operatorname{argmin}_{(w,b)} \|w\|/2 + C^* 1/n \sum_{i=1}^n \xi_i \\ \text{s.t. } &\forall i, y_i^*(w^T x_i + b) \geq 1 - \xi_i, \\ &\xi_i \geq 0 \end{aligned}$$

We have already seen that ξ_i is equivalent to $\max(0, 1 - z_i)$ [$z_i = y_i^*(w^T x_i + b)$]

Also it can be seen that we have changed $\|w\|/2$ to $\|w\|^2$, because minimizing both of them are equivalent.

In the loss formulation the hyperparameter λ is multiplied to the regularizer, so if the λ increases, the model underfits and if it decreases, the model overfits.

In the geometric interpretation the hyperparameter C is multiplied to the loss, so if C increases, the model overfits, and if it decreases, the model underfits.

36.5 Dual form of SVM formulation

Dual form of SVM

Primal of SVM

Dual

$$\text{Soft Margin} = \begin{cases} \min_{w, b} & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} & y_i(w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{cases}$$

$$\max_{\alpha_i} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j$$

s.t. $\alpha_i \geq 0$

(4) $\alpha_i > 0$ only for SUS

$\alpha_i = 0$ for non-SUS

(1) $x_i \rightarrow \alpha_i$

(2) x_i 's only occur in the form of $x_i^T x_j$

(3) $f(x_q) = \sum_{i=1}^n \alpha_i y_i x_i^T x_q + b$

Timestamp 6:45

Here we will look at the dual formulation of SVM. We will take Soft Margin SVM here, because it is more useful in real life scenarios.

We earlier saw that the geometric formulation of Soft Margin SVM is

$$(w^*, b^*) = \underset{(w, b)}{\operatorname{argmin}} \|w\|^2/2 + C^* \frac{1}{n} \sum_{i=1}^n \xi_i$$

s.t. $\forall i, y_i^*(w^T x_i + b) \geq 1 - \xi_i$,

$$\xi_i \geq 0$$

This is also called the Primal Form of SVM.

Now the Dual Form of SVM is

$$(\alpha^*) = \underset{(\alpha_i)}{\operatorname{argmax}} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j$$

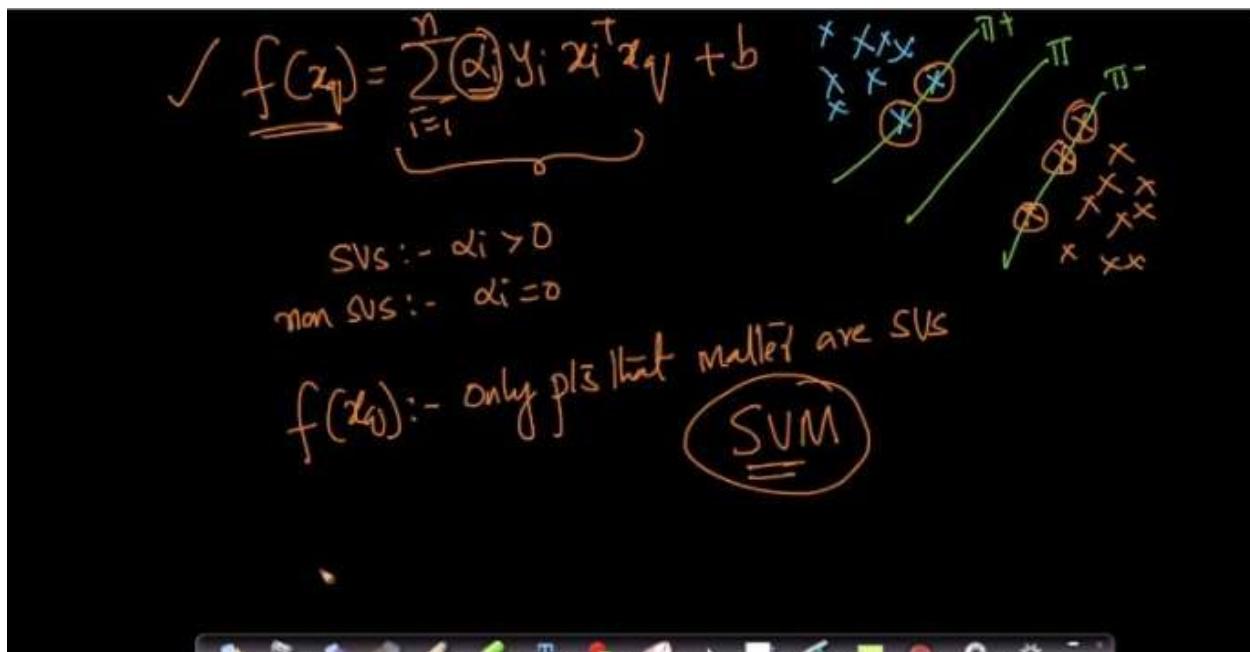
s.t. $C \geq \alpha_i \geq 0$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

Using techniques from optimization it can be shown that both primal and dual forms are equivalent. The exact proof is beyond the scope of this course.

Let's see some properties of this dual form:

1. In dual form also we are summing over n i.e. for every $x_i \rightarrow \alpha_i$
 2. x_i 's only occur in the form of $x_i^T x_j$
 3. In test time suppose we get a query point x_q , in the primal we can compute $f(x_q) = \text{sign}((w^T x_q + b))$. If the sign is positive we can call it a positive datapoint, otherwise a negative datapoint.
 - In the case of dual form we compute
- $$f(x_q) = \text{sign}\left(\sum_{i=1}^n \alpha_i y_i x_i^T x_q + b\right)$$
4. α_i 's are non-zero only for support vectors.



Timestamp 8:50

Let's look at the query function

$$f(x_q) = \sum_{i=1}^n \alpha_i y_i x_i^T x_q + b$$

We can see from this function that we are iterating over all the α_i 's, but as we know that α_i 's are non-zero only for support vectors. So for the α_i 's which are 0, the whole term for that becomes 0. So essentially we are summing over only the support vectors.

So for prediction only the support vectors matter and nothing else.

The image shows handwritten mathematical notes on a whiteboard. At the top, there is a maximization problem:

$$\max_{\alpha_i} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

Below this, there is a condition:

$$\text{s.t. } \alpha_i > 0 \quad \rightarrow \quad \begin{cases} \alpha_i = 0 \text{ for SVs} \\ \alpha_i > 0 \text{ for non-SVs} \end{cases}$$

At the bottom left, there is a circled equation:

$$\text{Sim}(x_i, x_j)$$

On the right, there is a note about cosine similarity:

$$\left\{ x_i^T x_j = x_i \cdot x_j = \overbrace{\text{Cosine Sim}(x_i, x_j)}^{\text{if } \|x_i\| = 1; \|x_j\| = 1}$$

Timestamp 12:30

We also saw that x_i 's only occur in the form of $x_i^T x_j$, here $x_i^T x_j$ is the dot product between x_i and x_j . It can be also thought of as the cosine similarity between x_i and x_j , if $\|x_i\| = \|x_j\| = 1$.

In place of cosine similarity we can use any similarity criteria $\text{sim}(x_i, x_j)$. This is what makes SVM's very powerful.

Essentially we use something known as kernel functions(K), which we will see in later chapters. So we can write our dual form as:

$$(\alpha^*) = \underset{(\alpha_i)}{\operatorname{argmax}} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

s.t. $\alpha_i \geq 0$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

Run-time

$$f(x_q) = \sum_{i=1}^n \alpha_i y_i K(x_i, x_q) + b$$

Timestamp 13:59

We can see that x_i 's also occur in the form of $x_i^T x_j$ in the query function. So, here also we can use our kernel functions.

$$f(x_q) = \sum_{i=1}^n \alpha_i y_i K(x_i, x_q) + b$$

36.6 kernel trick

Kernel Trick:

$$\left\{ \begin{array}{l} \max_{\alpha_i} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t. } \sum_{i=1}^n \alpha_i y_i = 0, \alpha_i \geq 0 \end{array} \right.$$

Sim(x_i, x_j)
 $K(x_i, x_j)$
 Kernel fn

$$\left\{ f(x_0) = \sum_{i=1}^n \alpha_i y_i K(x_i, x_0) + b \right.$$

Timestamp 2:02

As we saw in the previous chapter that instead of using $x_i^T x_j$, in our dual form we can use any similarity function. kernel functions are one special class of functions which can be used here. We can use the kernel function in the query function also, as we can see from the image above.

→ The most imp. idea in SVM is kernel-trick

soft-SVM-hyperplanes $\approx \log\text{-reg}$
 ↴ margin-max.

by-SVM: - $x_i^T x_j$ $K(x_i, x_j) = x_i^T x_j$

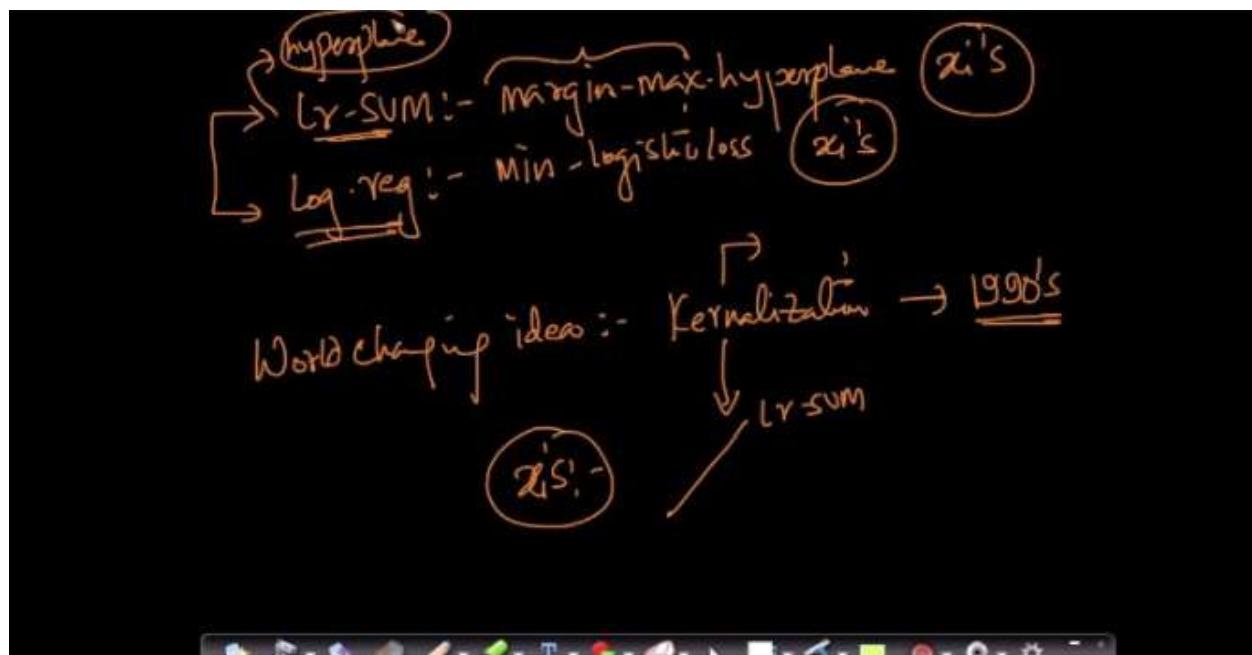
kernel-SVM: - $K(x_i, x_j)$

Timestamp 3:40

This is one of the most important ideas in SVM.

If we keep $x_i^T x_j$, as it is then the SVM is essentially known as linear SVM.

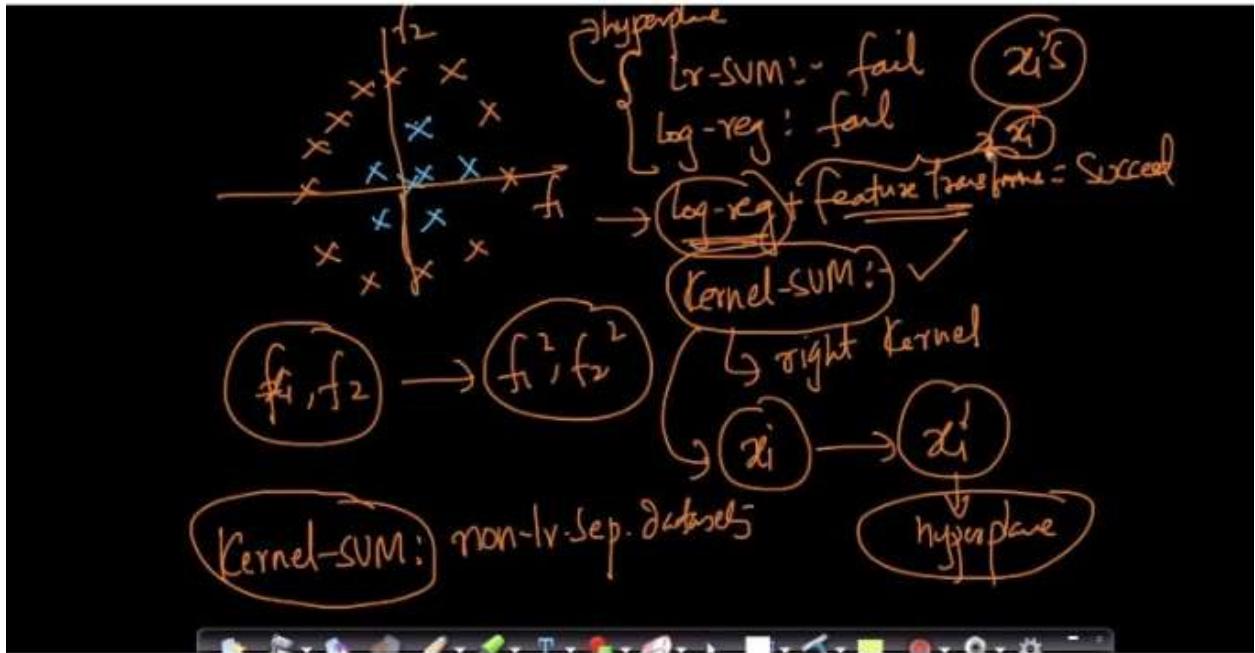
If we apply some kernel function at $x_i^T x_j$ i.e. $K(x_i, x_j)$ then it is essentially known as kernel SVM.



Timestamp 5:40

So, essentially in linear SVM we are trying to find the margin maximizing hyperplane in the space of x_i 's. Also in logistic regression we try to minimize the logistic loss in the space of x_i 's.

So, linear SVM and logistic regression are quite similar and often produce similar results.



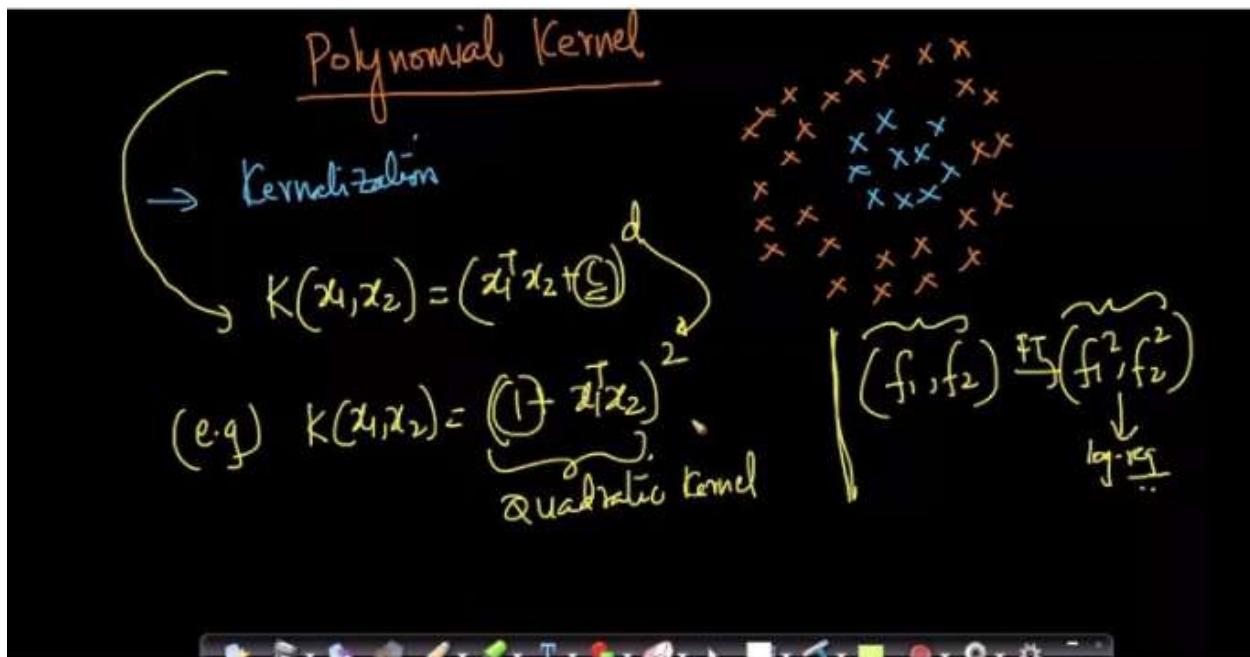
Timestamp 8:39

Consider a dataset as shown in the image above. Now for this dataset no linear hyperplane can work. So algorithm like linear SVM and logistic regression which work in the space of x_i 's will fail in this case. Of course we can try feature engineering and transform these features into different spaces and see if the algorithm works.

So, linear SVM and logistic regression will fail in datasets which are not linearly separable.

Kernel SVM can work in this case, it tries to transform our features into some other higher dimensions such that we may find a hyperplane in this transformed space.

36.7 Polynomial Kernel



Timestamp 2:07

Here we will talk about the polynomial kernel, which is one of the types of kernel.

Consider a dataset as shown in the image above, it is basically two concentric circles. Now this is not linearly separable, so in order to fit a hyperplane we can perform a feature transformation say our features f_1 and f_2 becomes f_1^2 and f_2^2 and then use logistic regression.

Now let's see how Kernelization solves this problem.

Polynomial Kernel :

$$K(x_1, x_2) = (x_1^T x_2 + c)^d$$

where c is a constant and d is the power of the polynomial.

Consider a case of quadratic kernel in this case $c = 1$, $d = 2$

$$\text{So, } K(x_1, x_2) = (x_1^T x_2 + 1)^2$$

$$\begin{aligned}
 K(x_1, x_2) &= (1 + x_1^T x_2)^2 \quad x_1 = \langle x_{11}, x_{12} \rangle \\
 &= (1 + x_{11}x_{21} + x_{12}x_{22})^2 \quad x_2 = \langle x_{21}, x_{22} \rangle \\
 &= \underbrace{1 + x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2}_{\text{Let } [1, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}, \sqrt{2}x_{12}, \sqrt{2}x_{11}x_{12}] : x_1'} + \underbrace{2x_{11}x_{21} + 2x_{12}x_{22} + 2x_{11}x_{12}x_{21}x_{22}}_{\text{Let } [1, x_{21}^2, x_{22}^2, \sqrt{2}x_{21}, \sqrt{2}x_{22}, \sqrt{2}x_{12}x_{22}] : x_2'} \\
 &= (x_1')^T (x_2')
 \end{aligned}$$

Timestamp 5:17

Suppose x_1, x_2 are 2D vectors, where $x_1 = \langle x_{11}, x_{12} \rangle$ and $x_2 = \langle x_{21}, x_{22} \rangle$

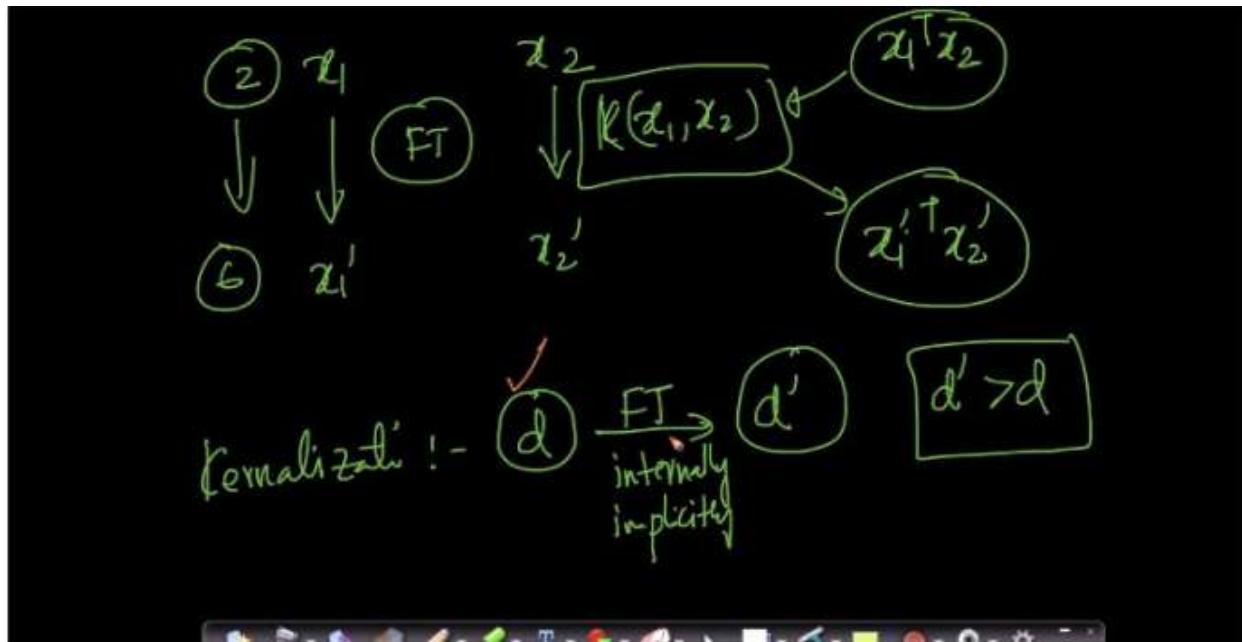
$$\begin{aligned}
 \text{So, } K(x_1, x_2) &= (1 + x_{11}x_{21} + x_{12}x_{22})^2 \\
 &= 1 + x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}x_{21} + 2x_{12}x_{22} + 2x_{11}x_{21}x_{12}x_{22}
 \end{aligned}$$

This above term can be written as product of two vectors say x_1^{\perp} and x_2^{\perp} , where

$$x_1^{\perp} = [1, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}, \sqrt{2}x_{12}, \sqrt{2}x_{11}x_{12}]$$

$$x_2^{\perp} = [1, x_{21}^2, x_{22}^2, \sqrt{2}x_{21}, \sqrt{2}x_{22}, \sqrt{2}x_{21}x_{22}]$$

$$\text{So, } K(x_1, x_2) = (x_1^{\perp})^T (x_2^{\perp})$$

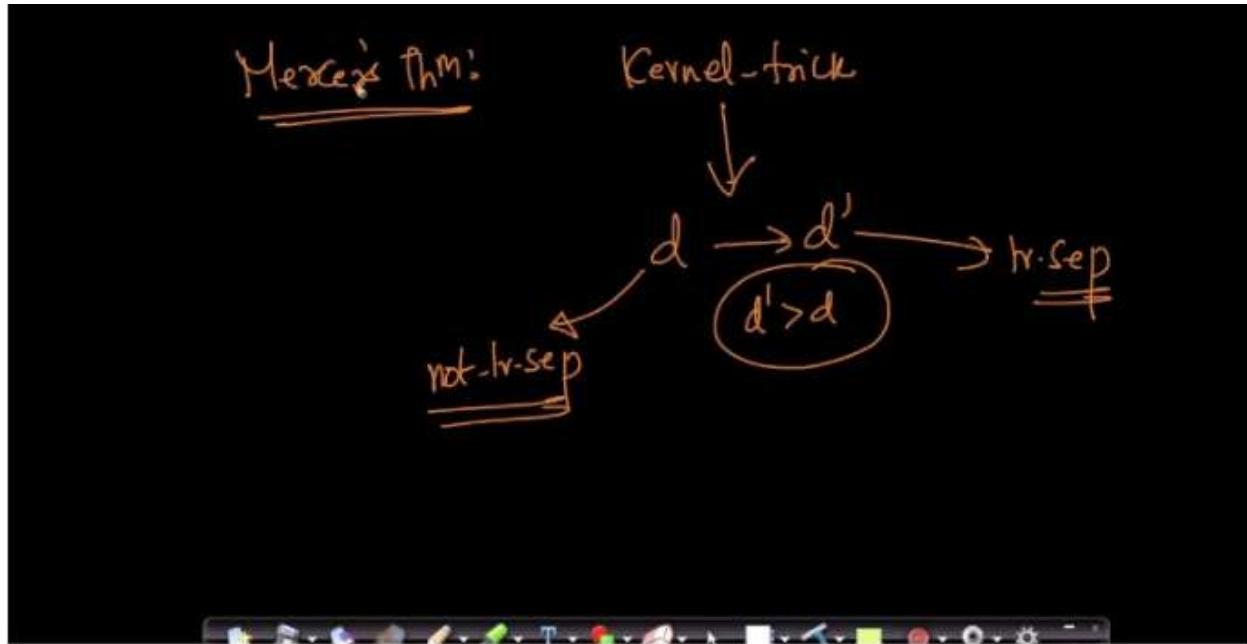


Timestamp 8:00

So essentially what we did was to take our data point which was in 2D and converted it into 6D as x_1' and x_2' contains 6 terms each.

Kernelization transformed our data from d dimensions into d' , where usually $d' > d$. So kernelization is just like feature transformation where we transformed our features. In kernelization we did it implicitly/internally unlike logistic regression where we did it explicitly. This is essentially known as the kernel trick.

In our example we took dot product of transformed features $K(x_1, x_2) = (x_1')^T (x_2')$. These features took our data point into higher dimensions. It contained a squared term similar to feature transformation that we did in logistic regression, so we will be able to find a hyperplane here.



Timestamp 9:00

Mercer's theorem : It essentially says if we are converting our data points using kernelization or kernel trick from d dimension to d' , where $d' > d$, it makes the data points linearly separable which was not the case in lower dimensions.

There is of course a proof behind this, but we won't go into much detail.

Now the problem is to find the right kernel. We saw in our concentric circle dataset that the polynomial kernel worked just fine. But it may not be the case in other datasets.

36.7 RBF-Kernel

Radial Basis Function (RBF)

SUM:- most popular / general-purpose : RBF

$$(x_1, x_2) \quad K_{RBF}(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

↑ hyperparameter

Self-marginal sum

$\|x_1 - x_2\|^2 = d_{12}^2$

RBF Kernel

$\hookrightarrow C$: hyperparameter

Timestamp 1:15

We will now look at one of the most general purpose kernels.

RBF or Radial Basis Function kernel between two data points x_1, x_2 is defined as :

$$K_{RBF}(x_1, x_2) = \exp(-\|x_1 - x_2\|^2 / 2\sigma^2)$$

where $\|x_1 - x_2\|^2$ is the distance between two data points x_1 and x_2 ,

σ is a hyperparameter

$\|x_1 - x_2\|$ can also be written as d_{12} , representing the distance.

$$K(x_1, x_2) = \exp\left(-\frac{d_{12}^2}{2\sigma^2}\right) \quad d_{12} = \|x_1 - x_2\|_2$$

① $\frac{d_{12}}{\sigma} \uparrow ; K(x_1, x_2) \downarrow$
Similitude

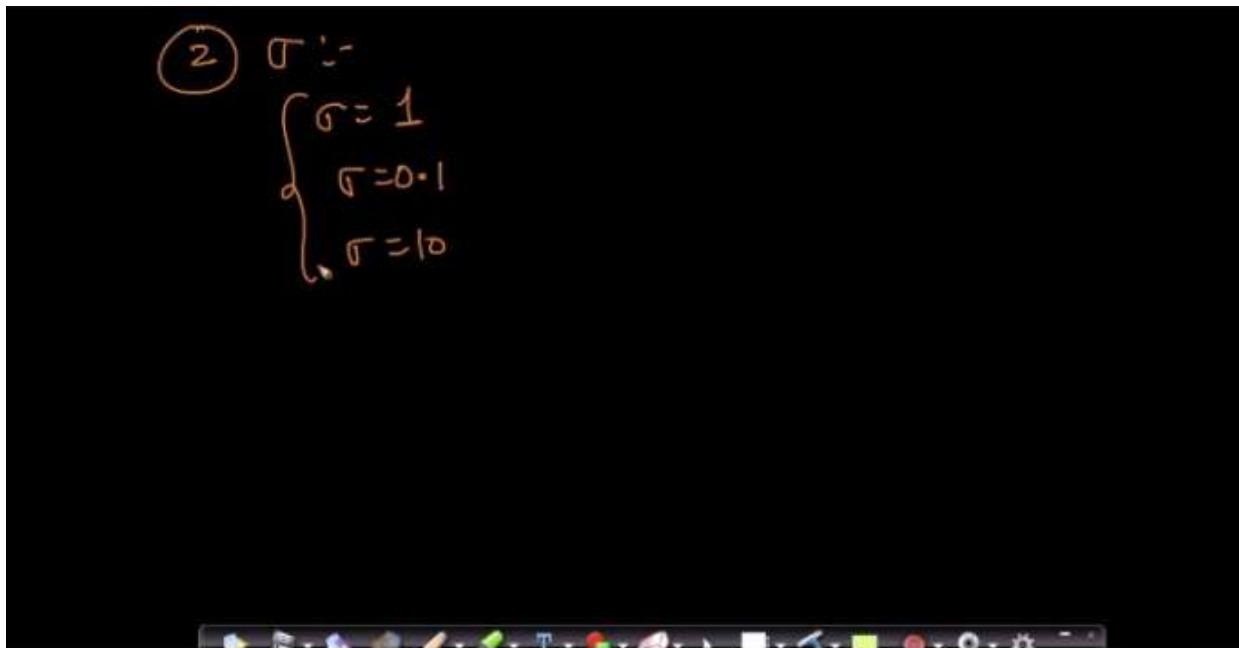
$K(x_1, x_2) > K(x_1, x_3)$

Timestamp 4:10

Now let's try to understand how this function behaves.

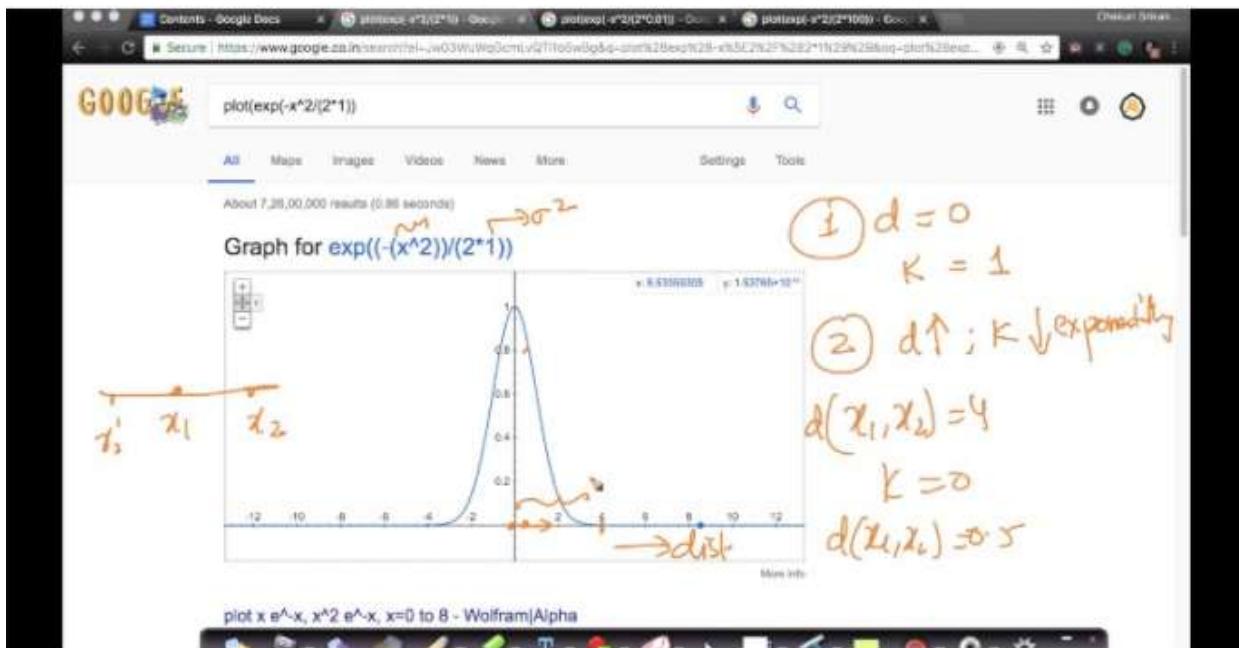
$$K_{RBF}(x_1, x_2) = \exp(-d_{12}^2 / 2\sigma^2)$$

1. If the value of d_{12}^2 increases the value of $K_{RBF}(x_1, x_2)$ decreases as it can be seen from the image above. We know that d_{12} signifies the distance between x_1, x_2 . So, $K_{RBF}(x_1, x_2)$ can be thought of as some sort of similarity where increase in distance is reducing the kernel value.



Timestamp 4:25

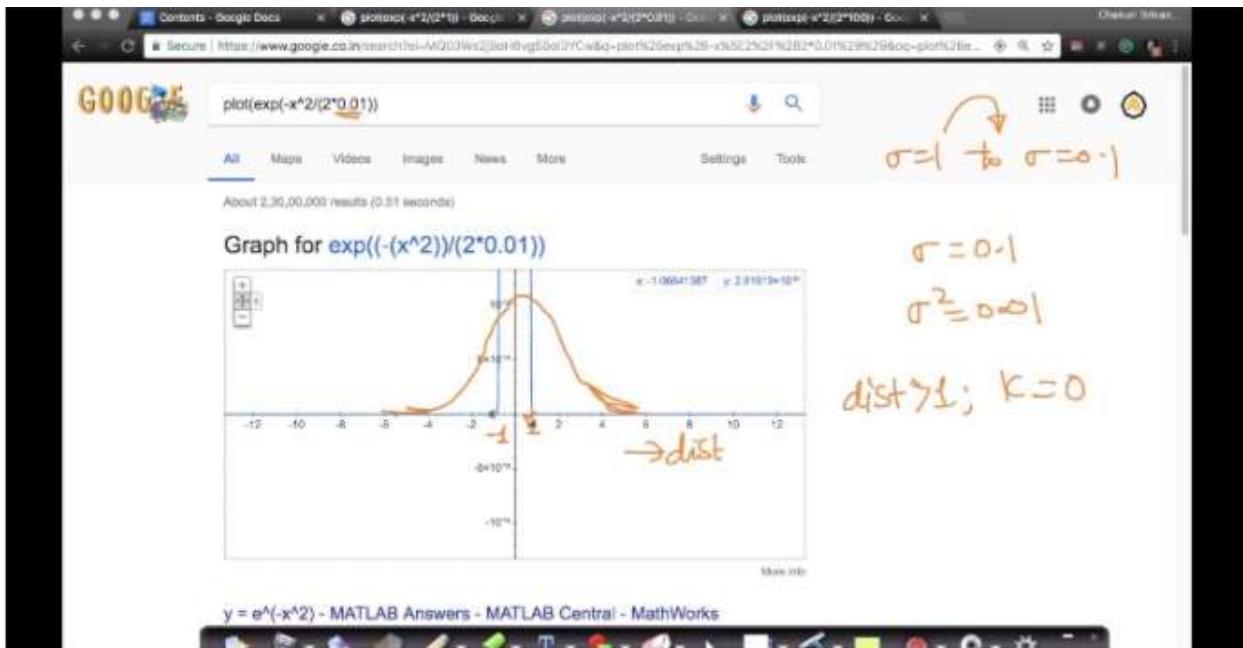
2. Let's try to see the impact of σ , for that we will see plots of $K_{RBF}(x_1, x_2)$ for different values of σ . Say we take 1, 0.1, 10



Timestamp 7:01

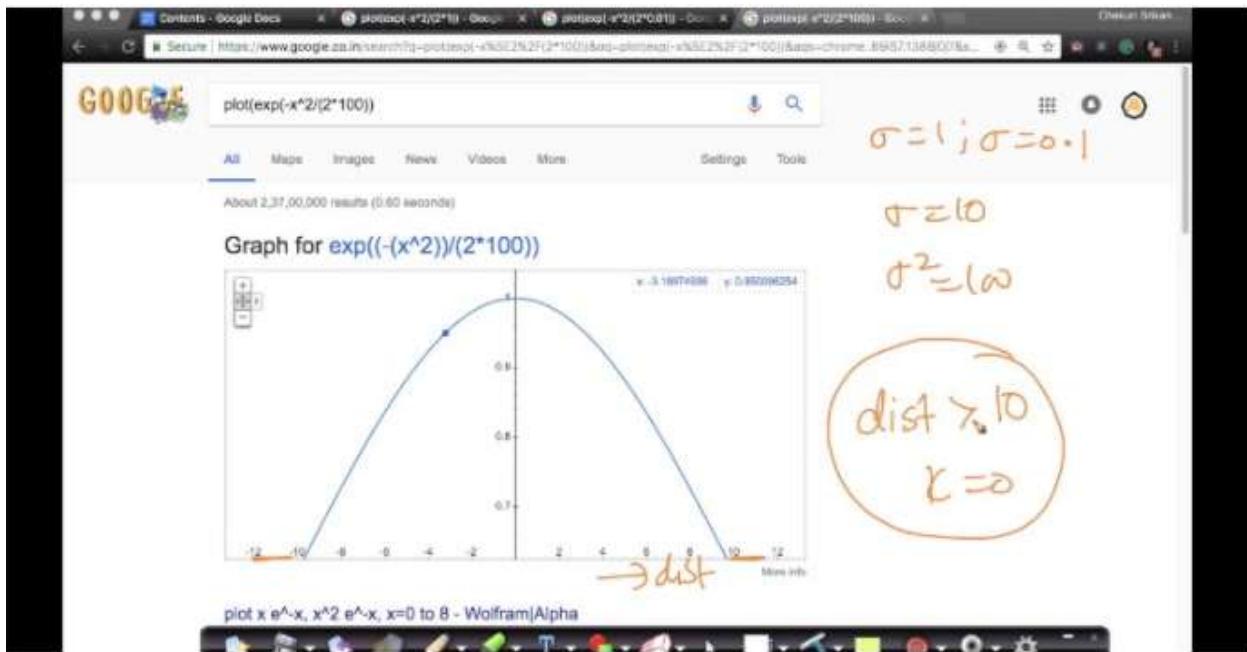
In the above plot we have drawn the RBF kernel with $\sigma = 1$. In the x-axis we represent the distance between two points and in the y-axis we have taken the rbf-kernel. We can see from

this plot that if the distance between two points is 0, the RBF kernel has a maximum value 1. This simply means that points which are at 0 distance away from each other are the most similar. Similarly we can see that as the distance increases, the rbf-kernel values fall exponentially. We can see that around distance 4 the rbf-kernel has a value of 0. Also note that the plot is symmetric since we are dealing with the squared distance. The plot looks like a gaussian curve because the formula is quite similar.



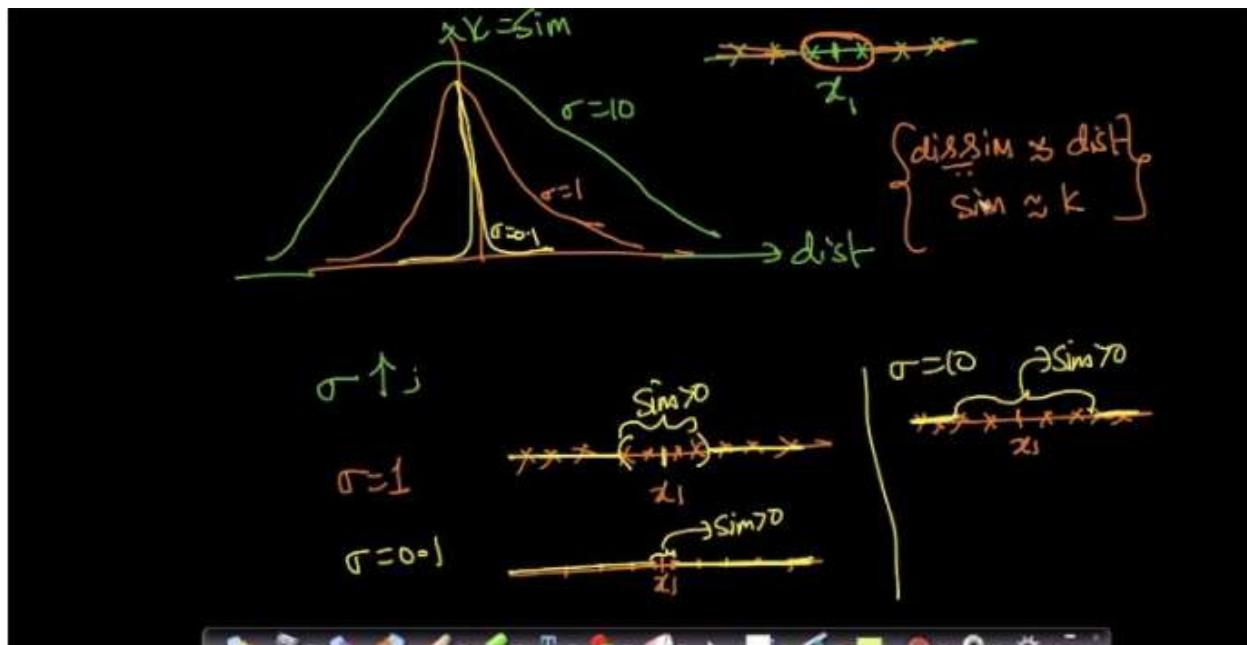
Timestamp 10:40

In the above plot we have drawn the rbf kernel with $\sigma = 0.1$. We can see that the curve has fallen faster than it did when $\sigma = 1$. For distances greater than 1, the rbf kernel has a value 0. The curve is much more peaked than the earlier curve.



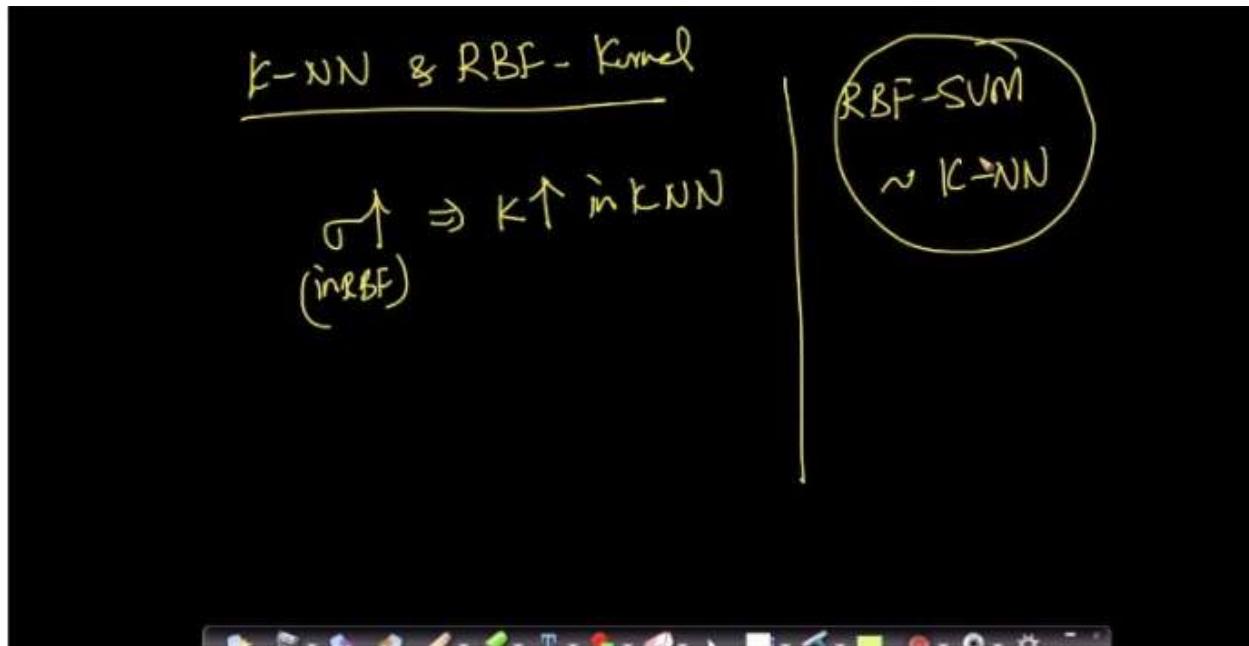
Timestamp 11:33

Now let's increase the value of σ . In the above plot we have drawn the rbf kernel with $\sigma = 10$. We can see that the curve is much wider than the earlier ones. For distances greater than 10, the rbf kernel has a value 0.



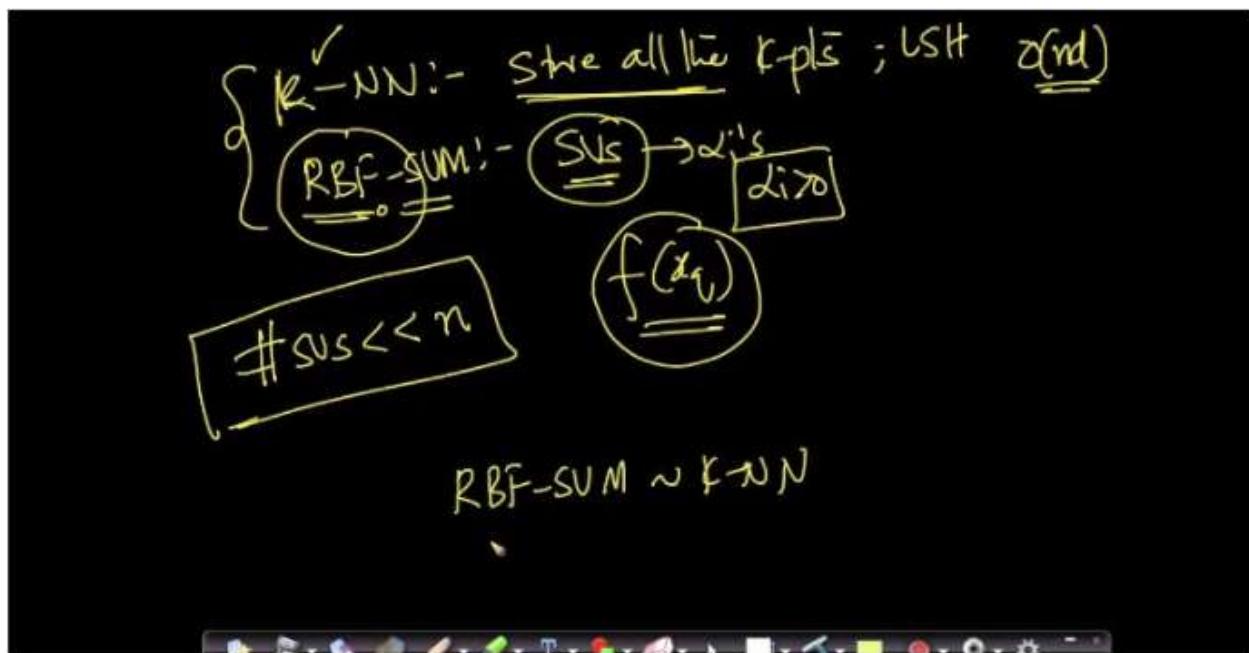
Timestamp 15:20

In the above plot we have drawn all the plots for different σ values. We can see that as the value of σ is increasing we are allowing more points having larger distance to have a similarity value > 0 . We can verify this from the above image.



Timestamp 17:24

There is a striking similarity between the rbf kernel and KNN. As we saw earlier that as the value of σ increased we had more points with a large distance having a similarity value. This is similar to K in KNN where a larger K means we can have more number of neighbours.

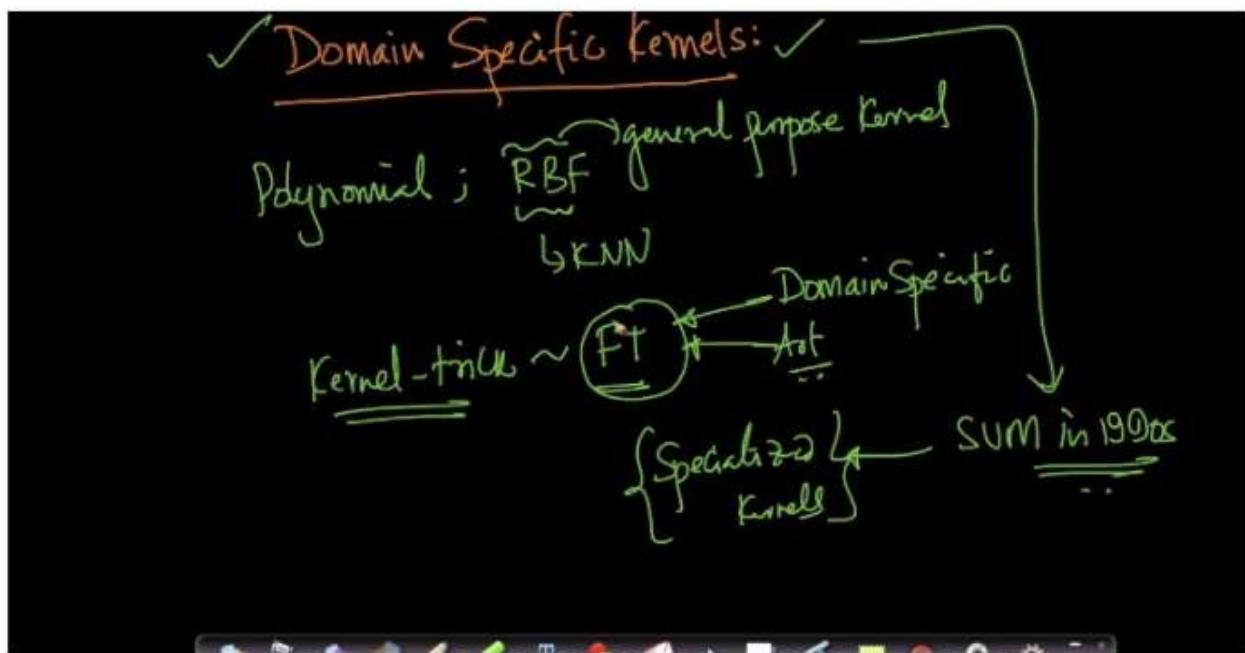


Timestamp 19:22

We know that in the case of KNN at test time we need all the data points in the memory, but for rbf kernels based SVM we only need support vectors. Generally #support vectors <<#datapoints. So the rbf kernel kernel is a nice approximation to KNN. This is the reason why rbf kernels are the best general purpose kernels.

If we don't know what kernel to use we can always go for the rbf kernel. There is however one drawback that now we need to find two hyperparameters i.e. C and σ .

36.8 Domain specific Kernels

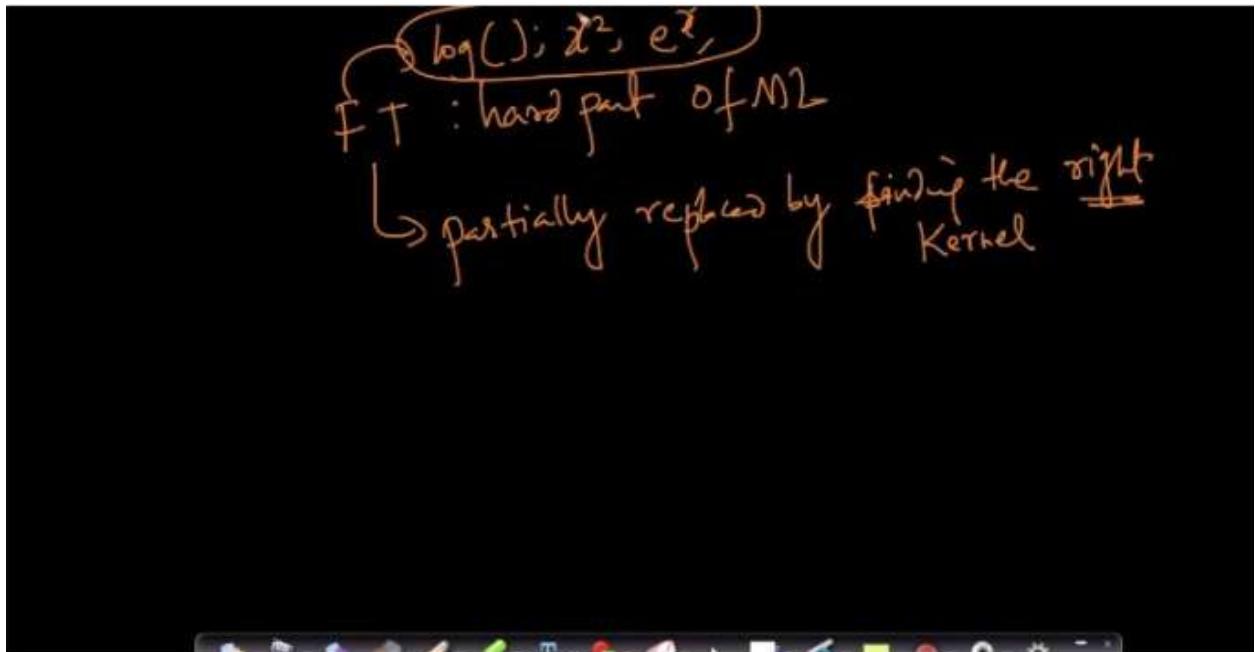


Timestamp 1:25

Till now we have seen a lot of kernels like polynomials, rbf kernels , etc. We have also seen that the kernelization is essentially the same as the feature transformation. Remember feature transformation is very domain specific. It is an art rather than science.

In the 1990's many specialized kernels were invented to suit various domains. Like there are string kernels, genome kernels, graph based kernels .etc.

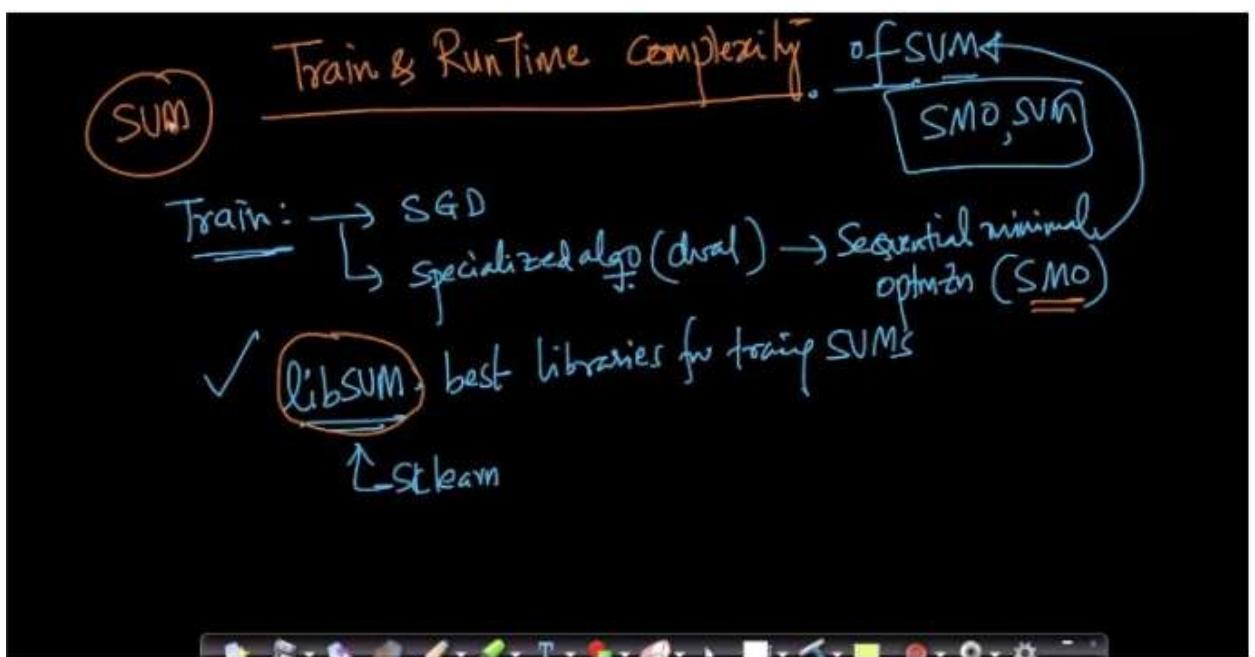
The choice of kernel depends on the type of problem that we are trying to solve.



Timestamp 5:23

So effectively feature transformation is replaced partially by finding the right kernel.

36.10 Train and run time complexities

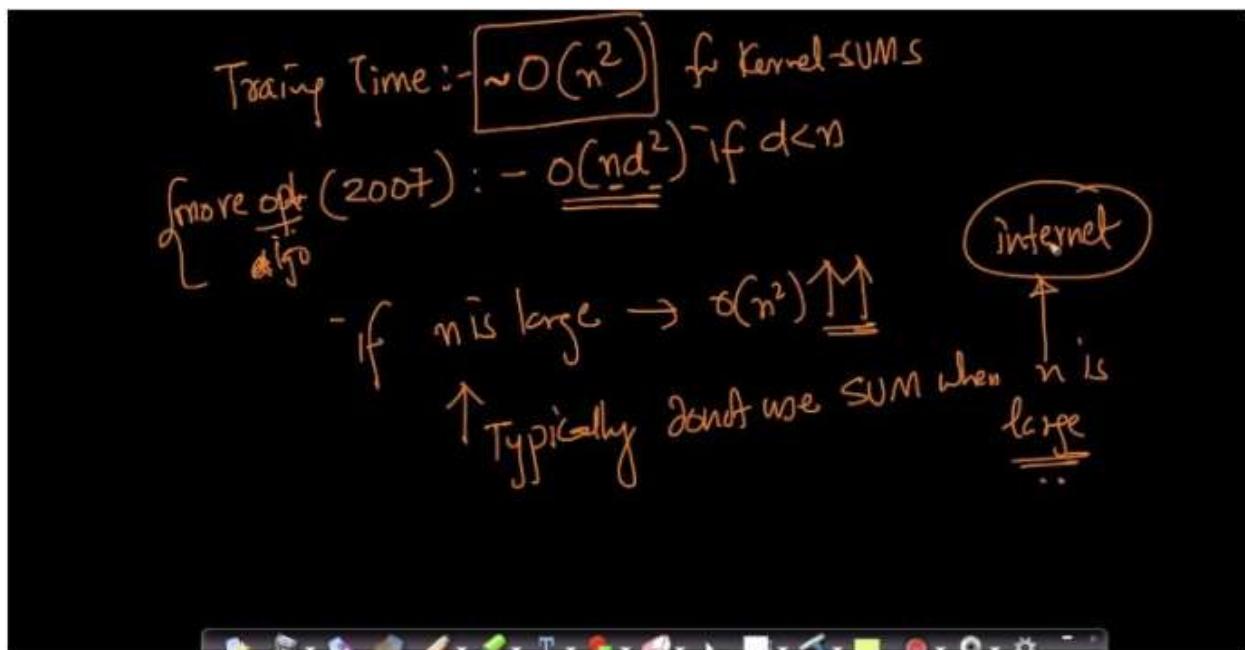


Timestamp 2:10

Here we will look at the time complexity of SVM.

We can train our SVM using gradient descent. However there is a specialized algorithm which solves the dual problem called the SMO or Sequential Minimal Optimization which does a much better job at solving it. We won't go into the details of this.

libSVM is a library which has a very good implementation of SVM. It is very optimized. Sklearn also has an implementation of SVM.

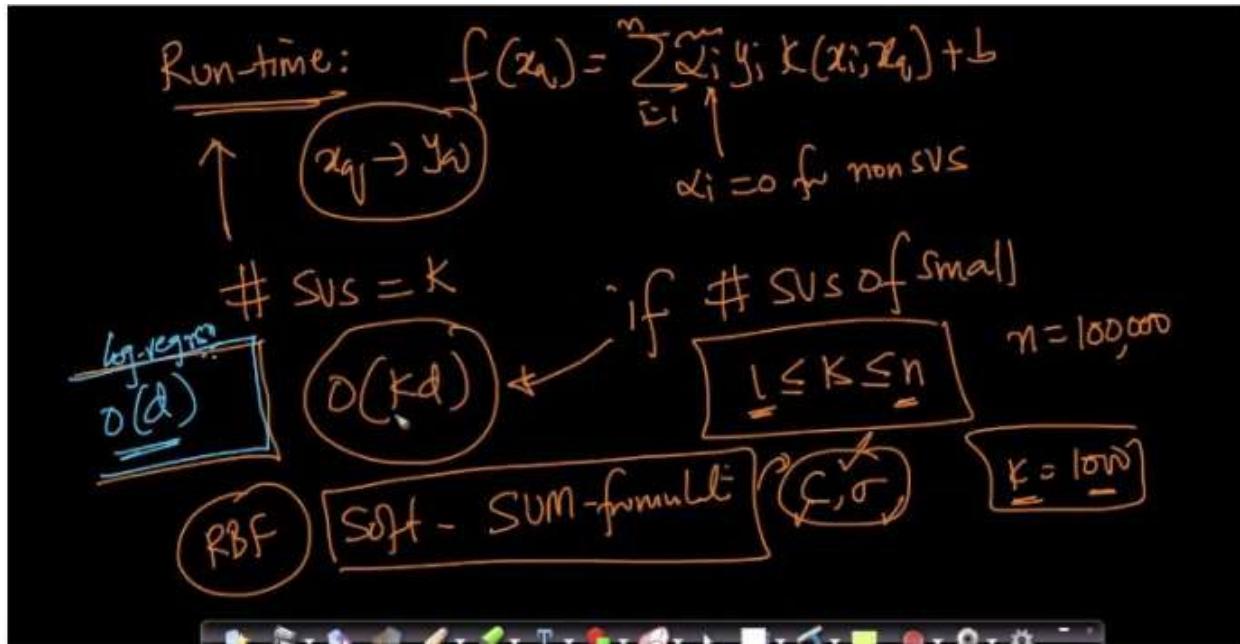


Timestamp 4:23

Train time complexity for kernel SVM is around $O(n^2)$, this is not exact.

So, in places where n is large we typically don't use SVM like internet applications. However this is not mandatory.

In 2007 there has been a more optimized algorithm which has train time complexity of $O(nd^2)$, if $d < n$. d is the number of dimensions.



Timestamp 7:11

The run time complexity for SVM

$$f(x_q) = \sum_{i=1}^n \alpha_i y_i K(x_i, x_q) + b$$

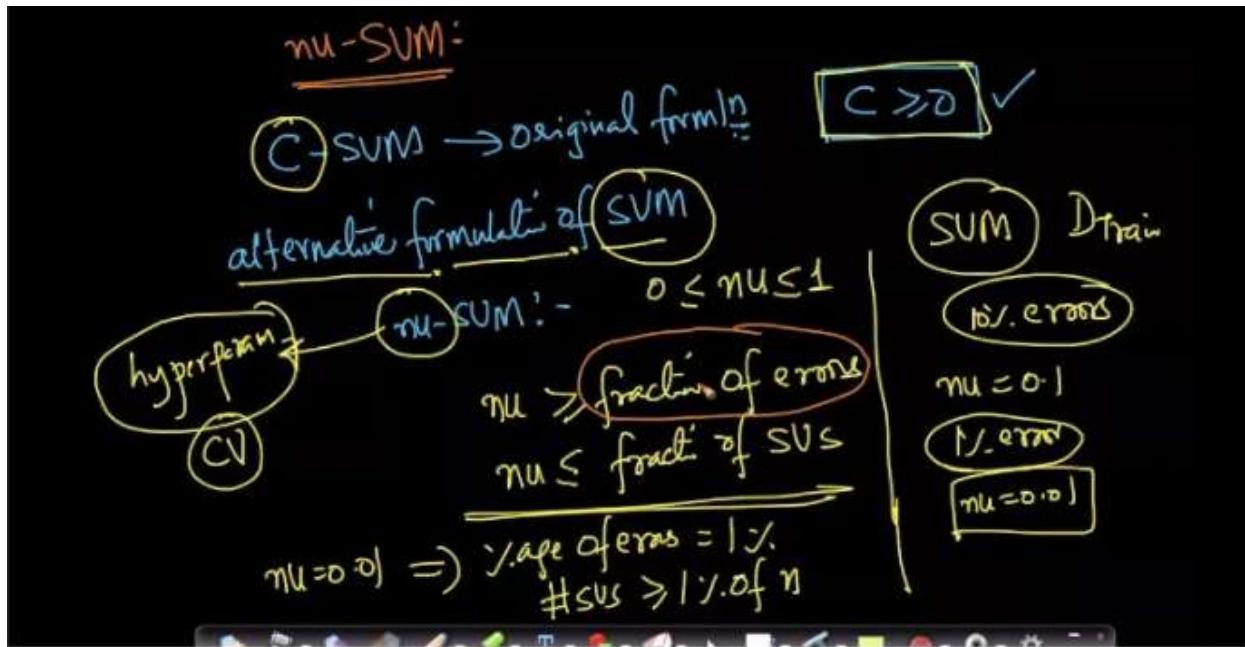
is $O(kd)$, where k is the number of support vectors and d is the dimension of a datapoint.
Remember that α_i 's are 0 for non support vectors.

Now, in Soft Margin SVM we don't have control over the number of support vectors we have. So in cases where we have a large number of support vectors the run time complexity becomes higher. Making it not suitable for low latency applications.

However we will look at some techniques where we try to control the number of support vectors but they are not that commonly used.

In logistic regression the run time complexity was $O(d)$, which typically is lesser than the run time complexity of kernel svm's.

36.11 nu-SVM: control errors and support vectors



Timestamp 3:10

nu-SVM is an alternative formulation of SVM, where nu is the hyperparameter. The original SVM is also called C-SVM because the hyperparameter C. $C \geq 0$.

Now, this nu-SVM has this hyperparameter nu which lies between 0 and 1. $0 \leq nu \leq 1$. This hyperparameter controls the fractions of errors and the fraction of support vectors.

More specifically, $nu \geq$ fraction of errors, and $nu \leq$ fraction of support vectors.

Say we have $nu = 0.01$ then it implies that fraction of errors $\leq 1\%$ of n and number of support vectors $\geq 1\%$ of n.

Run-time complx :- fewer SVs

$$\text{nu} = 0.01 \Rightarrow$$

$$\begin{aligned} \text{errors: } &\leq 1\% \\ \text{SVs: } & > \frac{1}{\nu} - f(n) \end{aligned}$$

$$n = 100,000$$

$$\# \text{SVs} \leftarrow k \geq 100$$

Timestamp 5:30

If we carefully observe the nu-SVM formulation we can see that there is no upper limit to the number of support vectors. There is a limit to errors but not support vectors. So using nu-SVM can give us a fair bit of understanding about the number of support vectors that we can see. But not control over the number of support vectors instead.

Proofing the equivalence between nu-SVM and C-SVM is beyond the scope of this course.

36.12 SVM Regression

\checkmark Support Vector regression (SVR) $y_i \in \mathbb{R}$

SVM - Classfn:- SVC $\rightarrow y_i \in \{-1, +1\}$

Math:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

s.t.

$$y_i - (w^T x_i + b) \leq \epsilon$$

$$(w^T x_i + b) - y_i \leq \epsilon$$

$E > 0$

Timestamp 2:40

SVM's can be used for both classification and regression. If we use it for classification we generally call it Support Vector Classifier where $y_i \in \{-1, +1\}$. If we use it for regression we generally call it Support Vector Regressor where $y_i \in \mathbb{R}$.

The mathematical formulation for SVR is :

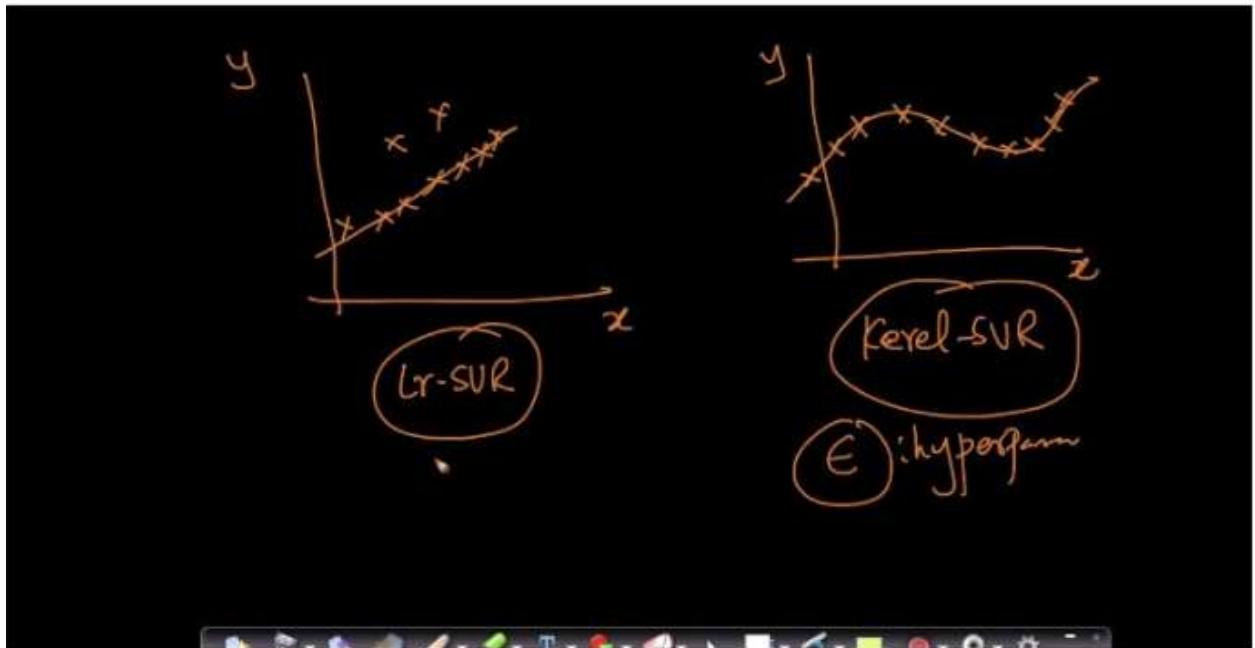
$$(w^*, b^*) = \underset{(w, b)}{\operatorname{argmin}} \frac{1}{2} \|w\|^2,$$

s.t. $\forall i, y_i - (w^T x_i + b) \leq \epsilon$

$$(w^T x_i + b) - y_i \leq \epsilon$$

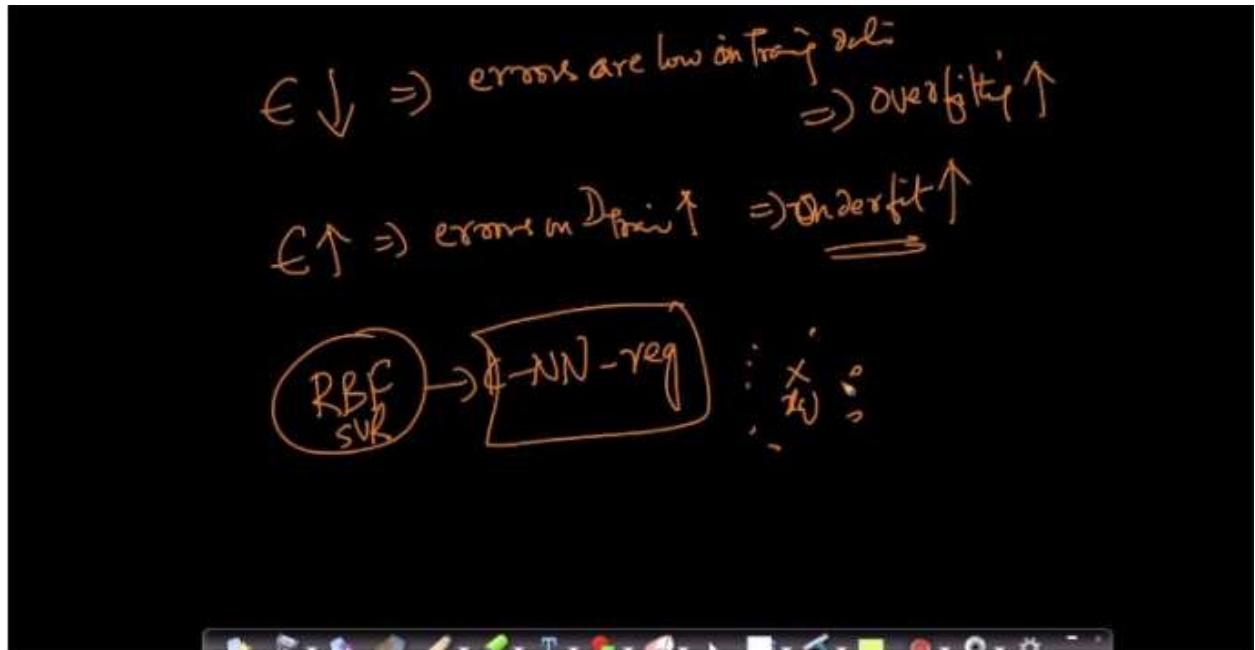
$$\epsilon \geq 0$$

The constraints essentially means that the difference between the predicted value and the actual value should less than or equal to ϵ , as you can see from the image above. Here, ϵ is a hyperparameter. This is essentially the linear formulation of SVR which can also be kernelized.



Timestamp 3:27

As we can see from the image above. Linear SVR can only fit linear hyperplanes. To fit a non-linear hyperplane we need kernelization.

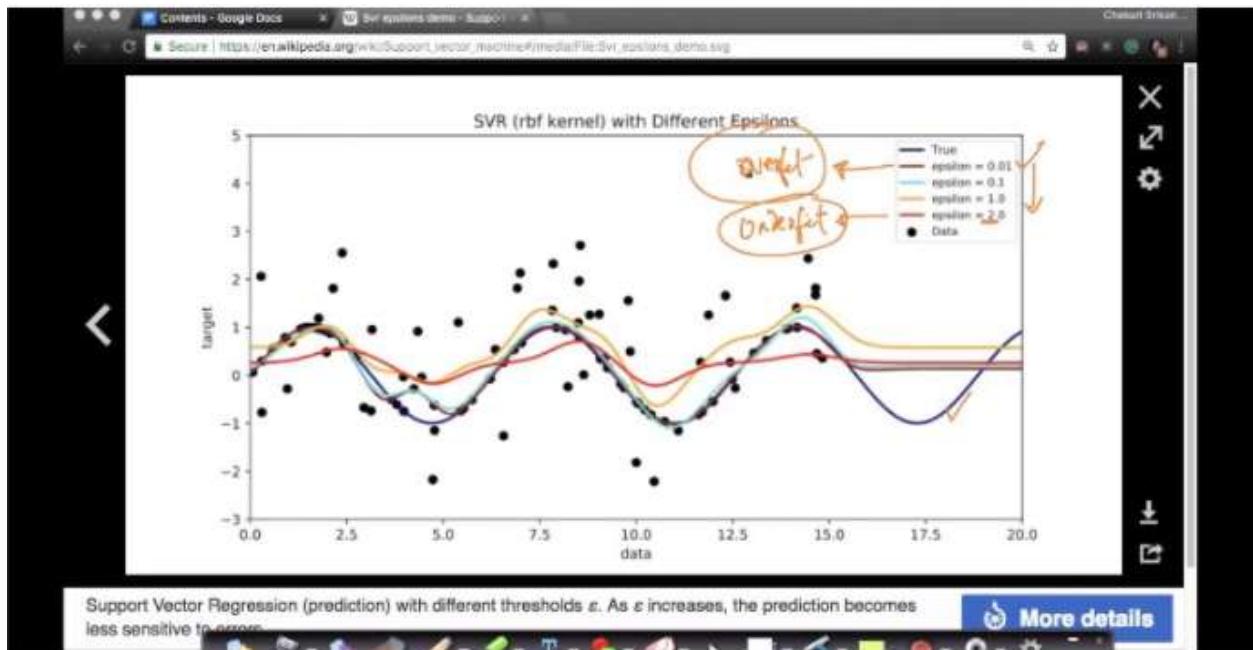


Timestamp 4:30

So we have our hyperparameter ϵ . If the value of ϵ is decreased this effectively means that we want lesser errors on training data. Resulting in overfitting.

Similarly if the value ϵ is increased that means we can tolerate high errors, that means underfitting.

Roughly RBF-SVR behaves similarly to KNN-Regression.



Timestamp 6:12

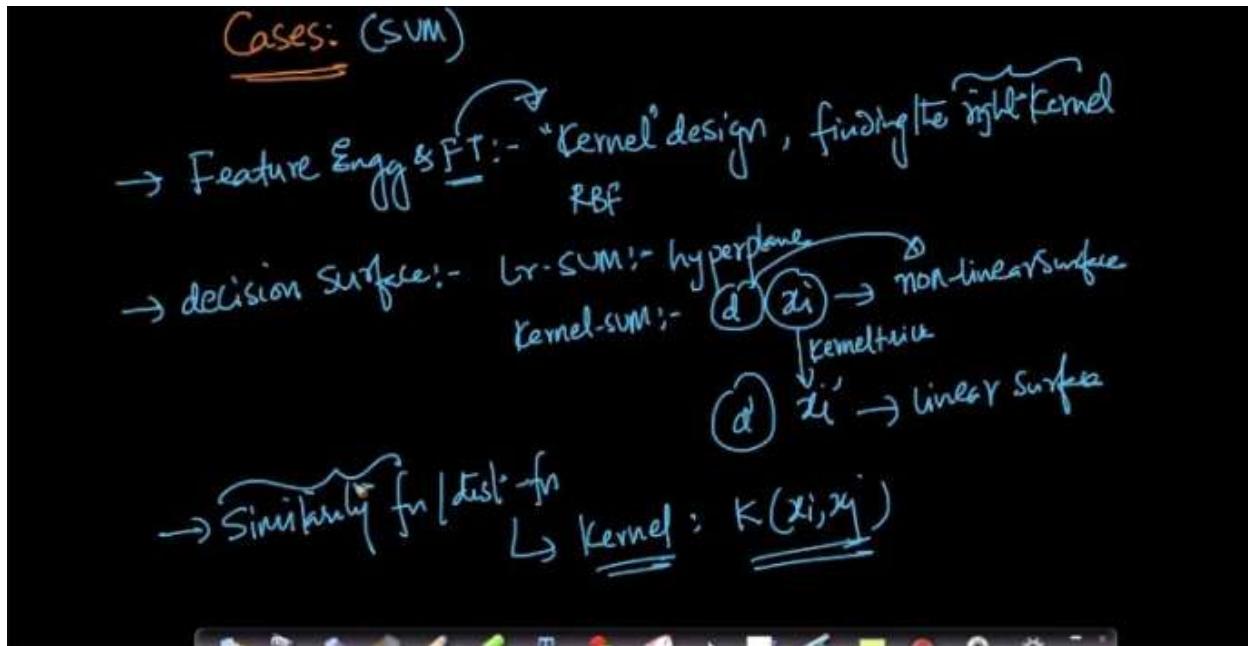
We can see from the image above, how SVR based on rbf kernel behaves for different values of ϵ . For lower values it is overfitting and for higher values it is underfitting.

You can also see these:

Confusion regarding SVR constraints: <https://youtu.be/kgu53eRFERc>

For more mathematical details, check out <https://alex.smola.org/papers/2004/SmoSch04.pdf>

36.13 Cases



Timestamp 2:40

Here we will look at various cases for SVM:

1. Feature Engineering and Feature Transformation: We have seen that we can do this using kernels. The trick is to find the right kernels. Of course in general we can use the rbf-kernel.
2. Decision Surface: If we are using a linear SVM then the decision surface will always be linear. In order to fit data on a non-linearly separable data we can do kernelization. In kernelization we essentially take our data into higher dimensions in the hope that in higher dimensions we can find some hyperplane which can separate our data.
3. Similarity / Distance function: We can very easily use any similarity or distance function in SVM. Kernels are essentially similarity/distance measures.

→ Interpretability & feature importance → kernel-SVM
 ↳ F.I. for various features
 ↳ forward feature Sel.cn

→ Outliers → v. little impact
 ↳ SV's that matter
 ↳ RBF with a small σ → r-NN with small K

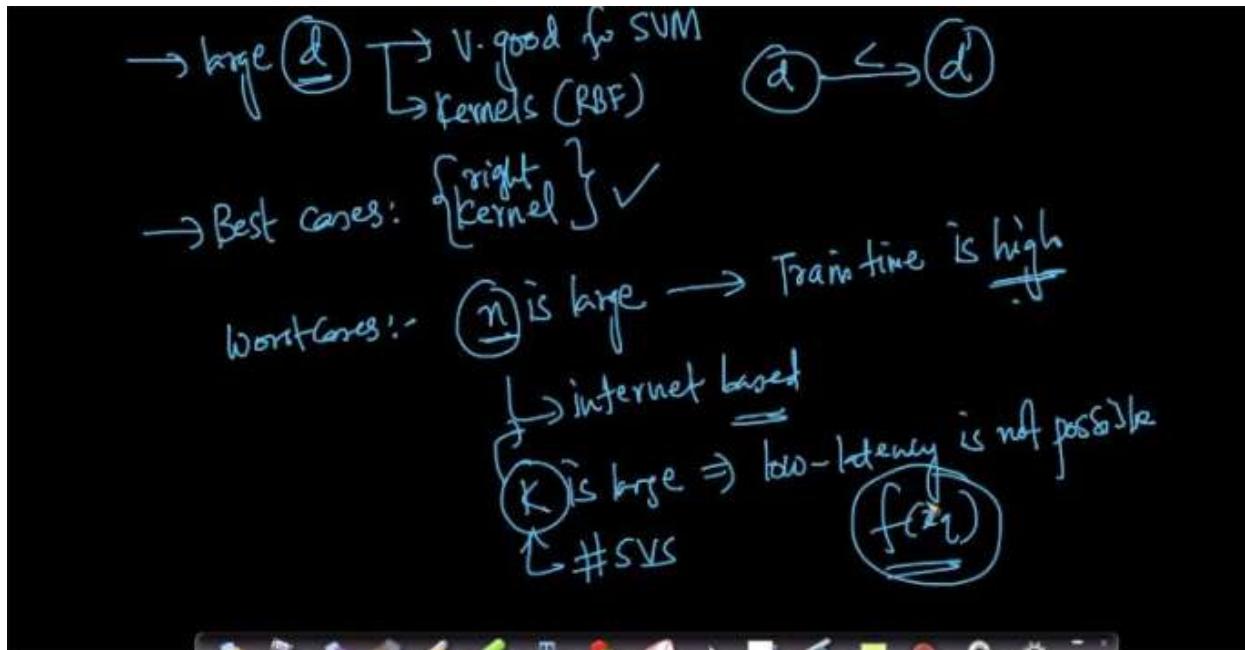
→ Bias-Vari:-
 C ↑ ⇒ overfit ⇒ high Vari
 C ↓ ⇒ underfit ⇒ high bias
 (RBF SUM: - $\sum C_i \Gamma_i^T \Gamma_i$)

Timestamp 6:00

4. Interpretability and Feature Importance: SVM's are not that interpretable especially Kernel SVM's. Also it is difficult to find feature importance directly. We can use techniques like Forward/Backward feature selection but they are time consuming.

5. Outliers: SVM's are not much impacted by outliers, because only support vectors matter. However if we use a rbf kernel with a very low σ , then there can be overfitting. We can think of this as choosing a small k in KNN.

6. Bias - Variance: In SVM's C is our hyperparameter. If the value of C increased we have overfitting i.e. high variance. If its value is decreased then we have underfitting or high bias.



Timestamp 8:27

7. Large d : Large dimensions are very good for SVM as with kernelization we are essentially taking our data points to higher dimensions.

So, we will have the best cases when we have the right kernel.

Worst cases will occur when we have large n , this increases our training time. Large n 's are typically seen in internet applications. Large k i.e. no of support vectors is also an issue, because it increases our test time complexity considerably. Cases where low-latency is the requirement this can be an issue.

Typically when we have a large n we use logistic regression with feature transformation/engineering.

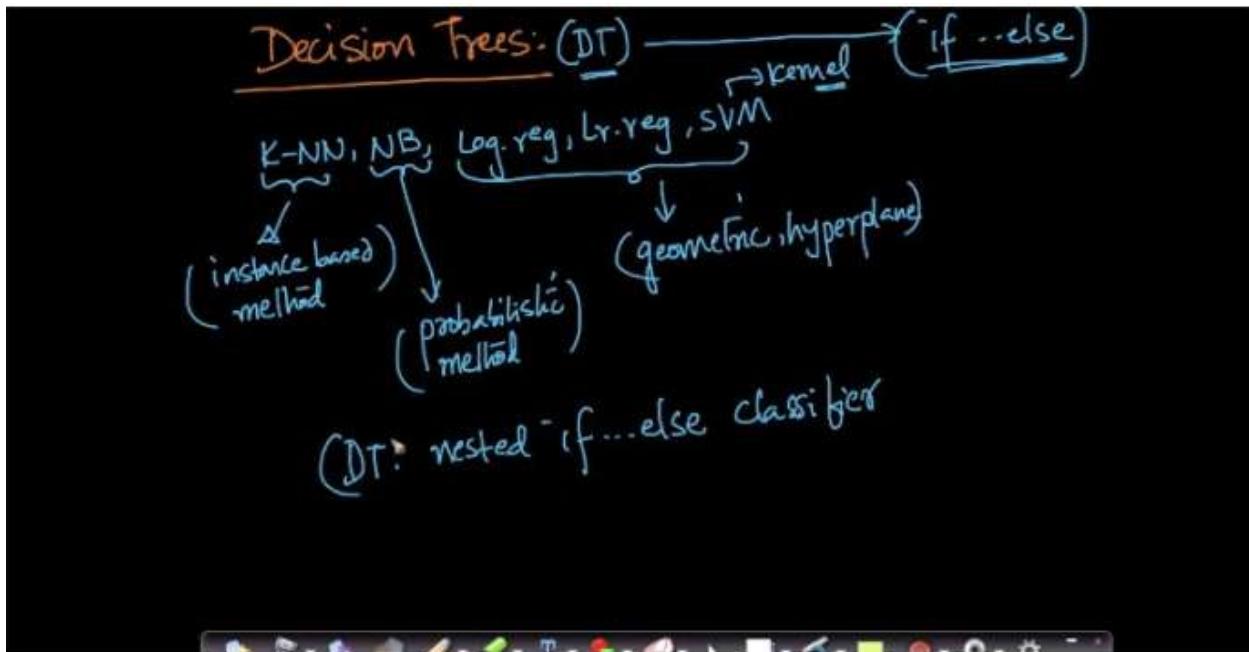
36.14 Code Sample

We can see the scikit learn's implementation of SVM and its variation here =>

<https://scikit-learn.org/stable/modules/svm.html>

In this chapter we will look at an algorithm called Decision Trees

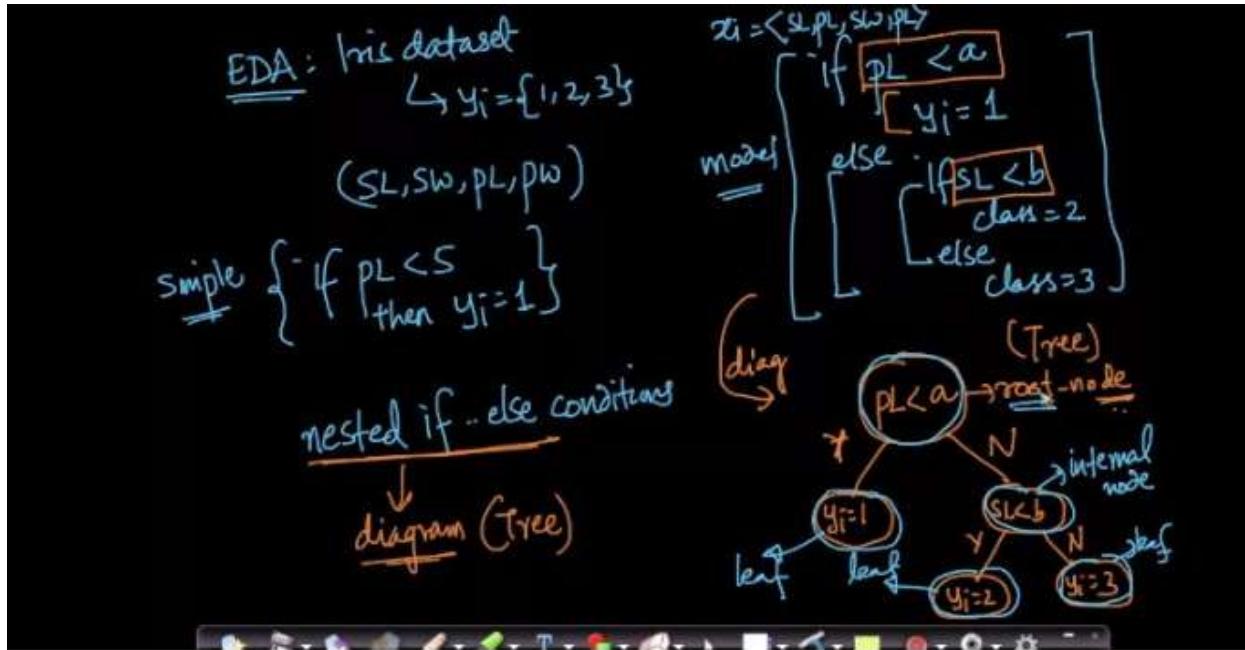
37.1 Geometric Intuition of decision tree: Axis parallel hyperplanes



Timestamp 2:19

Till now we have seen many different types of algorithms based on different ideas. Like KNN is an instance based algorithm, it is mostly heuristics. Naive Bayes is a probabilistic method. Algorithms like logistic regression, linear regression, SVM are mostly geometric in nature.

In this chapter we will look at another type of algorithm called Decision Trees which behaves like a nested if else classifier.



Timestamp 8:02

Let's take a look at the iris dataset that we saw during EDA. In the iris dataset we have $y_i \in \{1, 2, 3\}$. It contains 4 features: Sepal Length(SL), Sepal Width(SW), Petal Length(PL) and Petal Width(PW).

Now for this dataset we can create a very simple rule, say if the PL is less than 5 then $y_i = 1$.

We can write this in the if else format as

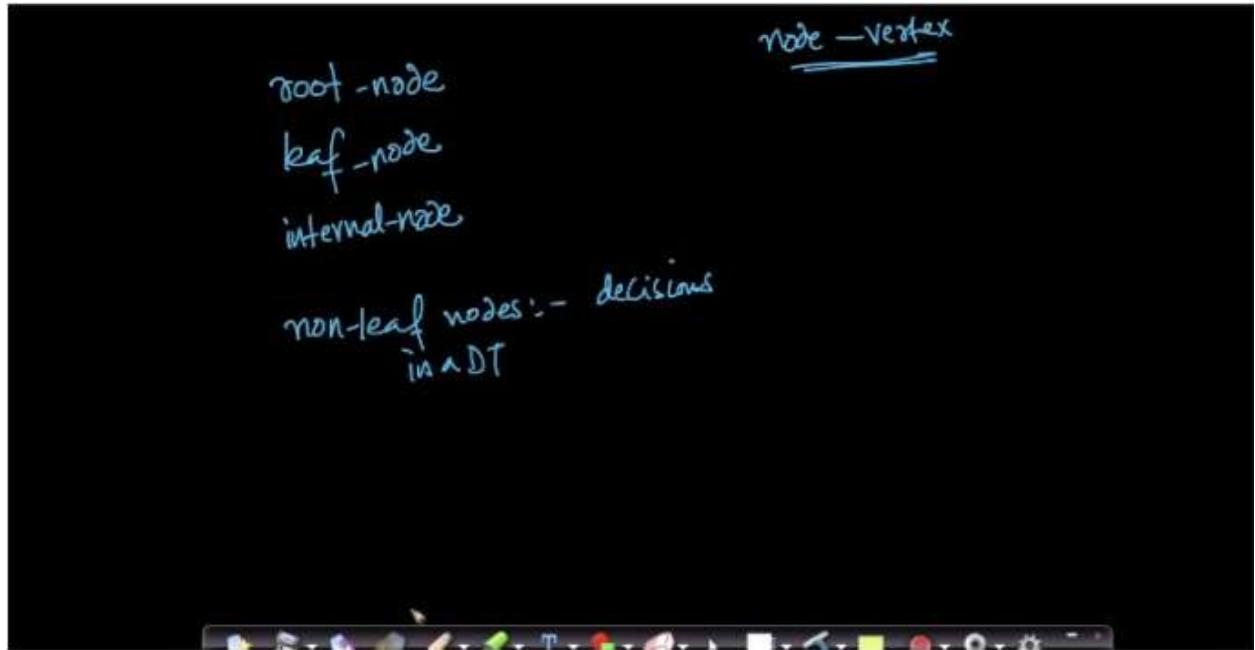
if PL < 5:

then $y_i = 1$

This type of rule is called a if else rule.

Based on our dataset we can create a nested rules of if else as shown in the image above. This nested if else rules can also be effectively shown using a data structure called trees.

We can see in the image above how we have converted our nested if else rules into a tree. At each node of the tree we are essentially checking a condition similar to the if checks and based on the result of this checking we are deciding which path to take.



Timestamp 8:40

Let's briefly discuss trees.

A node is a structure which may contain a value or condition. Each node in a tree has zero or more child nodes. Nodes are often called vertices.

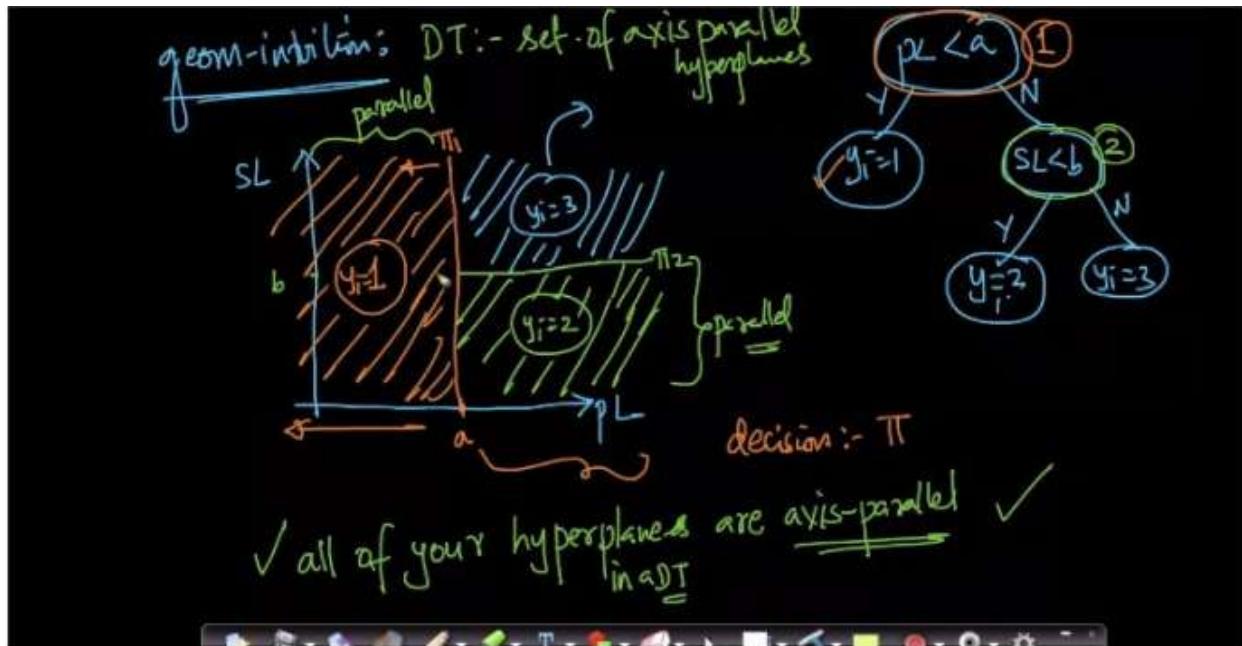
A root node is the first node in a tree.

Leaf nodes are the last nodes of a tree.

Any node which is not a leaf node is called an internal node.

In Decision Trees all the decisions are taken in non-leaf nodes.

Decision Trees are highly interpretable because we can read the conditions in simple plain english.



Timestamp 15:03

Let's now look at the geometric interpretation of Decision Trees.

Consider the same Iris Dataset that we saw earlier but with only two features, say Petal Length(PL) and Sepal Length(SL). Now we can make a decision tree using these features as shown in the image above.

Let's draw these features i.e., PL and SL on x - axis and y-axis respectively.

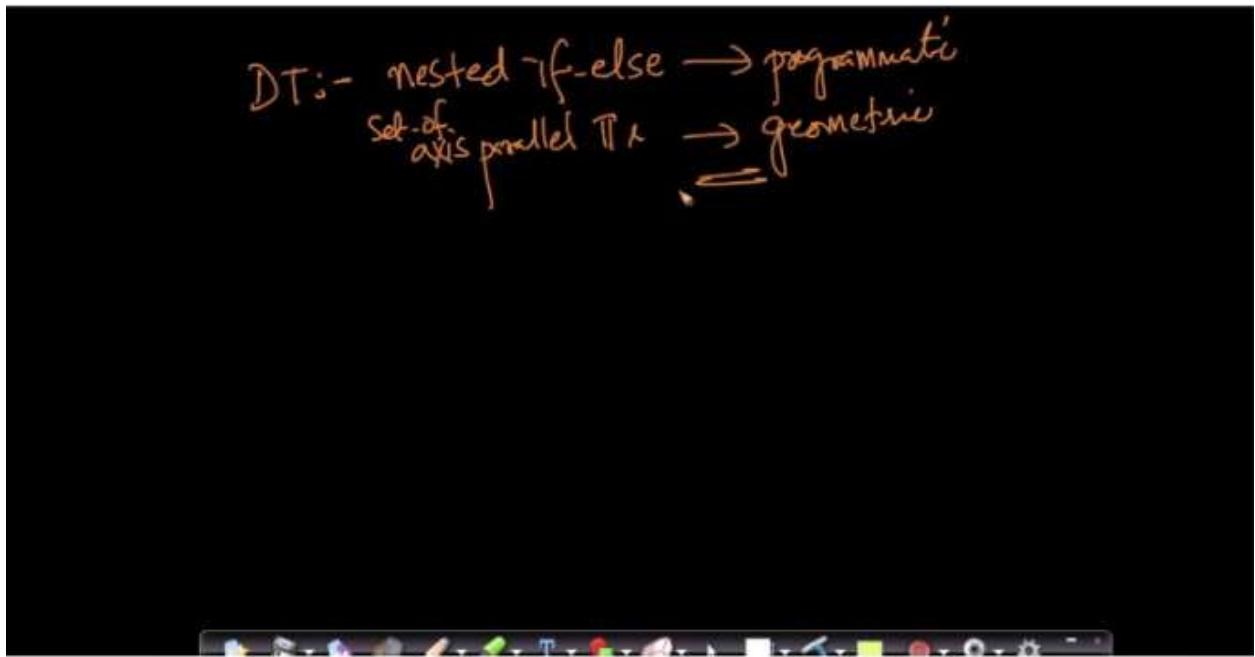
Now let's look at the root node of the decision tree where it says that if $PL < a$ (some threshold) then the class label $y_i = 1$, otherwise we need to make another decision.

We can draw a line on PL at a and say that if the value of PL is less than a then the class label is 1.

Now say for a data point its PL value was greater than a , then we will move to the second condition marked 2 in decision tree, it says that if $SL < b$ (some threshold) then the class label $y_i = 2$, otherwise $y_i = 3$.

For this, first we will go right of the threshold i.e. $PL = a$, and in this region we will draw a line on SL at b and say that if the value of SL is less than b then the class label is 2 , otherwise the class label is 3. Note that this line didn't pass through the $PL = a$, because we had a value of $PL > a$.

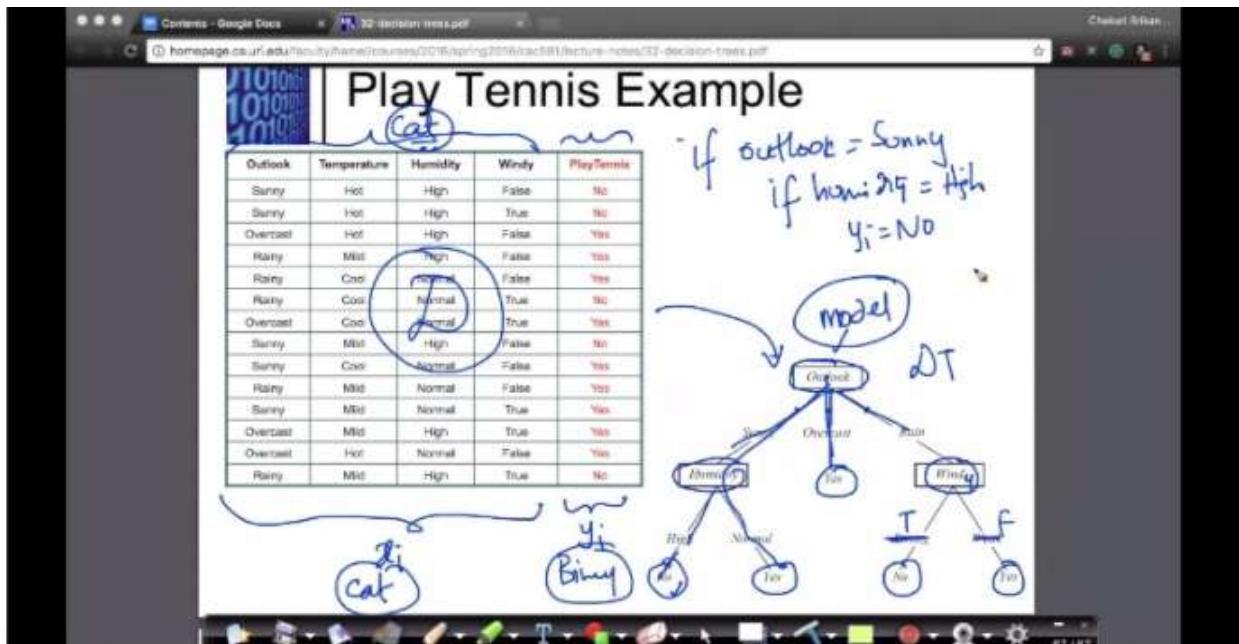
In this case we have drawn lines, but in higher dimensions we would have drawn hyperplanes. Also notice that essentially we are dividing up our space into various regions. Each region indicates different classes. These regions can be hyper cubes or hyper cuboids. All the hyperplanes that we are drawing are essentially parallel to the axes. So it can be said that geometrically Decision Trees are a set of axes parallel hyperplanes.



Timestamp 16:34

So to conclude we can say that programmatically we can think of Decision trees as nested if-else classifiers and geometrically Decision trees are a set of axis parallel hyperplanes.

37.2 Sample Decision Tree



Timestamp 5:30

Here we have taken a small dataset and built a decision tree using that. The dataset can be accessed from here => <https://homepage.cs.uri.edu/faculty/hamel/courses/2014/spring2014/csc581/lecture-notes/31-decision-trees.pdf>

It's a binary classification problem with target variable play, $y_i \in \{\text{Yes(Y)}, \text{No(N)}\}$. Our dataset contains 4 features => Outlook, Temperature, Humidity, Wind. All these features are categorical features.

Outlook can have 3 categories: sunny, overcast, rain. Temperature can have 3 categories: hot, mild and cool. Humidity can have 2 categories: high, normal. Wind can have 2 categories: strong, weak.

Now the decision tree built can be seen on the image above. At the root node we have the feature outlook, since it contains three features it has three paths one each for sunny, overcast and rain. Let's see the leftmost path (a path is essentially a set of connected nodes), it says that if the outlook is sunny and then if the humidity is high we cannot play tennis.

The screenshot shows a Google Doc with the title "Play Tennis Example". On the left, there is a table of data points:

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	No
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	No
Rainy	Mild	High	True	No

On the right, there is a handwritten note: $x_q = [\checkmark \text{sunny}, \text{hot}, \checkmark \text{high}, \text{X}]$ and $y_q = \text{No}$.

Below the table is a decision tree diagram:

```

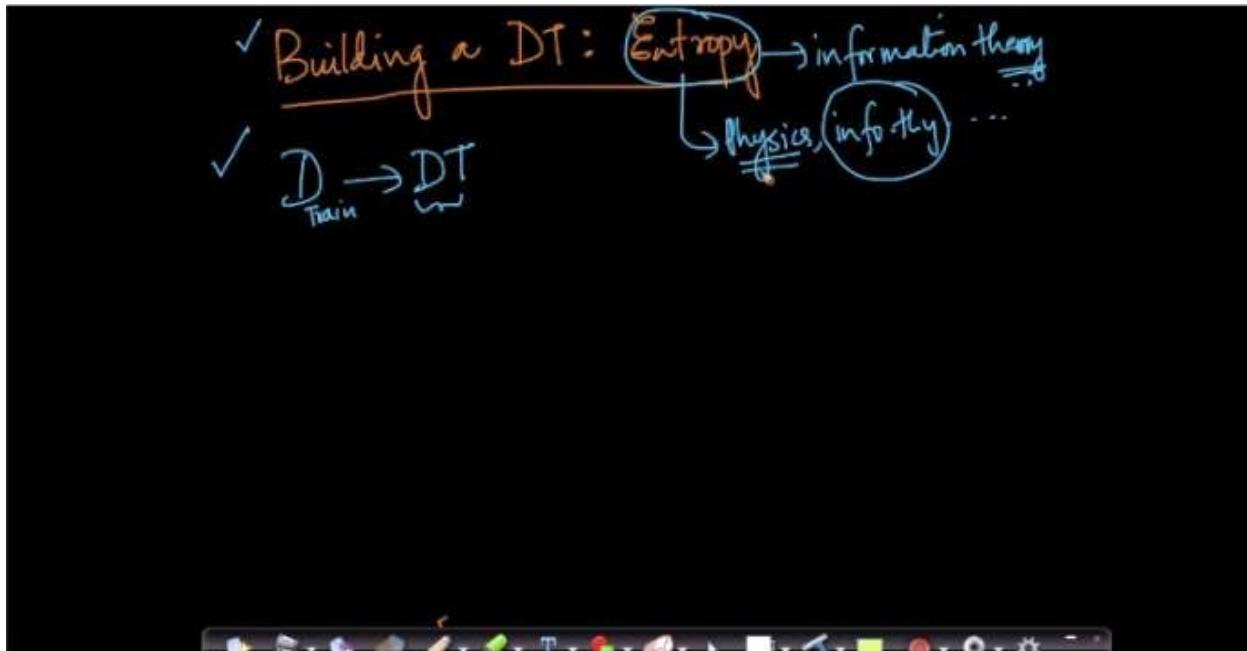
graph TD
    Root[Outlook] -- Sunny --> Humidity[Humidity]
    Root -- Overcast --> Rain[Rain]
    Root -- Rain --> Wind[Wind]
    Humidity -- High --> No((No))
    Humidity -- Normal --> Yes((Yes))
    Wind -- Strong --> No
    Wind -- Weak --> Yes
  
```

Timestamp 6:32

Now during test time suppose we get a query point $x_q = [\text{sunny}, \text{hot}, \text{high}, \text{strong}]$. We will take this query point and pass it over the decision tree. At the root node we have the feature outlook, our data is sunny so we take the left path, then the feature is humidity, our data is high so we take the left path, reaching a leaf node. The value of the leaf node is No, so we cannot play tennis.

So now we know how to predict a data point if we know the decision tree. The challenge is to build a decision tree.

37.2 Building a decision Tree: Entropy



Timestamp 1:17

Here we will learn how to build a decision tree. Before doing that we need to learn about a few concepts.

Entropy is used in various disciplines of science, like physics, information theory, electronics,etc. We will focus more on the information theory's interpretation of entropy.

$\gamma \rightarrow y_1, y_2, y_3, \dots, y_k$

$$H(Y) = - \sum_{i=1}^k P(y_i) \log_b(P(y_i))$$

entropy

$$P(y_i) =$$

$$\underline{P(y_i) = P(Y=y_i)}$$

$$\begin{cases} b=2 \\ n \\ b=e=2.718 \end{cases}$$

$$\begin{cases} \log_2 = \lg \\ \log_e = \ln \end{cases}$$

Timestamp 3:29

Let Y be a random variable which can take k values $\Rightarrow \{y_1, y_2, y_3, \dots, y_k\}$. Entropy which is usually represented by H for this random variable will be,

$$H(Y) = - \sum_{i=1}^k P(y_i) \times \log_b(P(y_i))$$

Here, b is the base which we usually take as 2 or e (euler no) = 2.718. Typically we use base 2. Base 2 logarithm is typically written as \lg , whereas for base e we write it as \ln .

$P(y_i)$ represents the probability of the event where the random variable y_i occurs.

Contents - Google Docs

Play Tennis Example

Y

Outlook	Temperature	Humidity	Windy	Play Tennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overscast	Hot	High	False	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	No
Overscast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	No
Rainy	Mild	Normal	False	No
Sunny	Mild	Normal	True	Yes
Overscast	Mild	High	True	Yes
Overscast	Hot	Normal	False	No
Rainy	Mild	High	True	No

$\text{P}(Y_+) = \frac{9}{14}$

$\text{P}(Y_-) = 1 - \text{P}(Y_+) = \frac{5}{14}$

```

graph TD
    Outlook[Outlook] --> Sunny[Sunny]
    Outlook --> Overcast[Overcast]
    Outlook --> Rainy[Rainy]
    Sunny --> HighH[High]
    Sunny --> NormalH[Normal]
    Sunny --> NoW[No]
    Sunny --> YesW[Yes]
    Overcast --> YesO[Yes]
    Rainy --> YesR[Yes]
  
```

Timestamp 4:34

In our tennis dataset the target variable => Play Tennis(Y) can take two values i.e. {Yes, No}. In this dataset we have a total of 14 rows out of which 9 are Yes and 5 are No.

So, $\text{P}(Y_+) = 9/14$ [probability of yes]

and, $\text{P}(Y_-) = 1 - \text{P}(Y_+) = 5/14$ [probability of no]

$$H(Y) = - \sum_{i=1}^k p(y_i) \log_2(p(y_i))$$

$$H(Y) = - \left(\frac{9}{14} \log_2 \left(\frac{9}{14} \right) + \frac{5}{14} \log_2 \left(\frac{5}{14} \right) \right) = 0.94$$

$\frac{\# \text{ +ve } p/s}{\text{Total } \# \text{ pts}} = \frac{1}{n} \text{ age of +ve p/s in D}$

$\% \text{ age of -ve pts}$

Timestamp 6:27

So, the entropy for the random variable(Y) :

$$\begin{aligned} H(Y) &= - \sum_{i=1}^2 P(y_i) \times \log_2(P(y_i)) \\ &= -9/14 * \lg(9/14) - 5/14 * \lg(5/14) \\ &= 0.94 \end{aligned}$$

9/14 represents the ratio of positive points in our dataset($P(Y_+)$).

5/14 represents the ratio of negative points in our dataset($P(Y_-)$).

Properties: $Y \rightarrow Y_+, Y_-$ (2 class, 2 category)

Case 1: $D \xrightarrow{y_+ \rightarrow 99\%} \quad \left. \begin{array}{l} y_- \rightarrow 1\% \end{array} \right\} H(Y) = -0.99 \lg 0.99 - 0.01 \lg 0.01 = 0.0801$

Case 2: $D \xrightarrow{y_+ \rightarrow 50\%} \quad \left. \begin{array}{l} y_- \rightarrow 50\% \end{array} \right\} H(Y) = -0.5 \lg 0.5 - 0.5 \lg 0.5 = 1$

Case 3: $D \xrightarrow{y_+ \rightarrow 0\%} \quad \left. \begin{array}{l} y_- \rightarrow 100\% \end{array} \right\} H(Y) = 0$

Timestamp 8:32

Now we will look at some of the important properties of entropy, we will see it in the 2 class case say $Y \rightarrow (Y_+, Y_-)$, however these will hold true in multiclass setting as well:

Let's see how the value of entropy changes with changes in the distribution of data.

case 1: Say in our dataset D , $Y_+ \rightarrow 99\%$ and $Y_- \rightarrow 1\%$. In this case entropy

$$H(Y) = -0.99 * \lg 0.99 - 0.01 * \lg 0.01 = 0.0801$$

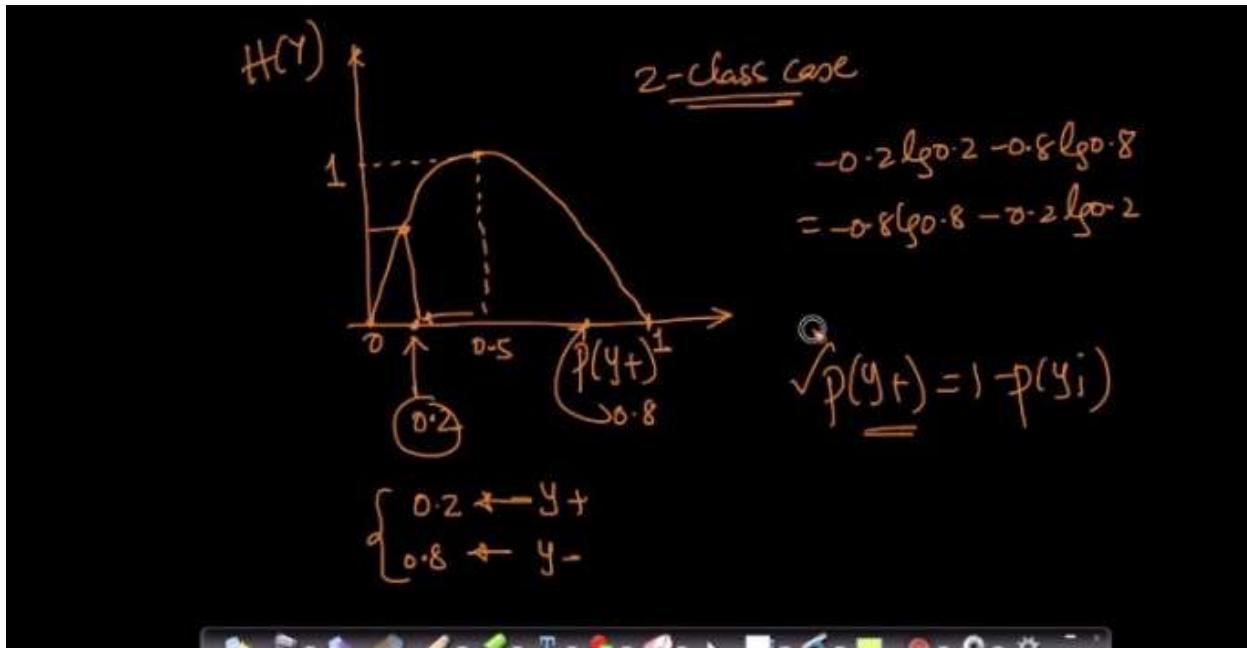
case 2: Say in our dataset D , $Y_+ \rightarrow 50\%$ and $Y_- \rightarrow 50\%$. In this case entropy

$$H(Y) = -0.5 * \lg 0.5 - 0.5 * \lg 0.5 = 1$$

case 3: Say in our dataset D , $Y_+ \rightarrow 0\%$ and $Y_- \rightarrow 100\%$. In this case entropy

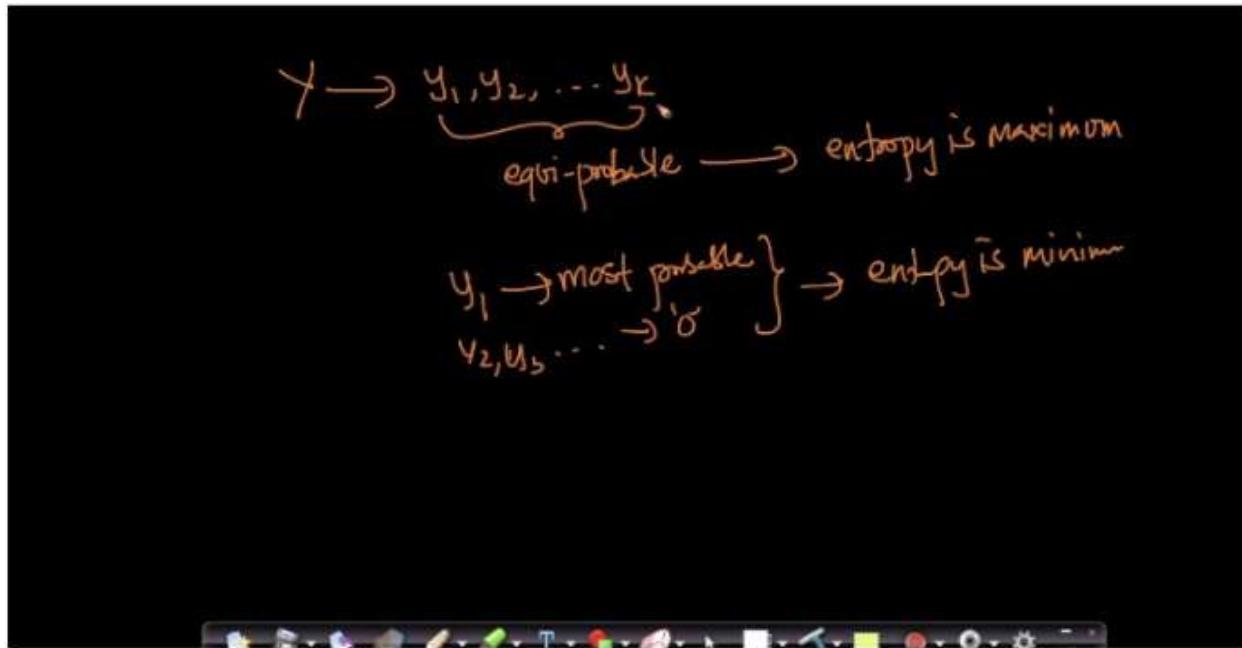
$$H(Y) = 0$$

We can see that the entropy value is maximum when the dataset is equally distributed. If one class fully dominates the other class like in case 3 the entropy is 0.



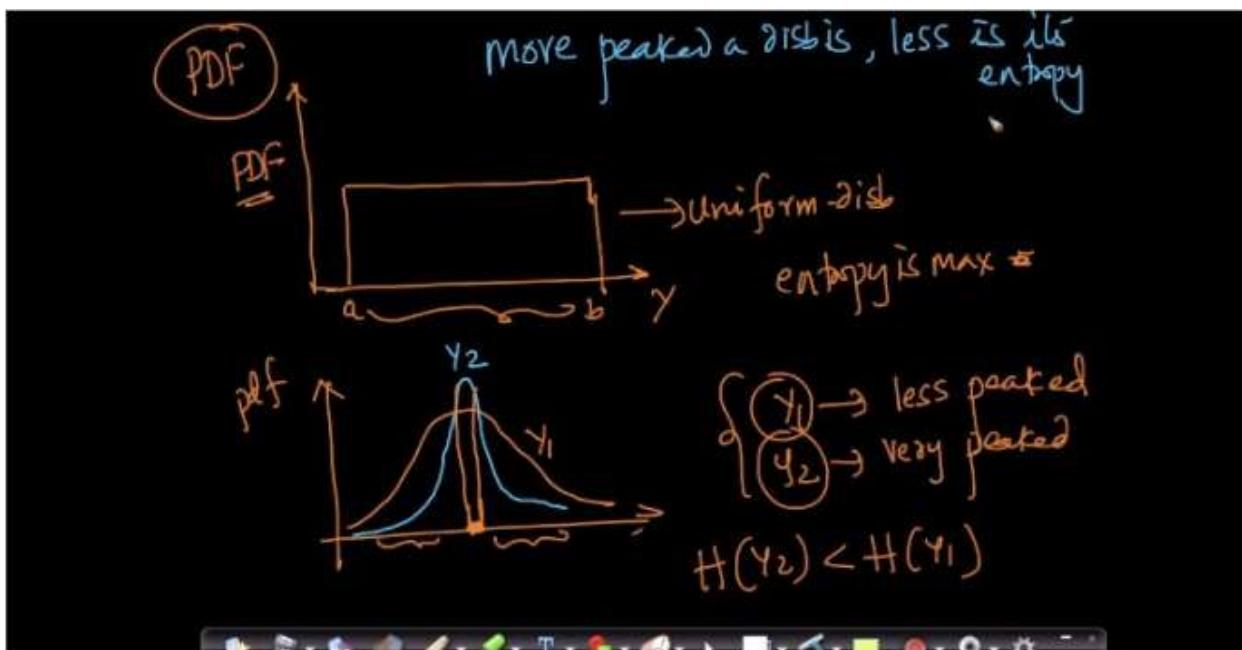
Timestamp 11:41

We can draw a plot for the probability and entropy as shown in the image above. We can see that the entropy is maximum at probability 0.5. If one class has probability 0.5, the other class will also have a probability of 0.5, since $P(Y_+) = 1 - P(Y_-)$. Also note that the graph is symmetric.



Timestamp 12:55

Similarly in a multi class setting where a random variable Y can take multiple values say $\{y_1, y_2, y_3, \dots, y_k\}$, the entropy will be maximum when all the values are equi-probable. Also the entropy will be minimum when one class heavily dominates all others.



Timestamp 17:40

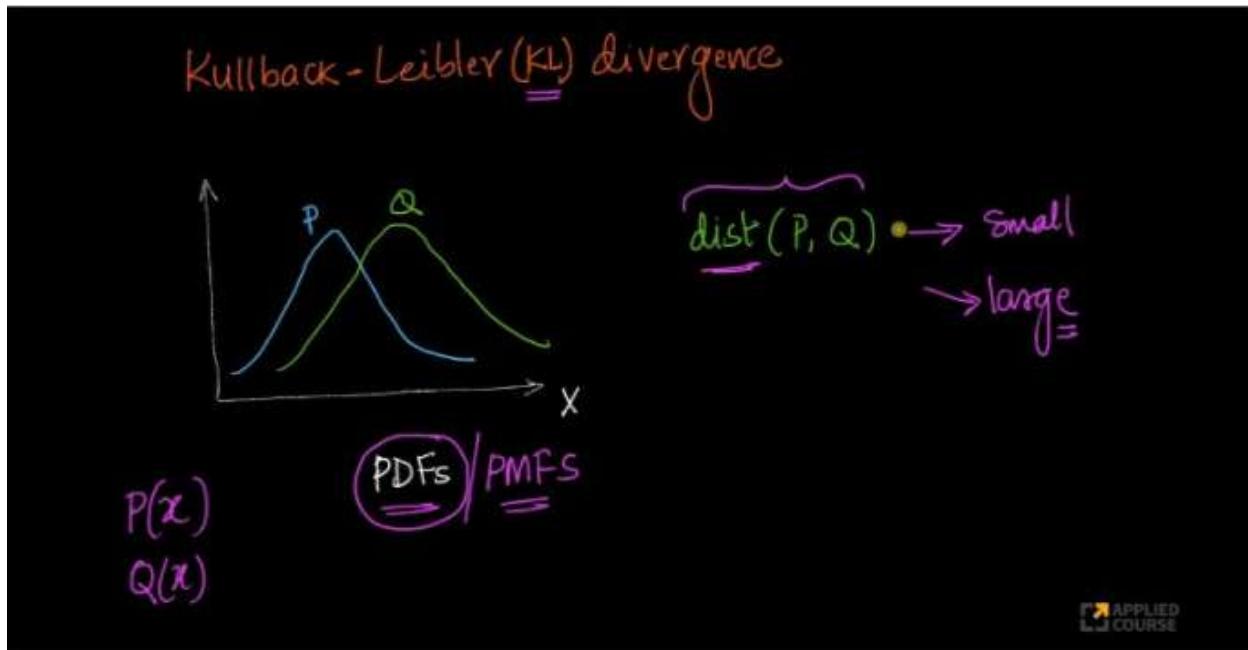
Now let's look at entropy from a probability density point of view.

Say we have a random variable Y which follows a uniform distribution, as shown in image above. Now we know that in uniform distribution all the values are equally likely. That means here we have maximum entropy.

Now let's see another case. Consider two random variables Y_1 and Y_2 both of them following a distribution as shown in the image above. We can see that Y_2 is much more peaked than Y_1 , this means that Y_1 is much closer to a uniform distribution than Y_2 . This means Y_1 will have a larger entropy than Y_2 .

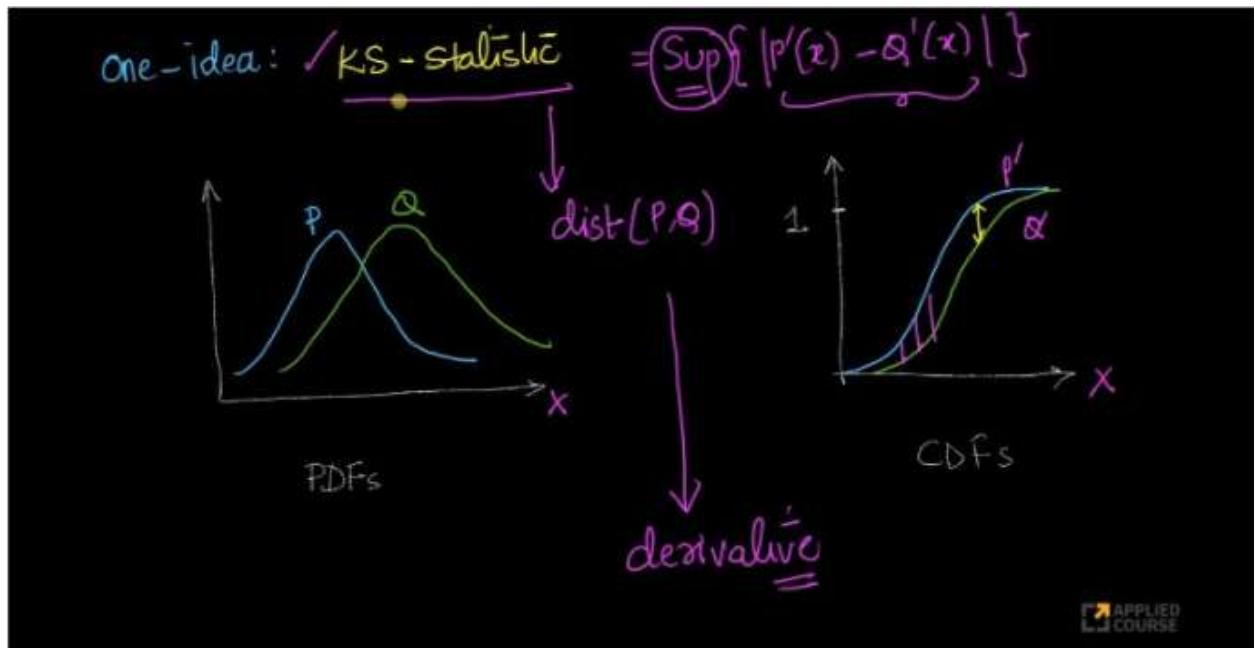
Entropy plays a major role in building Decision Trees, which we will see when we build a decision tree from scratch.

37.4 KL Divergence



Timestamp 2:07

Consider two random variables P and Q . We can see their pdf's in the above image. Now, what we are interested in is finding how similar these distributions are, i.e. we want to find $\text{dist}(P, Q)$. If this quantity is small then it means that they are very similar and if this quantity is large then it means that they are very different.



Timestamp 6:11

In the statistics chapter we learnt something known as KS statistic which can be used to find out the difference between two pdf's.

Let's see how we can use KS statistic to determine the distance between two pdfs.

First we draw the cdf's of the random variables P & Q. Let's call it P^l and Q^l as shown in the image above.

After this we compute the quantity => $\sup \{ |P^l(x) - Q^l(x)| \}$.

This quantity basically takes the absolute differences between the values of $P^l(x)$ and $Q^l(x)$, and then select the maximum among them.

It is very effective in calculating the similarity between two pdfs, but there is one issue with this, it is not differentiable.

In many of the ML algorithms when we work with optimization problems we want differentiable functions.

Info-theoretic measure

$$D_{KL}(P \parallel Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

or

$$C.R.V \leftarrow \int_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$


APPLIED COURSE

Timestamp 10:00

This brings us to KL divergence. It has its origins in Information - Theory.

KL divergence between two random variables P and Q can be defined as,

$$D_{KL}(P \parallel Q) = \sum_x P(x) \log(P(x)/Q(x)) \quad [\text{for discrete case}]$$

or

$$\int_x P(x) \log(P(x)/Q(x)) \quad [\text{for continuous case}]$$

If PDF's of two random variables are very similar, like as shown in the image above, then the quantity $P(x)/Q(x) \approx 1$. If this becomes close to 1, log becomes close to 0. So, the term $D_{KL}(P \parallel Q)$ becomes close to 0, showing that these two distributions are similar. Similarly we can verify it for the case when the distributions are dissimilar.

So, KL divergence has higher value when the difference between two random variables is large and the value is small if the difference is small.

a.k.a relative entropy

$$D_{KL}(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

$$= \sum_x P(x) \log P(x) - \sum_x P(x) \log Q(x)$$

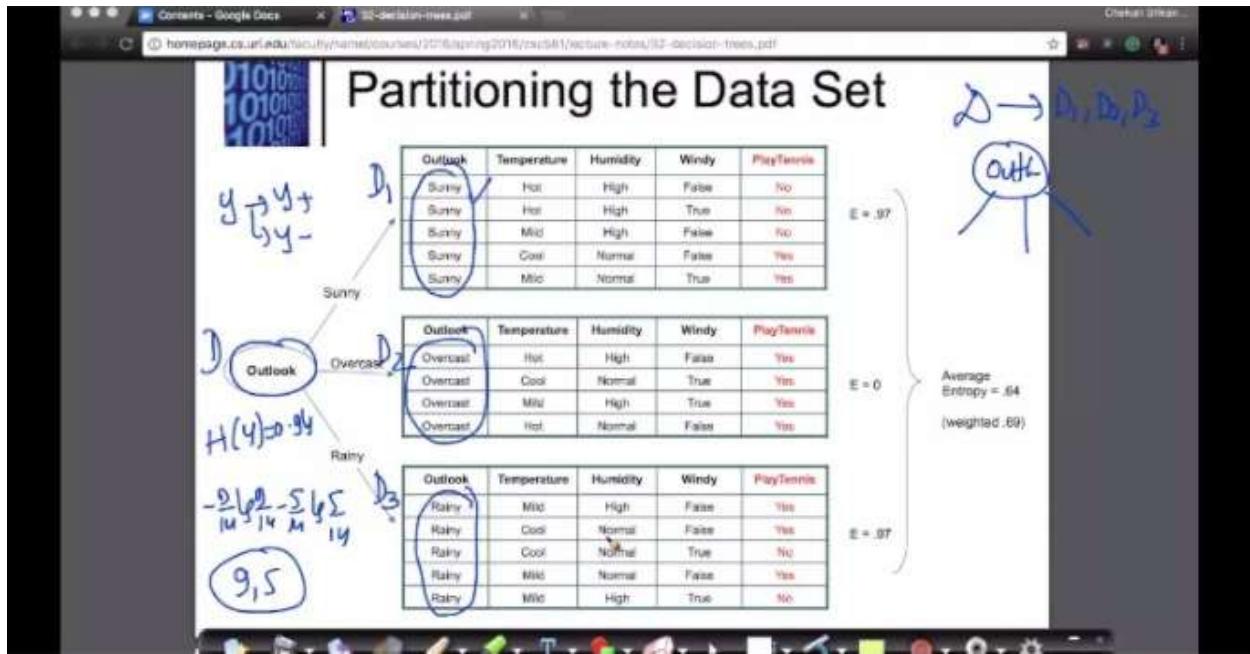
 

Timestamp 12:24

One can ask a question why there is a log in the KL divergence. The reason is that KL divergence is derived from entropy and entropy contains a log. In fact KL divergence is also known as relative entropy.

The most important factor of KL divergence is that it is differentiable unlike KS statistic.

37.5 Building a decision Tree:Information Gain



Timestamp 2:33

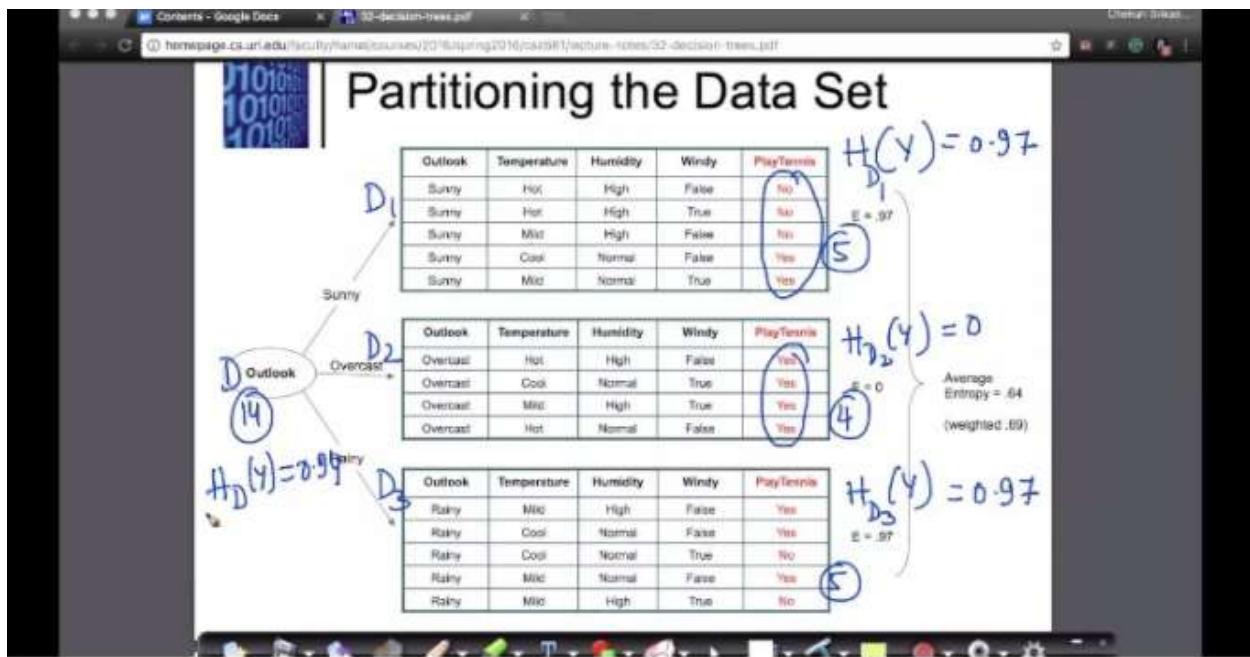
Let's again take the tennis dataset that we saw earlier.

Here we want to partition the dataset or split our dataset D.

Considering the feature outlook, we saw that it can have three values i.e. sunny, overcast and rainy. We will split the dataset D based on these categories. As you can see from the above image we have three sets, the first set D_1 has outlook = sunny, second set D_2 has outlook = overcast and third set D_3 has an outlook = rainy.

Now let's calculate the entropy value at the outlook node based on the target variable Y. We essentially want to calculate the entropy based on the target.

We had earlier calculated the entropy at outlook node $H_D(Y) = 0.94$



Timestamp 3:44

Now, we will calculate the entropy for each of the splitted datasets.

$$H_{D_1}(Y) = 0.97$$

$$H_{D_2}(Y) = 0$$

$$H_{D_3}(Y) = 0.97$$

In total D has 14 data points. After splitting D_1 has 5 data points, D_2 has 4 data points, D_3 has 5 data points.

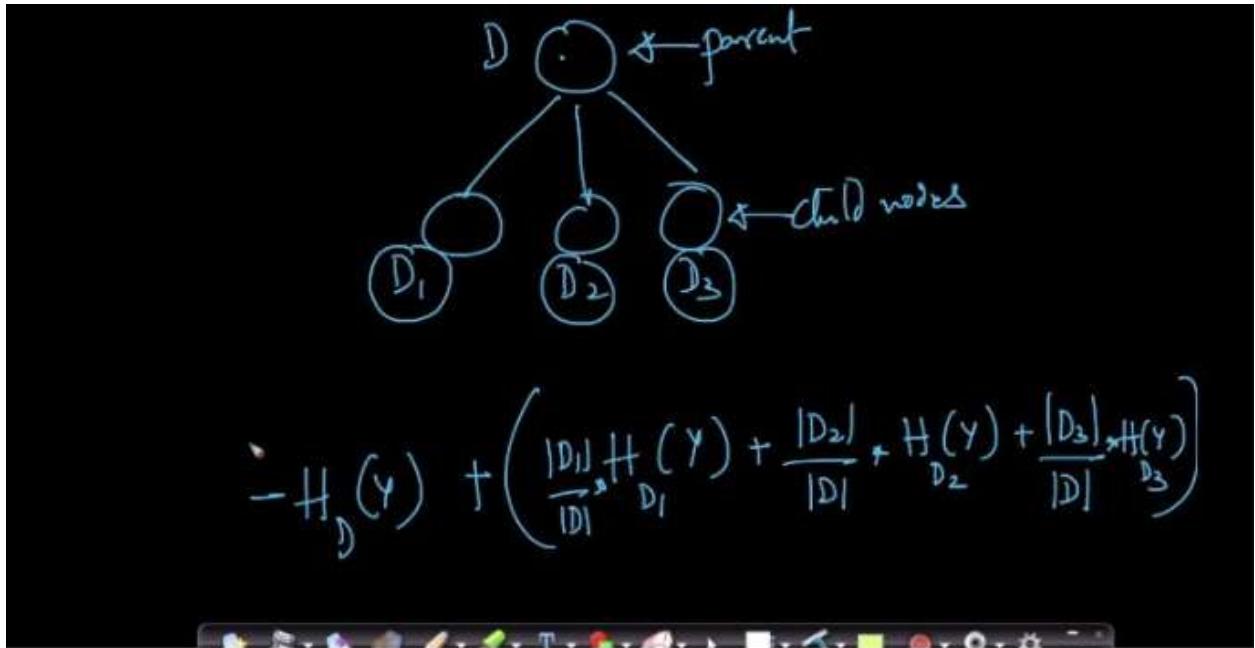
$$\begin{aligned}
 \text{IG}(Y, \text{outlook}) &= \left(\frac{5}{7} \times 0.97 \right) - (0.94) = \underline{\underline{1.6}} \\
 \text{Weighted entropy} &\quad \downarrow \\
 \text{after } D_1, D_2, D_3 & \quad \left(\frac{5}{14} \times 0.97 \right) + \left(\frac{4}{14} \times 0 \right) + \left(\frac{5}{14} \times 0.97 \right) \\
 &\quad \downarrow \\
 & \quad \frac{\sum 0.97 \times 2}{14}
 \end{aligned}$$

Timestamp 7:09

Note: There is a small correction in the formula from the one that we have on the slide. Corrections are reflected in the notes.

Now, Information Gain after splitting w.r.t outlook for target Y is,
 $\text{IG}(Y, \text{outlook}) = 0.94 - (5/14 \times 0.97 + 4/14 \times 0 + 5/14 \times 0.97)$

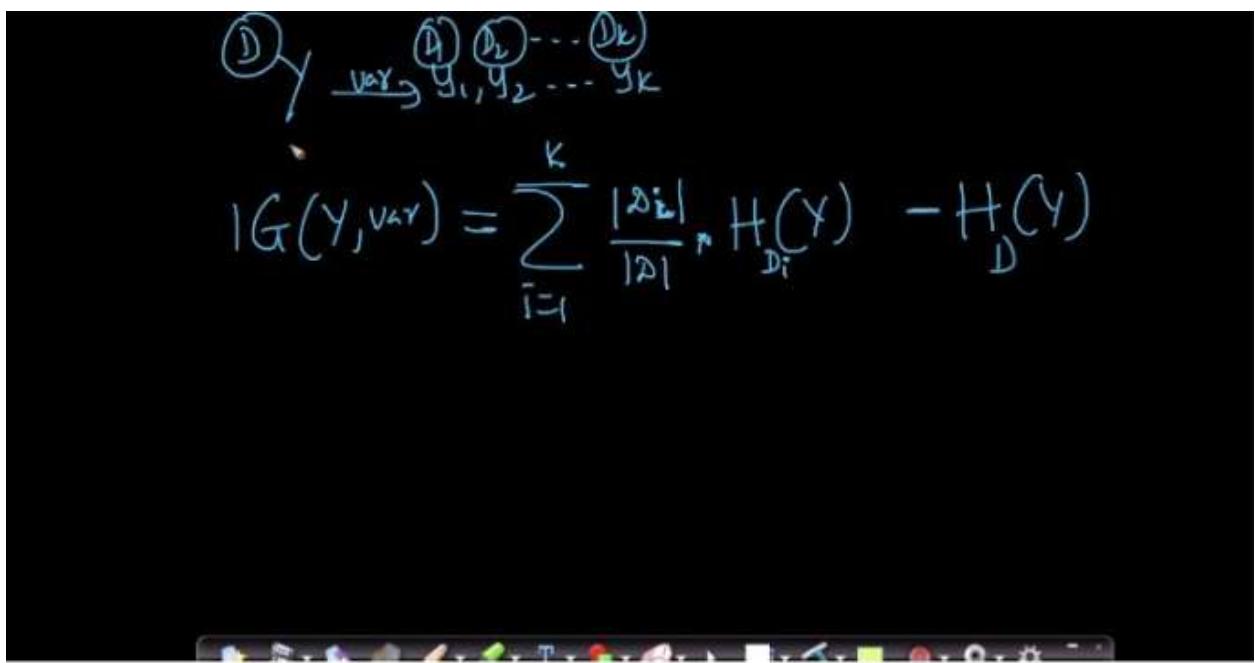
$(5/14 \times 0.97 + 4/14 \times 0 + 5/14 \times 0.97) \Rightarrow$ this is the weighted average of the entropy by the no of data points each set has.



Timestamp 8:25

So, for our dataset D the information gain after splitting the data into D_1, D_2, D_3 can be written as,

$$IG = H_D(Y) - (|D_1|/|D| * H_{D_1}(Y) + |D_2|/|D| * H_{D_2}(Y) + |D_3|/|D| * H_{D_3}(Y))$$



Timestamp 9:20

So, in general if we have dataset D which is broken into k subsets say D_1, D_2, \dots, D_k using some variable var.

Then, Information Gain w.r.t Y_i 's of these dataset can be written as,

$$IG(Y, \text{var}) = H_D(Y) - \sum_{i=1}^k |D_i|/|D| * H_{D_i}(Y)$$

Information_Gain = [Entropy(parent)] – [Weighted Average Entropy of child nodes]

37.6 Building a decision Tree: Gini Impurity

Gini Impurity ~ similar to Entropy

$$I_G(Y) = 1 - \sum_{i=1}^k (P(y_i))^2 \quad Y \rightarrow y_+$$

$$Y \rightarrow y_1, y_2, y_3, \dots, y_k$$

Case 1: $P(y_+) = 0.5$
 $P(y-) = 0.5$

Case 2: $P(y_+) = 1$
 $P(y_0) = 0$

$$I_G(Y) = 1 - (1+0) = 0$$

$$H(Y) = 1$$

$$I_G(Y) = 1 - (0.25 + 0.25) = 0.5$$

Timestamp 2:37

Here we will talk about Gini Impurity which is similar to entropy but has certain advantages over it. It is represented as $I_G(Y)$ (information gain is represented as $IG(Y)$ both are not the same).

Gini Impurity for a random variable Y , which is split into k sets say y_1, y_2, \dots, y_k is

$$I_G(Y) = 1 - \sum_{i=1}^k ((P(Y_i))^2$$

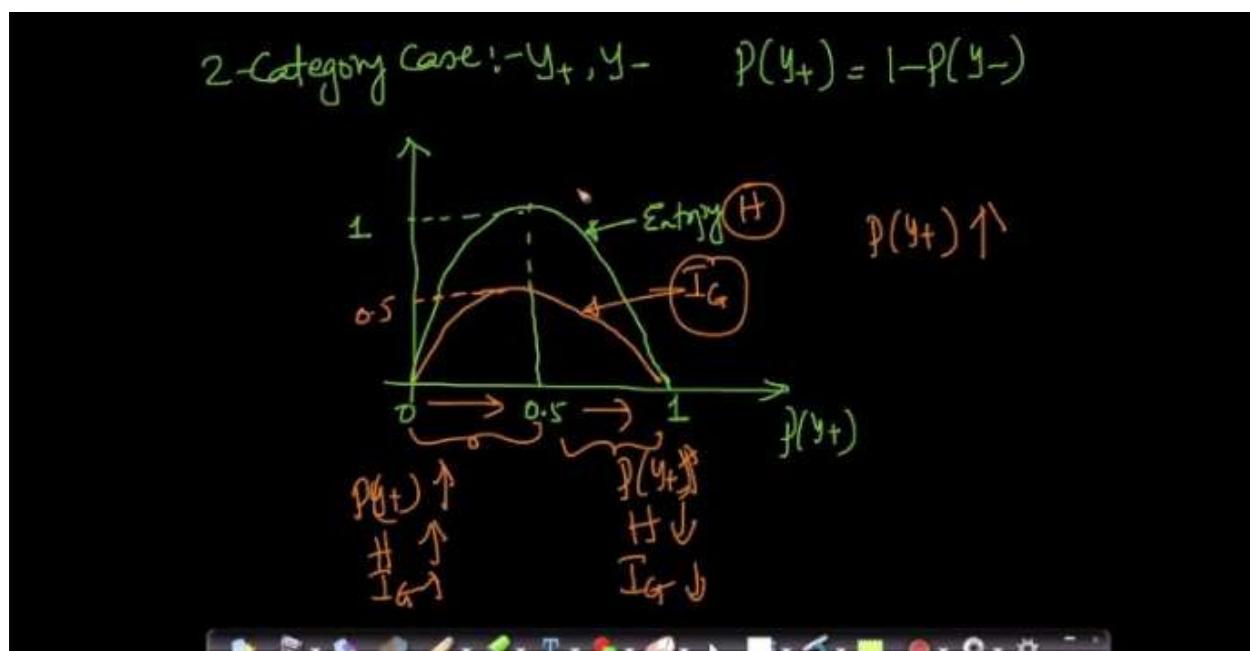
Let's compare it with entropy. Say we have a random variable Y with two classes Y_+, Y_- .

Case 1: $P(Y_+) = 0.5, P(Y_-) = 0.5$

then $I_G(Y) = 1 - (0.25 + 0.25) = 0.5, H(Y) = 1$

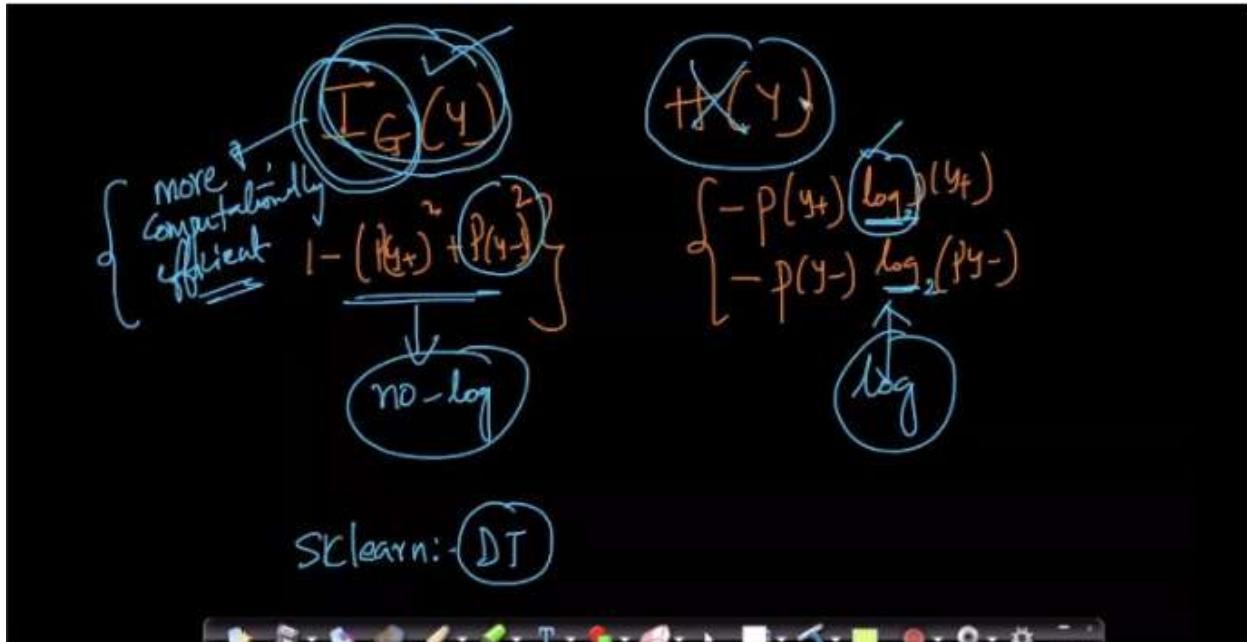
Case 1: $P(Y_+) = 1, P(Y_-) = 0$

then $I_G(Y) = 1 - (1 + 0) = 0, H(Y) = 0$



Timestamp 4:28

In the above image we have plotted gini impurity and entropy for the two class case. We can see that the maximum value gini impurity can take is 0.5 whereas the maximum value that entropy can take is 1. Gini Impurity seems like a scaled down version of entropy. Apart from this both are quite similar.

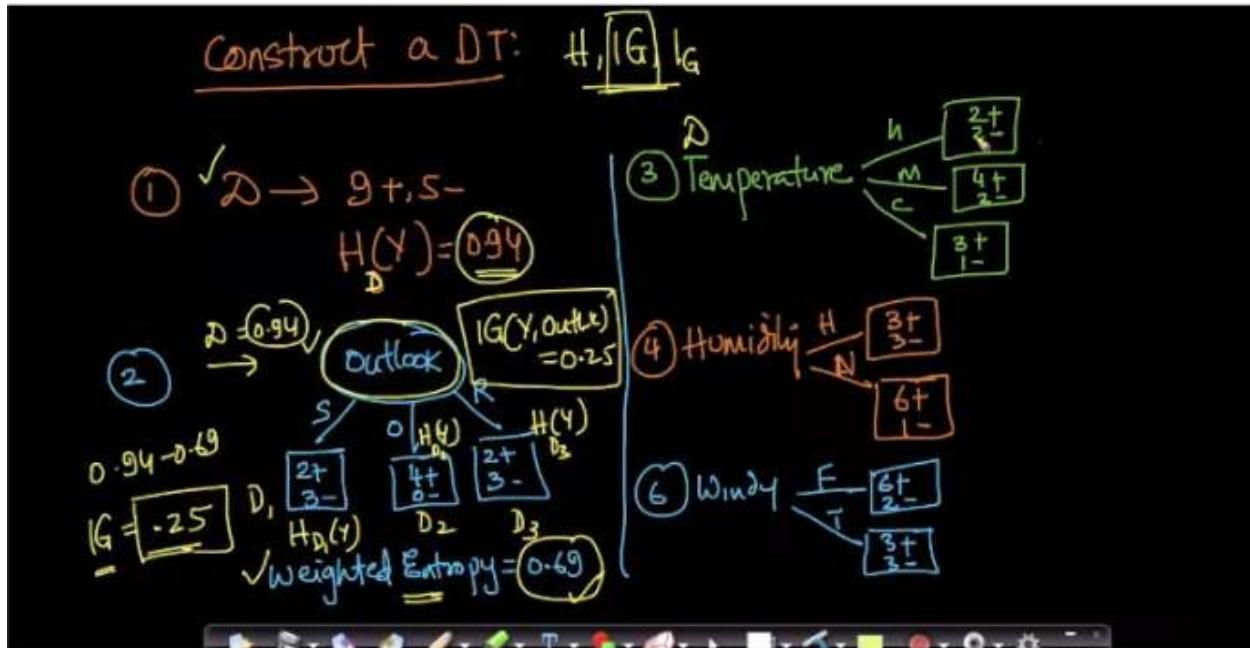


Timestamp 6:45

So, now the question is why did we study gini impurity? There is one fundamental advantage of gini impurity over entropy. As you can see from the above image we have written formulas for both these quantities. Entropy contains a log term which is computationally expensive whereas in gini impurity we just need to do simple arithmetic operations.

This is the reason why most people use gini impurity instead of entropy while calculating information gain. Also in sklearn implementation of decision trees gini impurity is used as default.

37.7 Building a decision Tree: Constructing a DT



Timestamp 2:50

Till now we have learnt about entropy, information gain, and gini impurity. Let's now put them to use and build a decision tree.

We build decision trees using the top down approach i.e we start with the dataset as a whole and then break them into chunks.

Here we will use the tennis dataset D which has 9 positive and 5 negative data points. So the entropy of whole dataset before splitting is $H_D(Y) = 0.94$.

Now for splitting we take all the features one by one and calculate the Information Gain.

Say at first we split our dataset D using outlook and calculate the weighted entropy for this splitted dataset which is 0.69 as shown in the image above. So the information gain is $0.94 - 0.69 = 0.25$

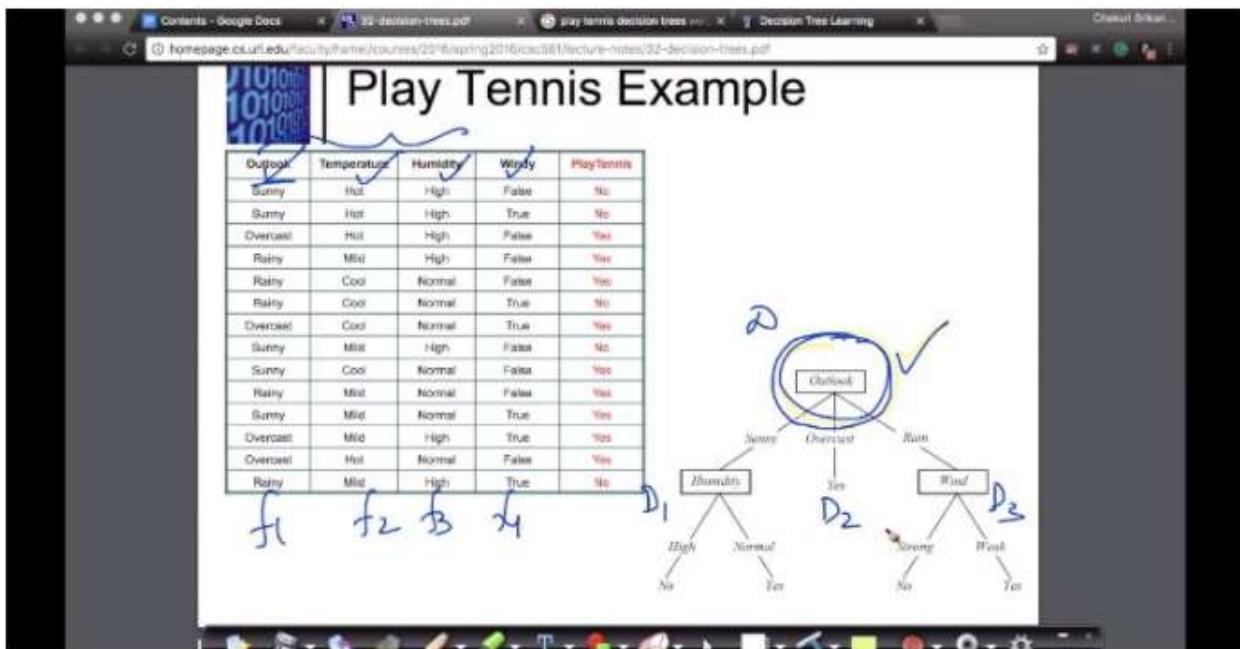
So, $IG(Y, \text{outlook}) = 0.25$

$$IG(Y, f) = H_D(Y) - \left(\sum_{f=1}^k \frac{|D_f|}{|D|} * H_D(Y_f) \right) \rightarrow \text{Choosing the root node}$$

$\left\{ \begin{array}{l} IG(Y, \text{outlook}) = 0.25 \\ IG(Y, \text{Temp}) = - \\ IG(Y, \text{Humidity}) = - \\ IG(Y, \text{Wind}) = - \end{array} \right.$

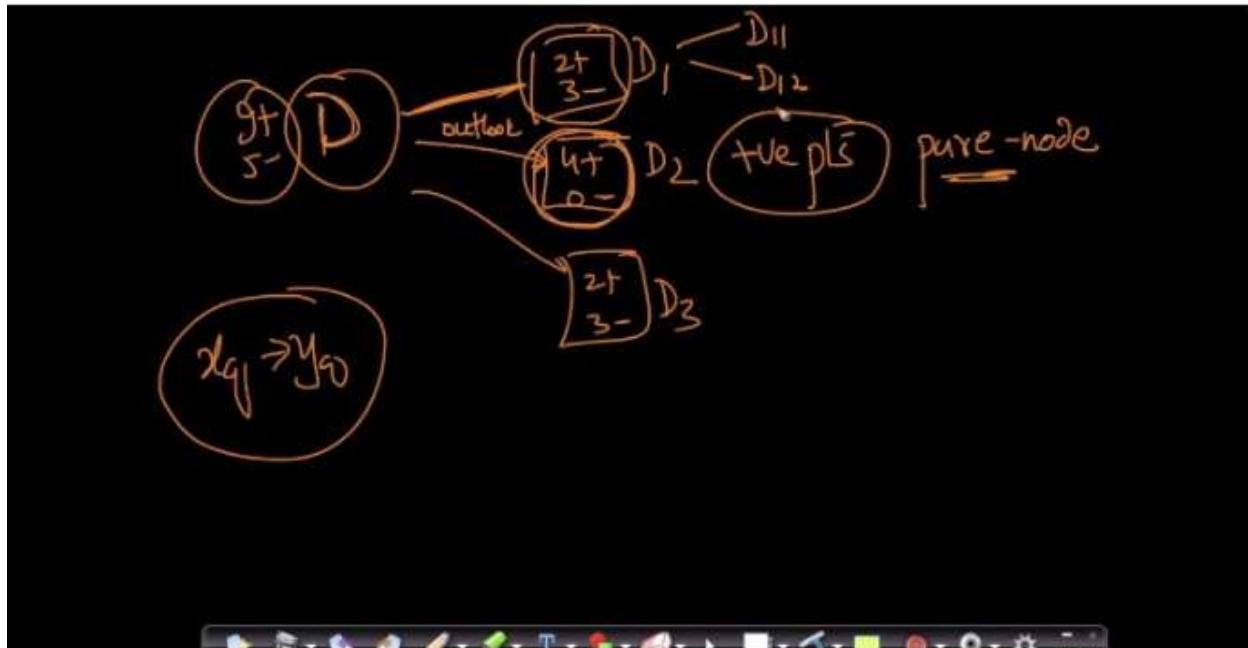
Timestamp 6:05

Right now we are finding a split for the root node. So, what we essentially do is to calculate the information gain for all the features and then select that feature which has the maximum Information Gain.



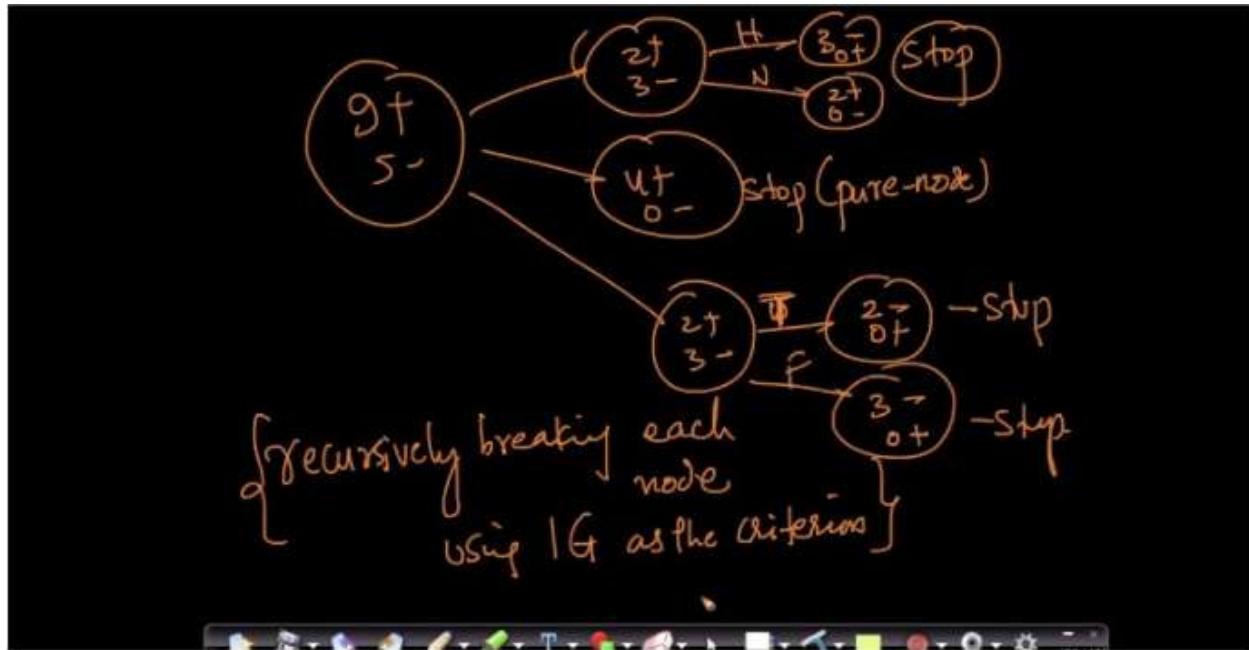
Timestamp 7:10

After calculating the information gain for all the features w.r.t D we find that outlook has the maximum information gain. So it becomes the root of the tree as shown in the image above. After splitting the dataset D with outlook it results in three different datasets say D_1 , D_2 and D_3 .



Timestamp 9:28

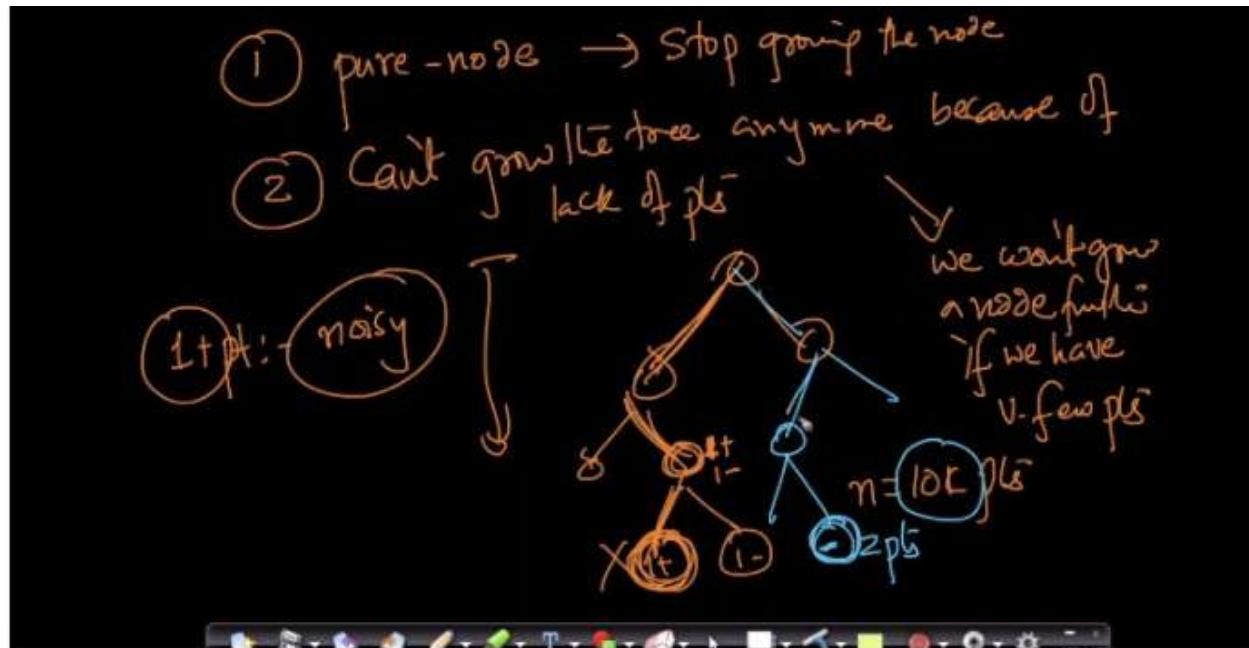
D_1 contains 2 positive points and 3 negative points, D_2 contains 4 positive points and 0 negative points and D_3 contains 2 positive points and 3 negative points. Now we can see that D_2 contains only positive points. Such a node is called a pure node and we don't split pure nodes.



Timestamp 15:02

Again we can break D_1 using humidity, this results in two pure nodes. Similarly D_3 can be splitted using wind that will result into two pure nodes.

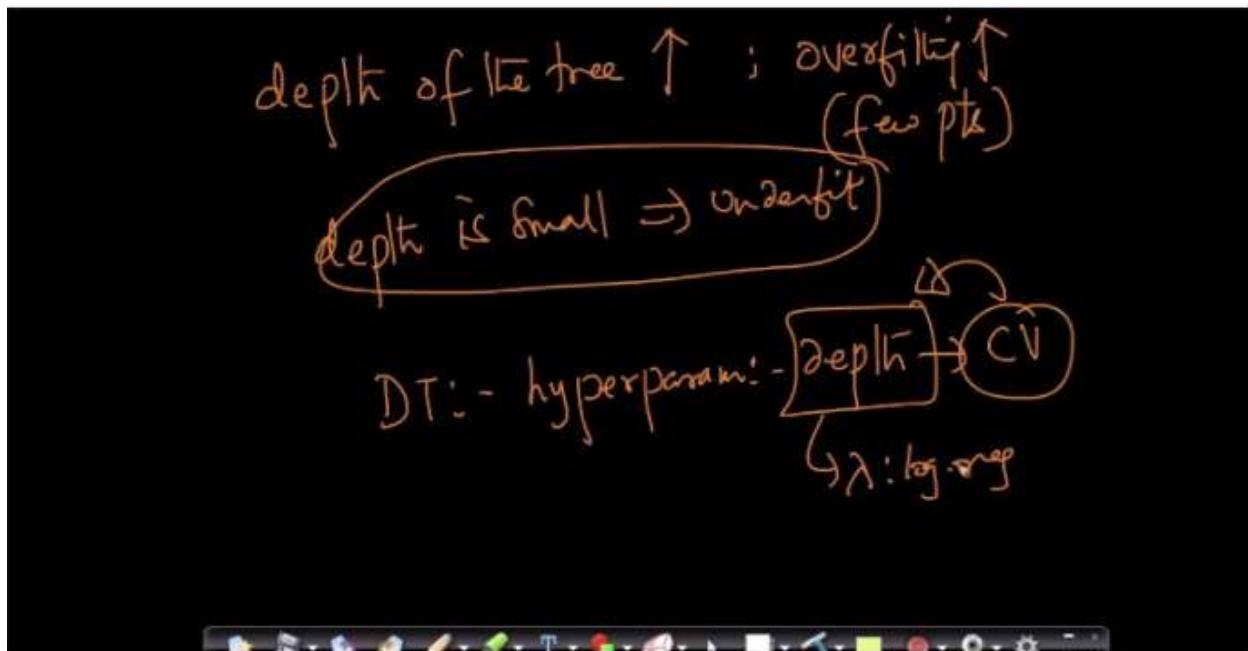
So, essentially we are recursively breaking each node into child nodes using Information Gain as the criterion.



Timestamp 18:12

Now when should we stop further splitting a node:

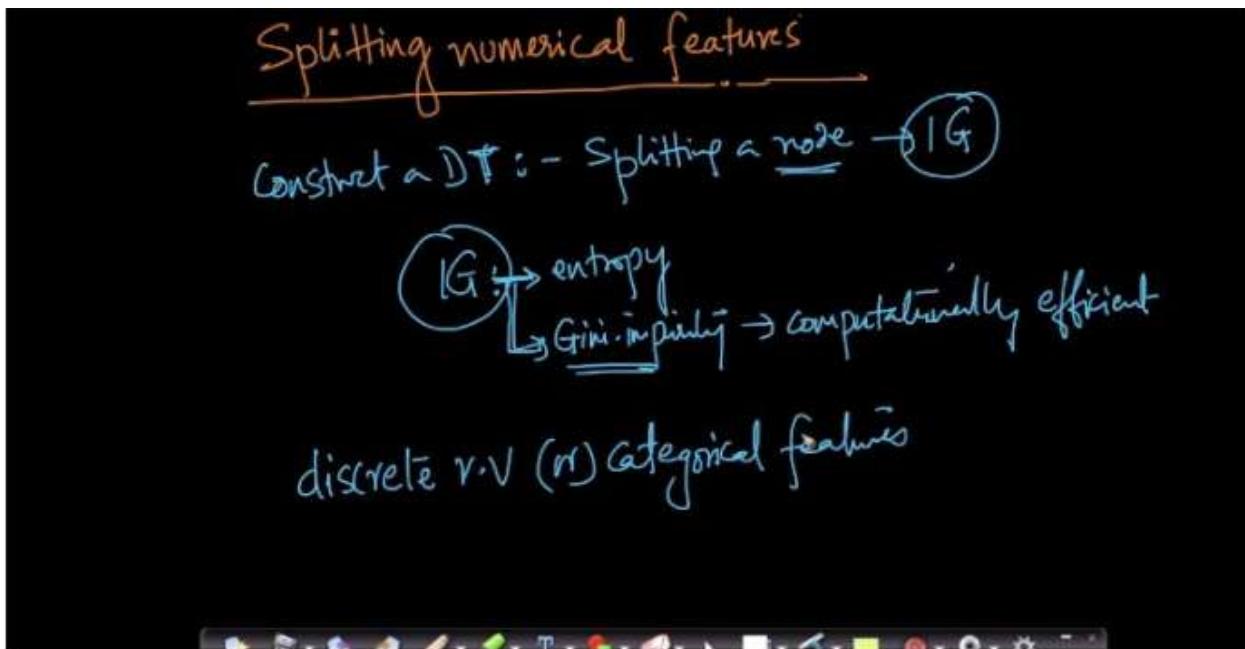
1. Pure node - we stop growing our tree when we have a pure node because there is no point in splitting our node after it.
2. We also stop splitting if there are very few points in a node. Say we have a dataset of 10k points and in one of the nodes we have only 2 points - 1 positive and the other negative. Now if we split this node then we are giving way to importance to a single data point. This data point can be an outlier. So in a way we will overfit.



Timestamp 20:25

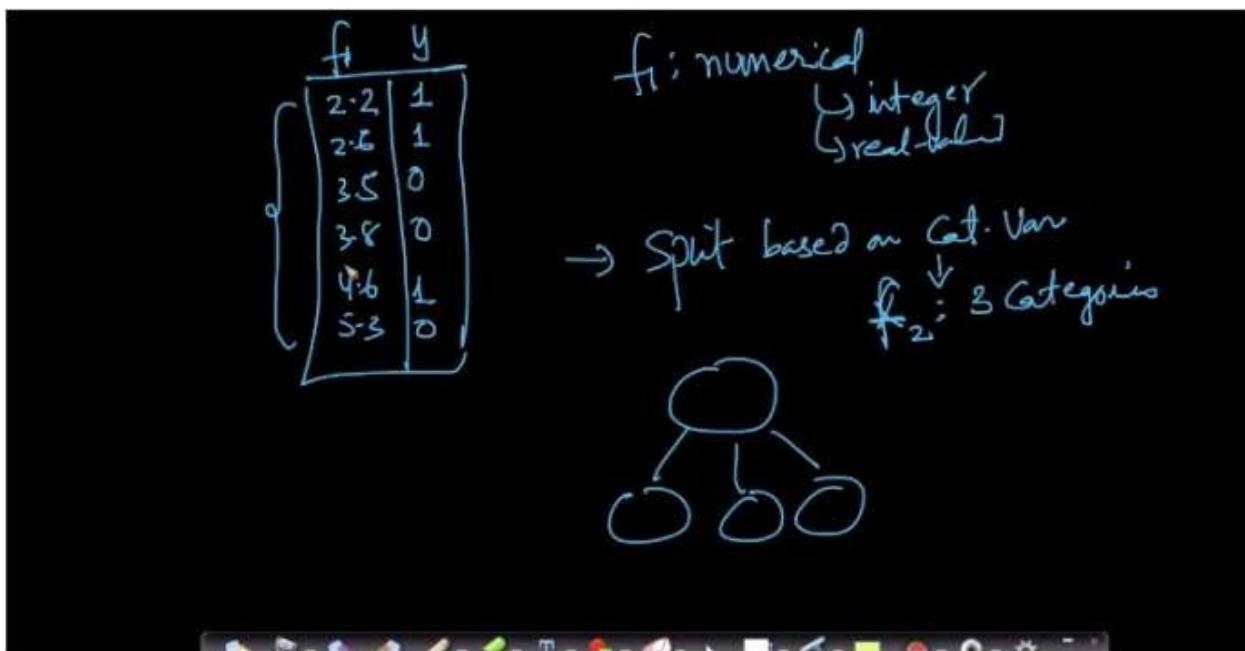
3. We also stop growing our tree beyond a certain depth, because as the depth increases there are fewer and fewer data points remaining. If we fit to these points then we are overfitting. So, in general if the depth is large we are overfitting and if the depth is small we are underfitting. depth is a hyperparameter, which we find using standard practices like cross-validation.

37.8 Building a decision Tree: Splitting numerical features



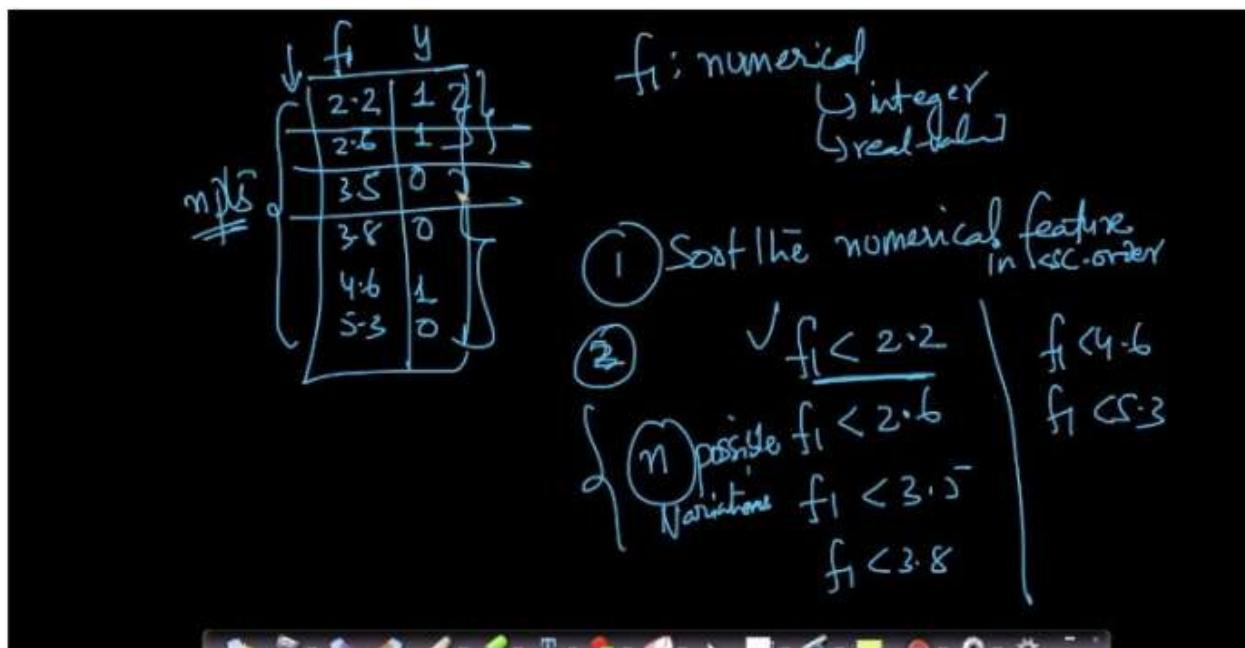
Timestamp 1:15

In the last lecture we learnt how to build a decision tree. The main operation for building a decision tree was splitting a node and calculating the information gain. All the examples we have seen till now were based on categorical variables. Now how do we handle numerical features?



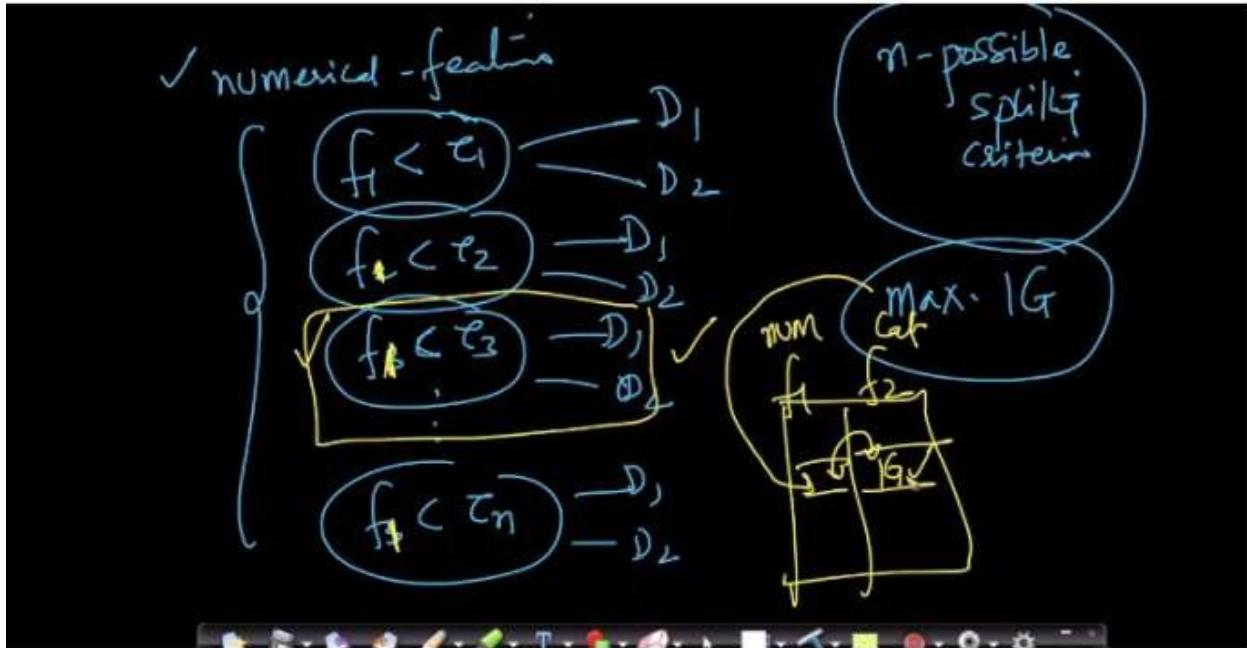
Timestamp 2:36

Numerical features are continuous in nature and it cannot be treated in the same way as categorical features. Say a categorical feature had 3 categories then we split it up into three 3 child nodes, but a numerical feature can have countless unique values.



Timestamp 4:54

Consider that f_1 is a numerical feature with values as shown in image above. Now what we can do is sort the values into ascending order. After sorting we can pick a value say 2.6 from this column and make a rule such that all rows having values less than this threshold will go to the left branch and otherwise right branch. If there are n rows in the dataset then there are at most n different possibilities for this threshold.



Timestamp 7:14

So essentially we need to check every value in our numerical feature. We take each value from the numerical feature column and use this as threshold and calculate the information gain. Among all the splits we select the one that caused the maximum information gain. This is extremely time consuming. Also we need to compare the information gain with other features information gain.

37.9 Feature standardization

Feature Standardization:

dist $\{ \begin{matrix} \text{Logistic regn} \\ \text{SUM} \\ \text{KNN} \end{matrix} \} \rightarrow \text{feature stdn}$

$y \quad \textcircled{5}$

$\begin{matrix} f_j \\ z_{ij} \\ d \\ \mu_j, \sigma_j \end{matrix}$

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

Timestamp 1:15

Algorithms like logistic regression, SVM, KNN are all distance based algorithms, so feature standardization was very important for them to work properly. Now decision trees is not a distance based method. So feature standardization is not important.

Decision Trees: - not a distance based metric

do not need to perform feature stdn

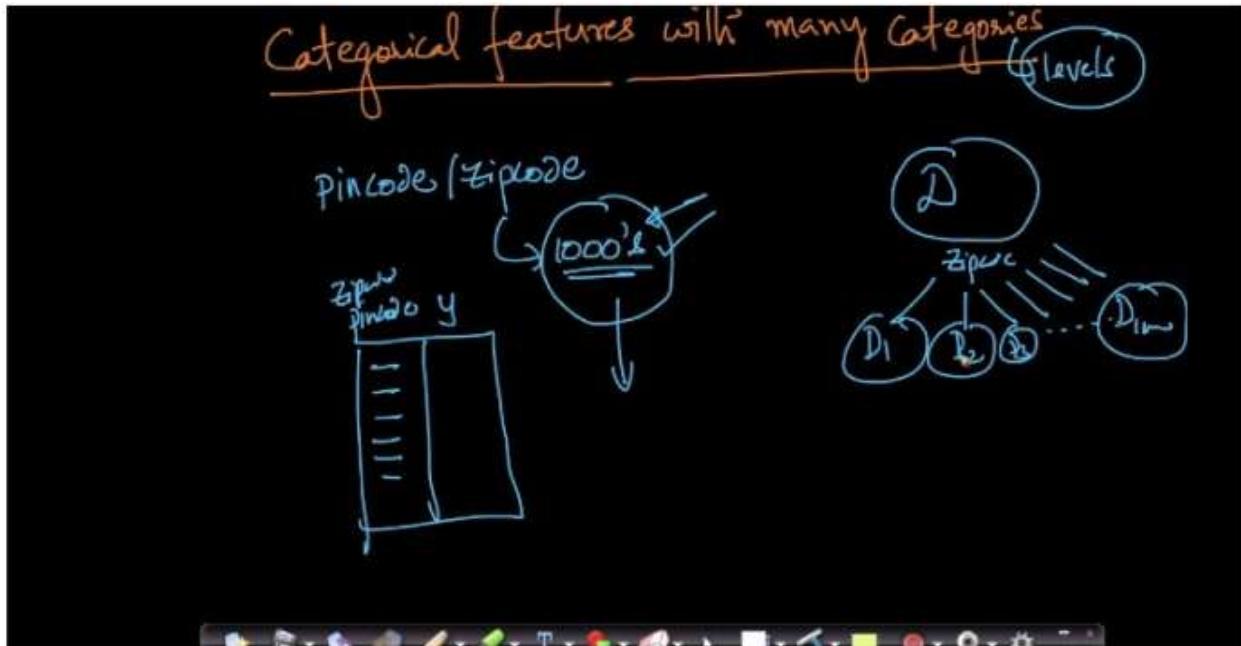
$\begin{matrix} f_j \\ z_{ij} \\ \text{threshold} \\ \text{sort this column} \\ \text{order} \end{matrix}$

$f_j < c$

Timestamp 4:01

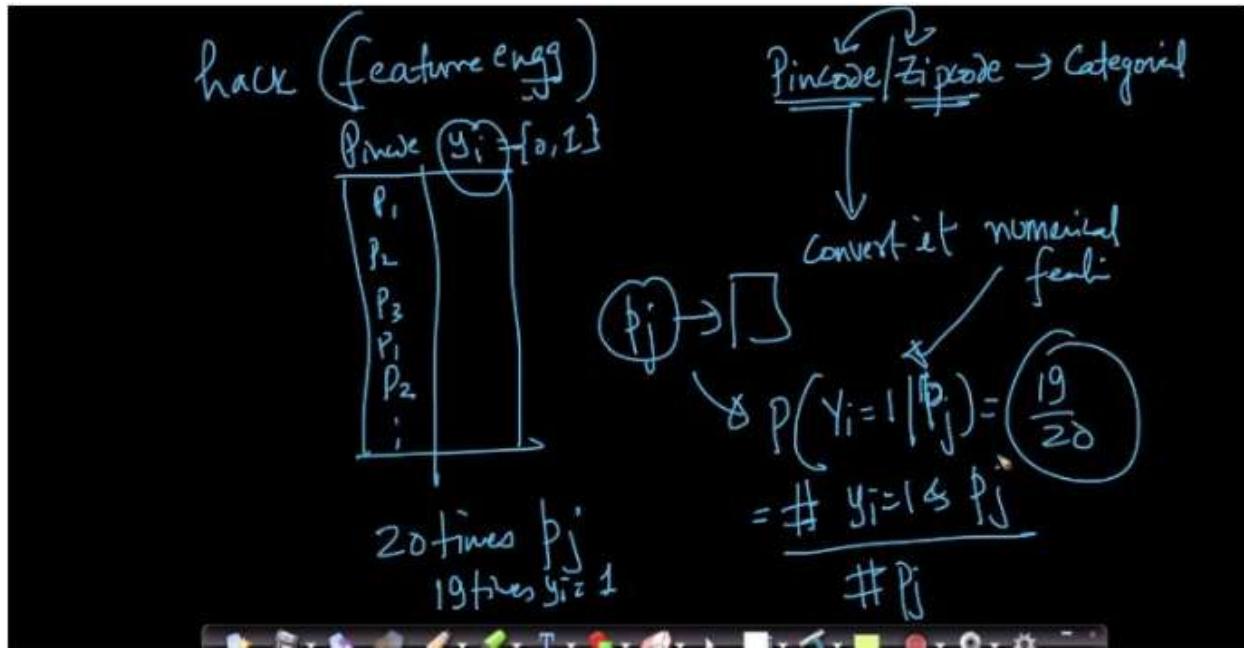
When we dealt with numerical features we sorted the column and chose a threshold value. So only ordering of the data mattered.

37.10 Building a decision Tree:Categorical features with many possible values



Timestamp 2:13

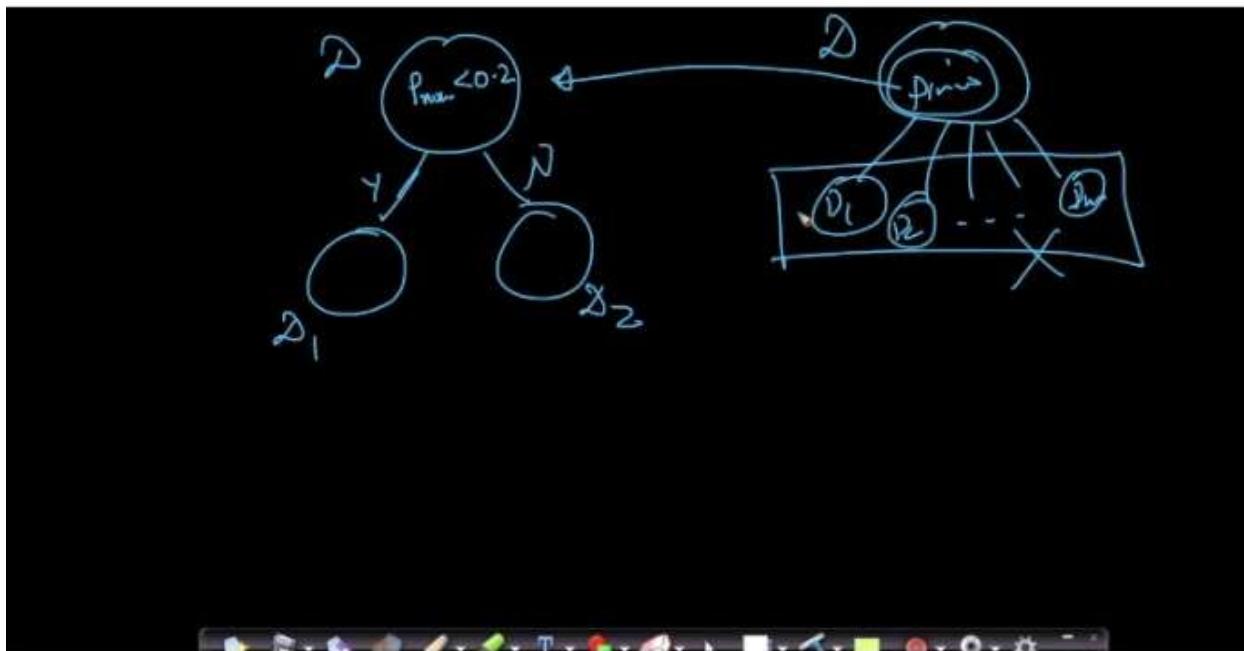
Suppose in our dataset we have a feature like PINCODE/ZIPCODE. Now it can contain 1000s of different values. It cannot be treated like numerical features because we cannot really compare two PINCODEs. If we treat it like a categorical feature then we need to split it into various categories, the categories can now run into thousands. Each split may not contain many data points. This can be problematic.



Timestamp 4:20

In order to deal with this problem we use a feature engineering hack that works quite well in practice.

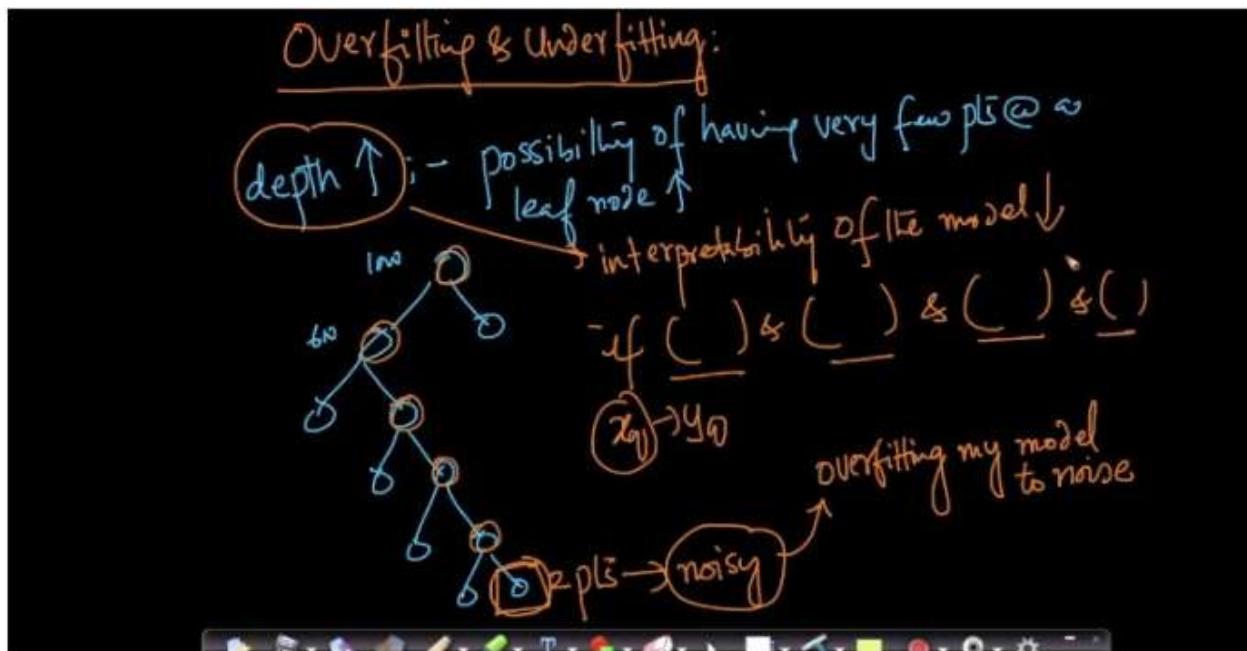
Essentially we convert our categorical data into numerical data. Consider the column PINCODE as shown in the image above. Say, a row has $\text{PINCODE} = P_j$ and class label $y_i = 1$, then we need to calculate the probability $P(y_i = 1 | P_j)$, then we can replace the PINCODE P_j with this probability value.



Timestamp 6:22

After converting the categorical data to a numerical feature we can use this column to split our features as we learnt earlier. This considerably reduces the number of child nodes we have.

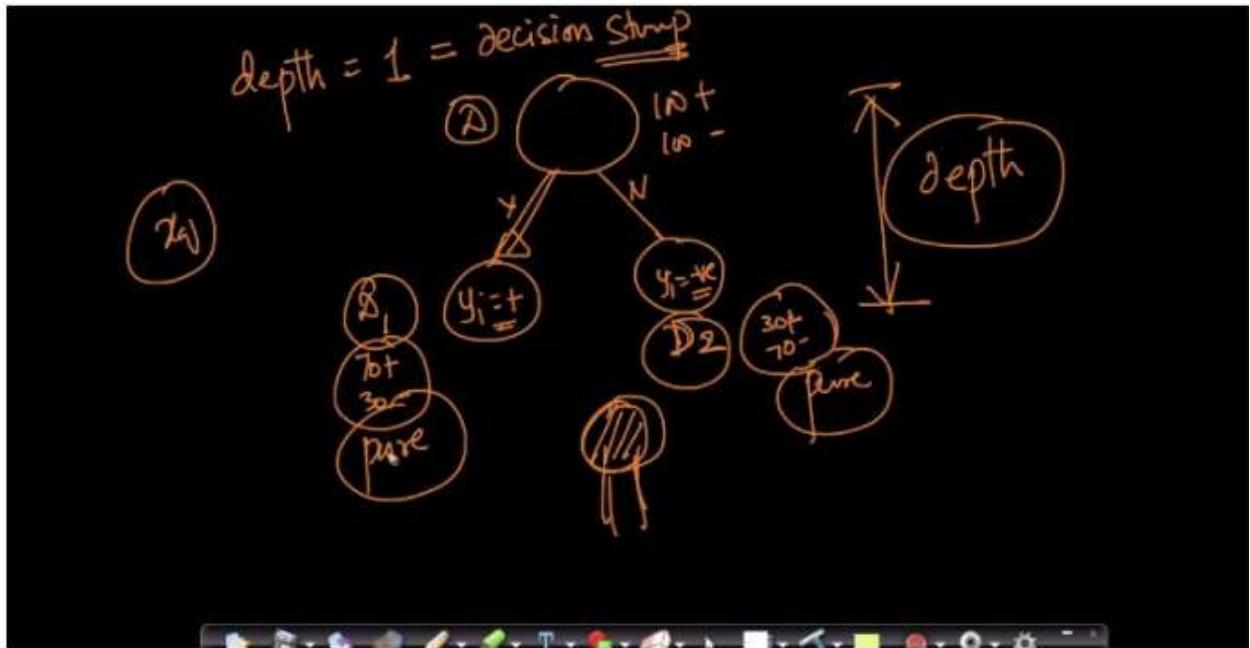
37.11 Overfitting and Underfitting



Timestamp 2:08

Now let's talk about the cases where our model may underfit or overfit.

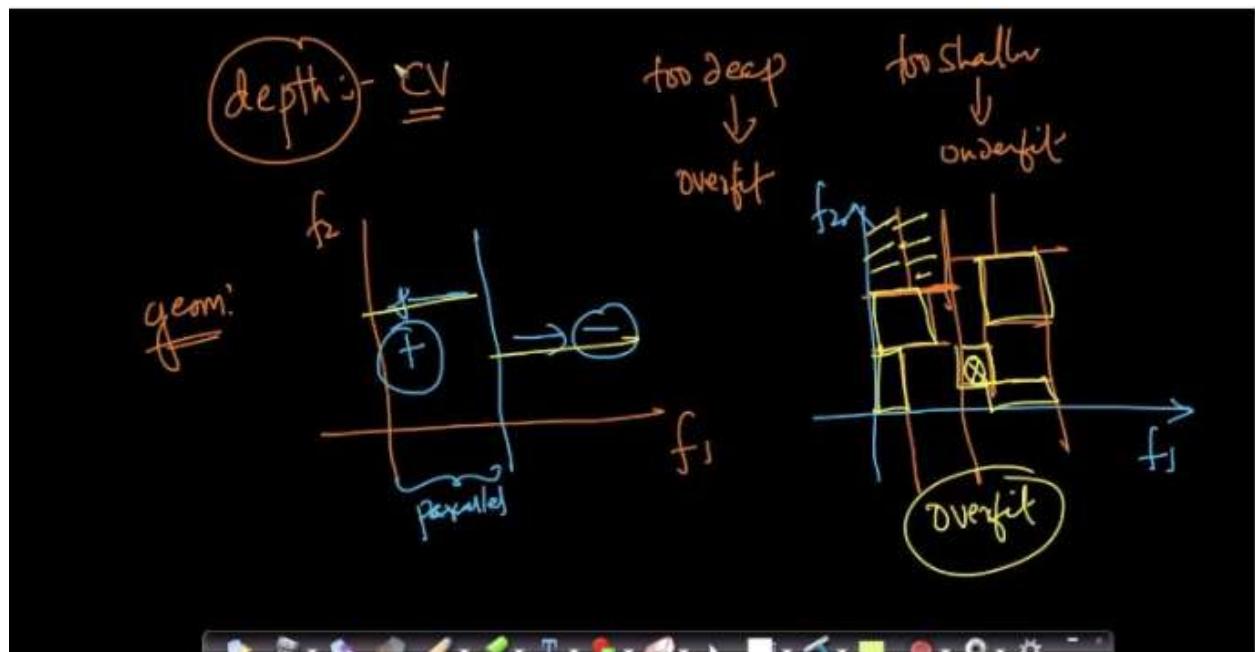
Suppose we have a tree with large depth then it may happen that the leaf nodes will contain very few points. These points can be noisy points. So, in a way we are overfitting our model. Also the interpretability of the model goes down as the depth of the tree increases.



Timestamp 4:27

A tree can also have very small depth. Consider the tree given in the image above it's height is 1. A decision tree with height = 1 is called Decision Stump. Say we have a dataset D with 200 points with 100 positive and 100 negative. Now we make a split into two nodes, one containing 70 positive and 30 negative and the other containing 30 positive and 70 negative. Now both of them are not pure nodes. To make a prediction we take a majority vote.

So, in a way we are underfitting because the tree contains very few decision nodes and also there are no pure nodes.



Timestamp 7:10

So essentially if the depth is too high we are overfitting and if the depth is low we are underfitting.

Now let's try to see what it means to have a large or small depth geometrically.

A tree with small depth essentially means that there are very few decision nodes. As we know that decision nodes are all axes parallel hyperplanes. If these hyperplanes are very few in number then there is not much splitting of the space of solutions as shown in image above. Essentially causing underfitting.

If the tree is of large depth then there are many hyperplanes in the solution space. This results in very small sized hyper cuboids. Essentially causing overfitting.

37.12 Train and Run time complexity

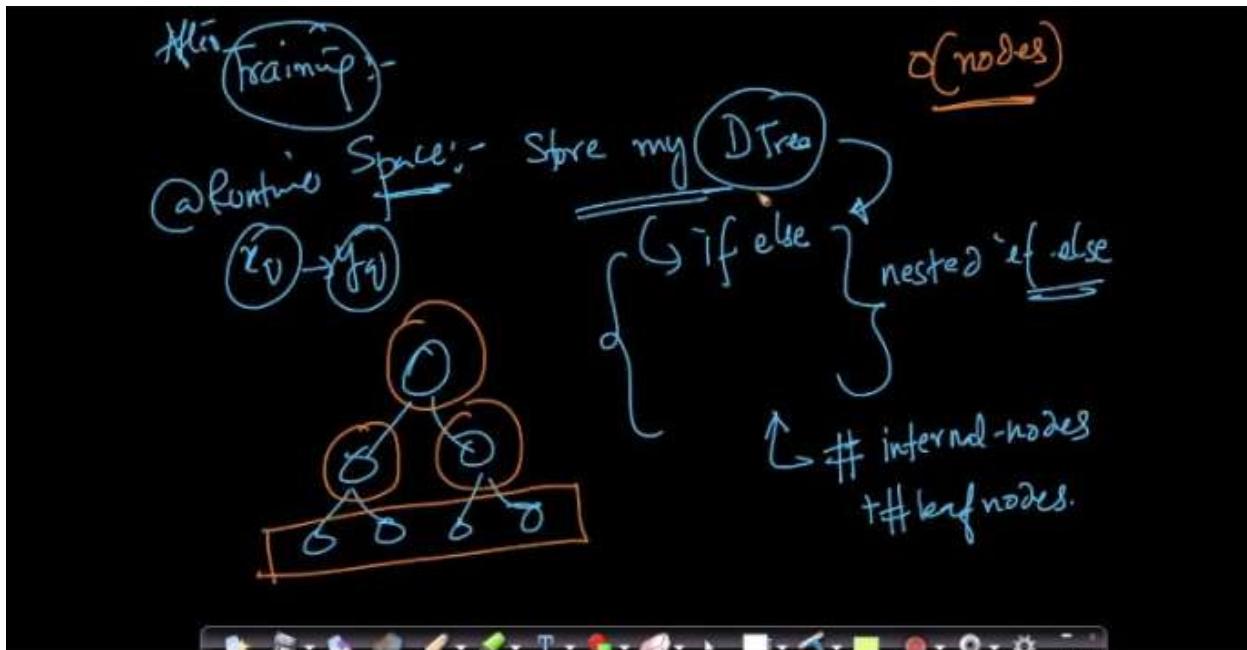
The image shows handwritten notes on a black background. At the top, the title "Train & Run time complx:" is written in blue ink. Below it, the word "Train:" is underlined in blue. To the right of "Train:", there is a formula $\sim O(n \lg n d)$. Above this formula, the words "Sorting" and "evaluate 1G" are written, with arrows pointing from "Sorting" to the first "n" and from "evaluate 1G" to the "d". To the right of the formula, there is a circled "n = #pts Train" and below it, "d = dim.". Below the main formula, the word "(Time)" is written in parentheses. To the right of "(Time)", the words "numerical features: - (threshold)" are written, with an arrow pointing from "numerical features" to "threshold". Below "numerical features", the words "algebraic methods" are written. To the left of the main formula, there is a circled "nlgn" with an arrow pointing to "Sorting". To the right of the main formula, there is a circled "O(n lg n d)" with an arrow pointing down to it, and a circled "larged" with an arrow pointing to it. At the bottom of the image, there is a toolbar with various icons.

Timestamp 1:40

Train time complexity for Decision Trees is $O(n \lg n * d)$, where $n = \#$ data points in train, $d = \#$ dimension.

The $n \lg n$ part is there because of the sorting we need for features to select a threshold. d represents the number of dimensions i.e. no of features in the dataset. We learnt earlier that numerical features were time consuming but there are certain hacks to get around this problem.

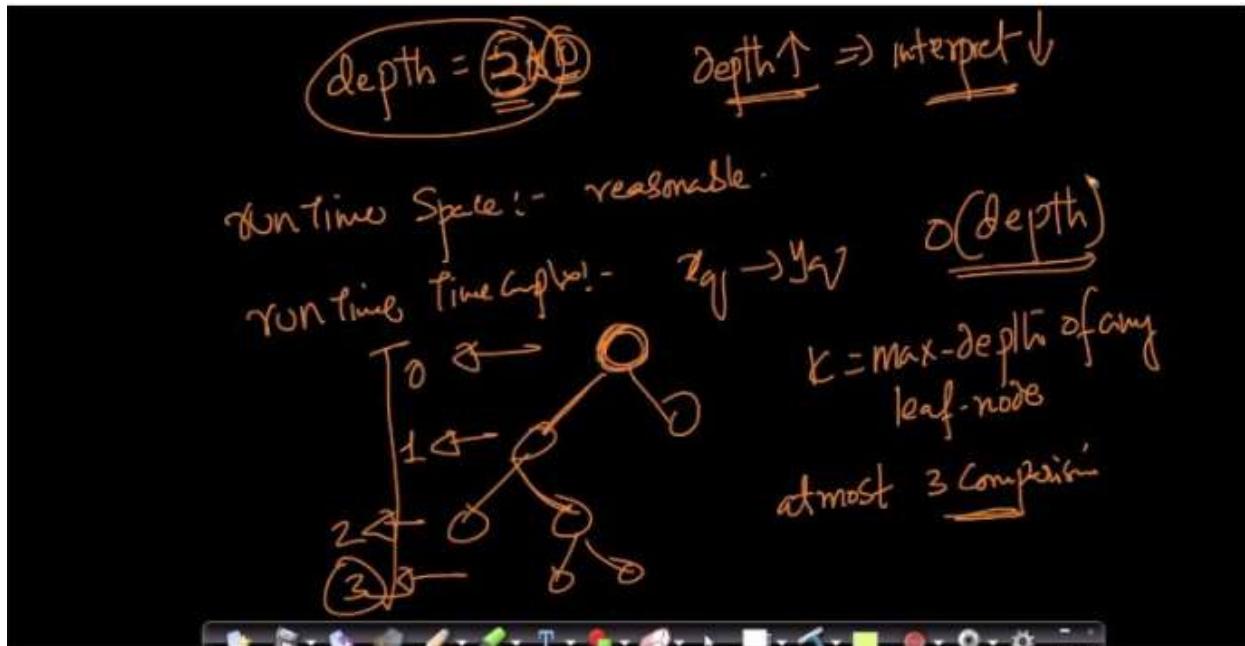
Decision trees are not very suitable for dataset having high dimension because of the d term present in the complexity.



Timestamp 3:20

Run time space complexity is $O(\text{nodes})$, where $\text{nodes} = \# \text{nodes}$

During run time we need the tree in memory to predict values. Now storing a decision tree isn't very space intensive. As we only need to store all the nodes i.e. internal nodes and leaf nodes. More often people convert decision trees into nested if else conditions and store them in memory.

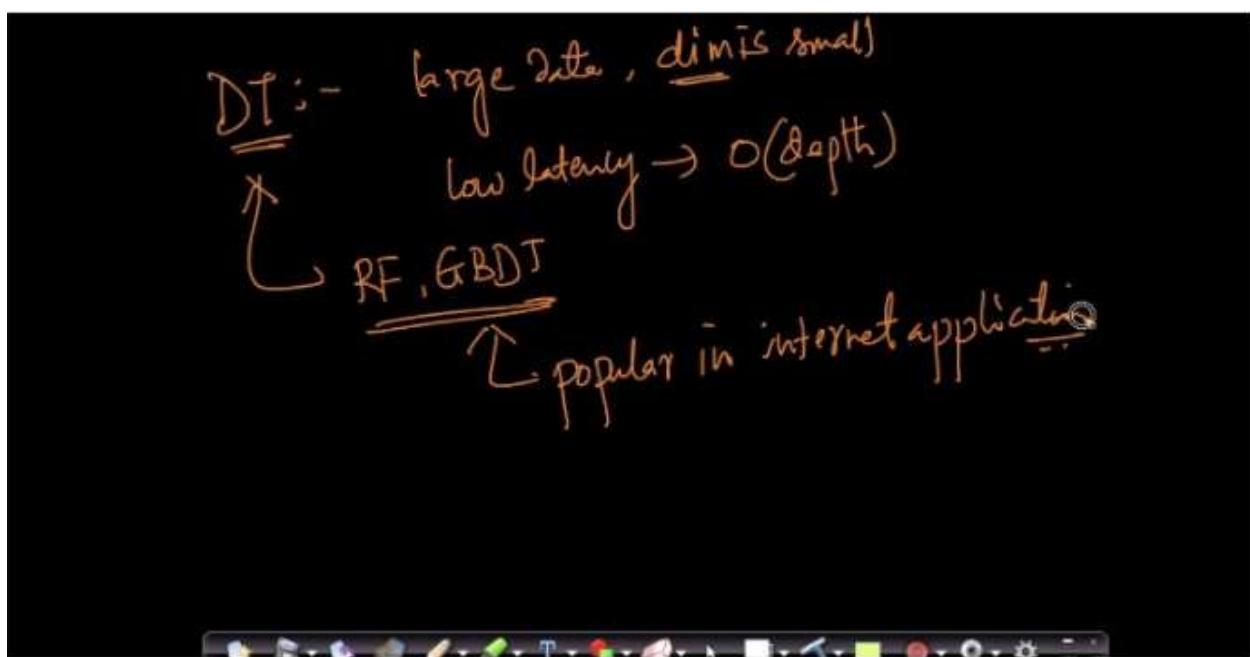


Timestamp 5:25

Runtime time complexity is $O(\text{depth})$, where depth = depth of the tree.

During runtime we just need to pass our query point x_q through the decision tree to get a output y_q i.e. $x_q \rightarrow y_q$. So for doing that in the worst case we need to traverse the depth of the tree.

Depth of the tree is defined as the longest path we can take from node to leaf node. Like in the image above the depth of the tree is 3. Depth of the root node is 0.

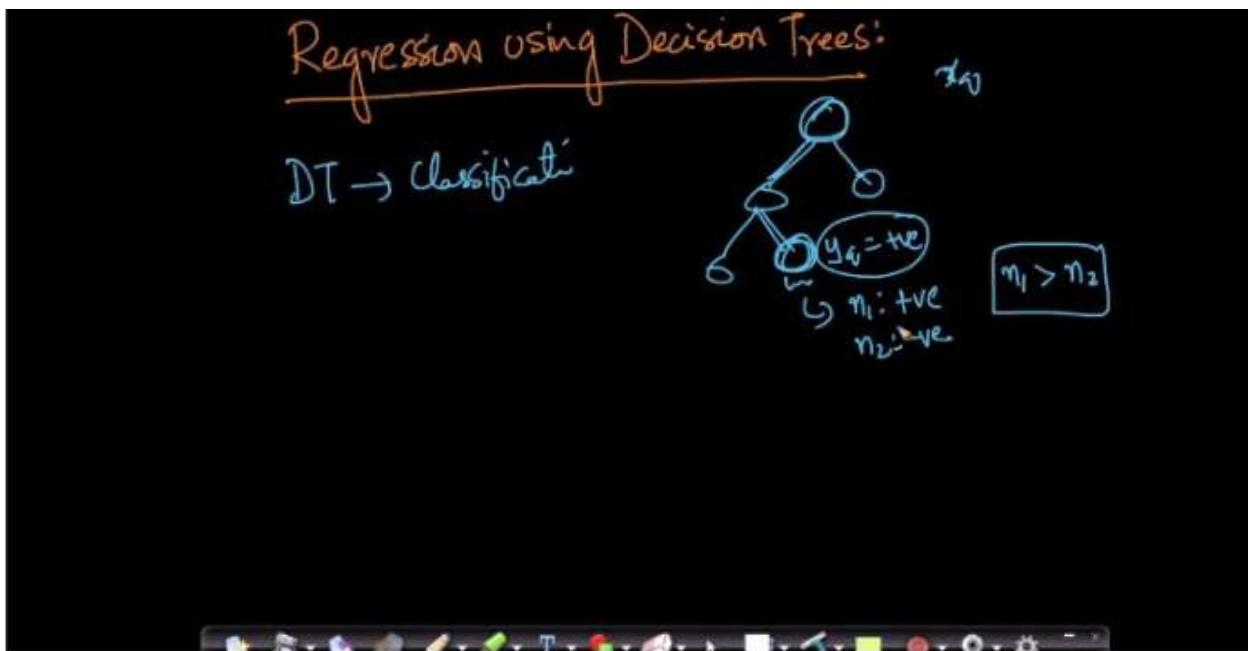


Timestamp 6:30

So, decision trees are extremely useful when we have large amounts of data, dimensionality is small.etc. Also it is extremely useful in cases where we have a requirement of low latency. As the run time complexity was only $O(\text{depth})$.

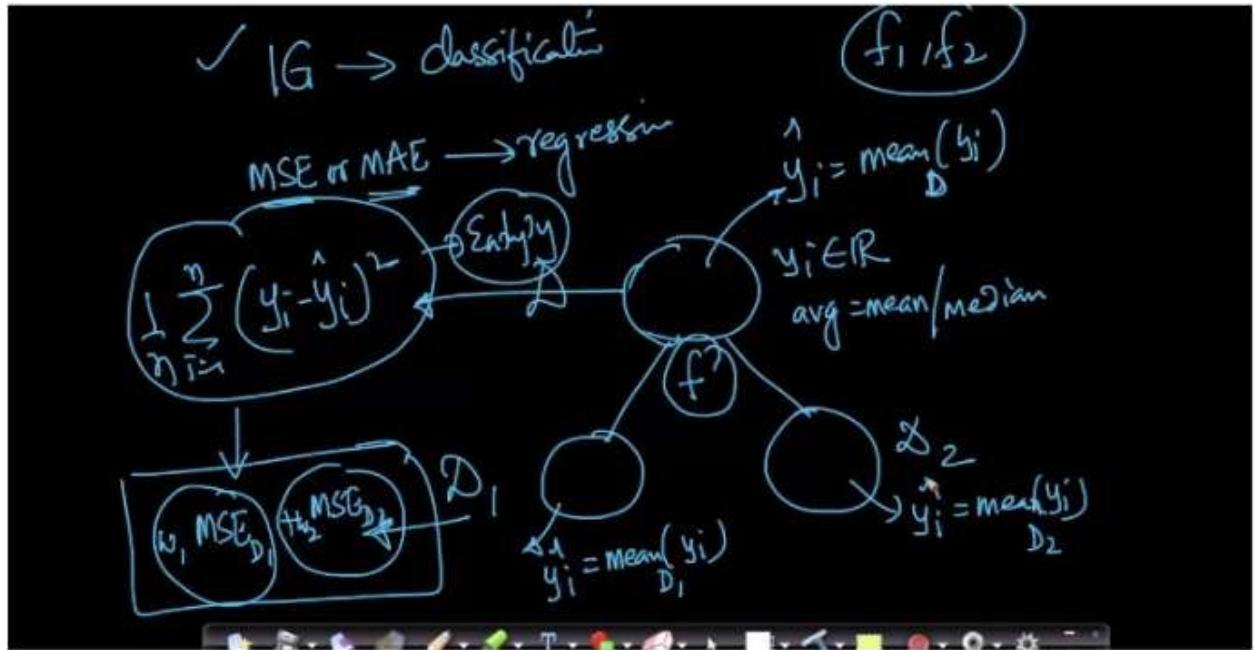
Later we will see variations of decision trees like Random Forests, Gradient Boosting Decision Trees.etc. These algorithms are very popular and are used in many internet companies.

37.13 Regression using Decision Trees



Timestamp 0:40

Till now we have seen decision trees for classification problems. We built trees using information gain as guidance. For prediction we traversed the tree and after reaching the leaf nodes we did a majority vote in the leaf node for our prediction.



Timestamp 3:37

Decision trees can also be used for regression problems. Infact, they work very well for regression problems. In classification problems we used information gain as criteria, in regression problems we use mean squared error(MSE) or median absolute error(MAE) as our criteria.

Suppose we have a dataset D with features f_1 and f_2 , with target variable $y \in \mathbb{R}$, now the mse of the dataset D can be calculated as,

$$MSE_D = 1/n * \sum_{i=1}^n (y_i - \hat{y}_D)^2$$

here $\hat{y}_D = \text{mean}(y_i)$

Now say we split our dataset into D_1 and D_2 , so we will calculate the mse for both these datasets and take the weighted sum of them, just like we did in classification and then take the difference between the parent mse and child nodes weighted mse.

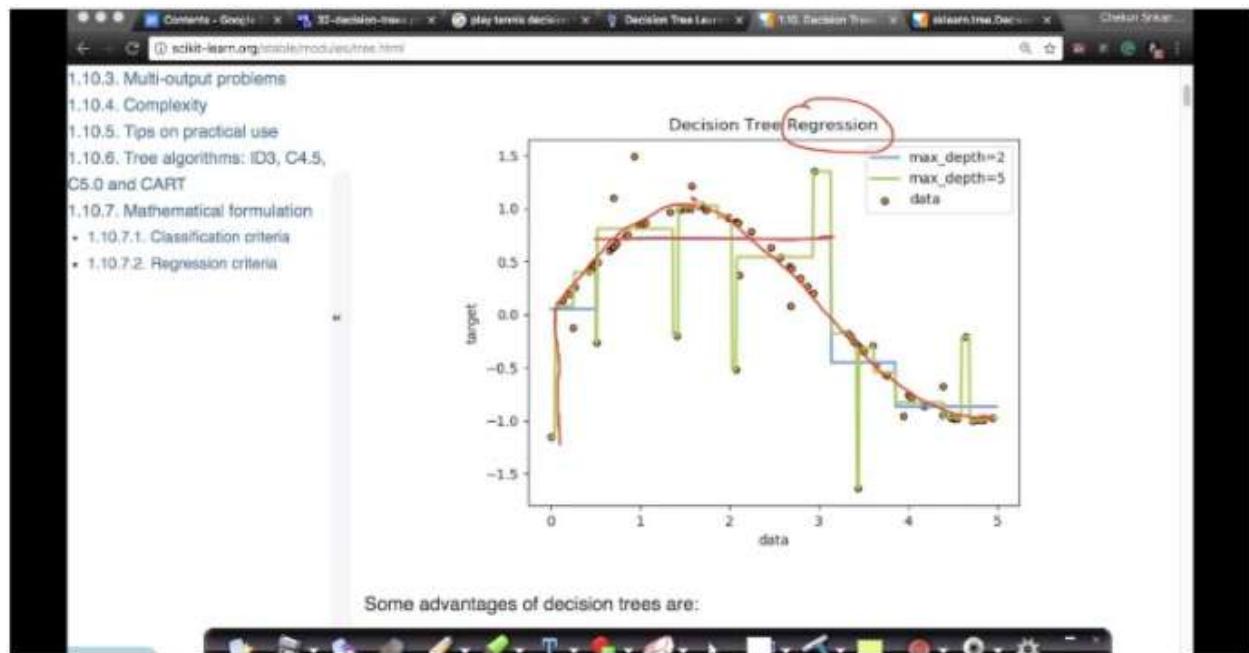
We do it for all the features and whichever feature reduces our mse most we select that feature just like we did with information gain.

Classif: f_i :- reducing entropy :- "0"
 Regn: f_i :- reduce MSE \rightarrow "0"
 $MSE(y_i) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ $MAE = \text{Median}(|y_i - \hat{y}_i|)$

Timestamp 5:32

So, like in classification we select features that reduce our entropy the most. Similarly in regression we select the feature that reduces MSE the most, both of them have a lowest value of 0.

We can also use MAE instead of mean, MAE is also robust to outliers.



Timestamp 8:39

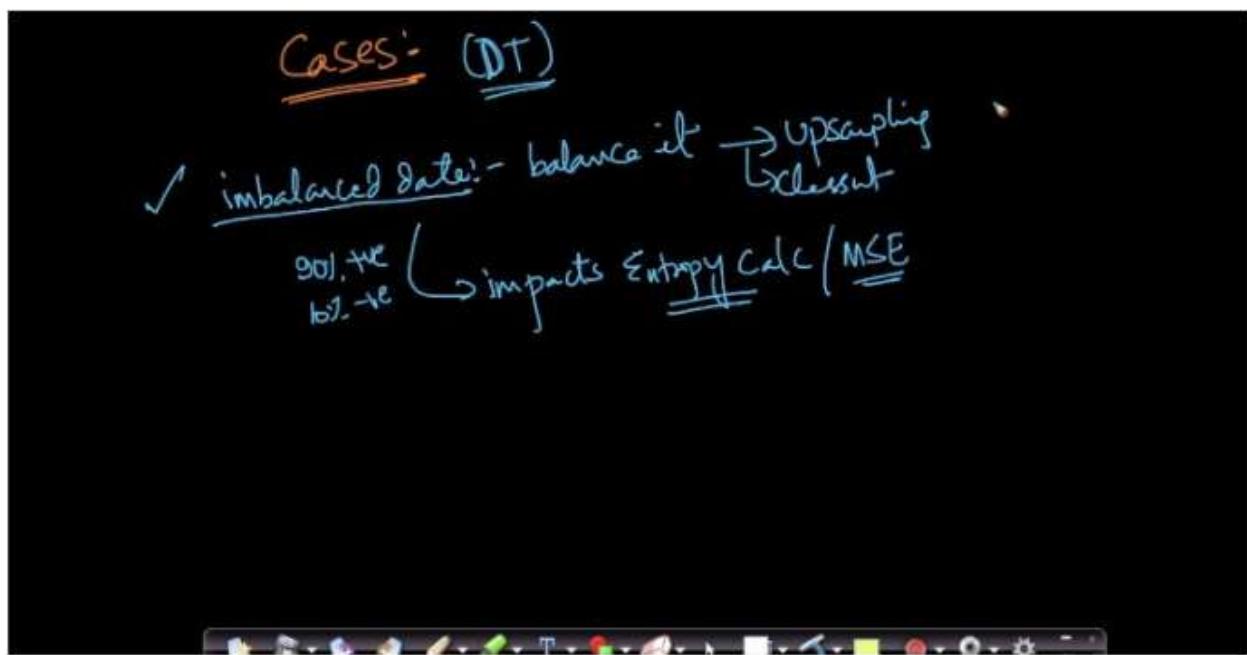
In the above diagram we have tried to show how a decision tree regressor behaves.

The red line with dotted points is our true function that we want to predict. Now we have fitted a decision tree regressor with max depth = 2 that can be seen in the blue line. We can see that it is underfitting.

Again we have fitted a decision tree regressor with max depth = 5 that can be seen with the green line. We can see that it is fitting even to the outliers in the data. It is overfitting.

The depth parameter behaves in a similar way in both regression and classification. Also if we notice we can see that all the lines are axis parallel just like we saw in classification.

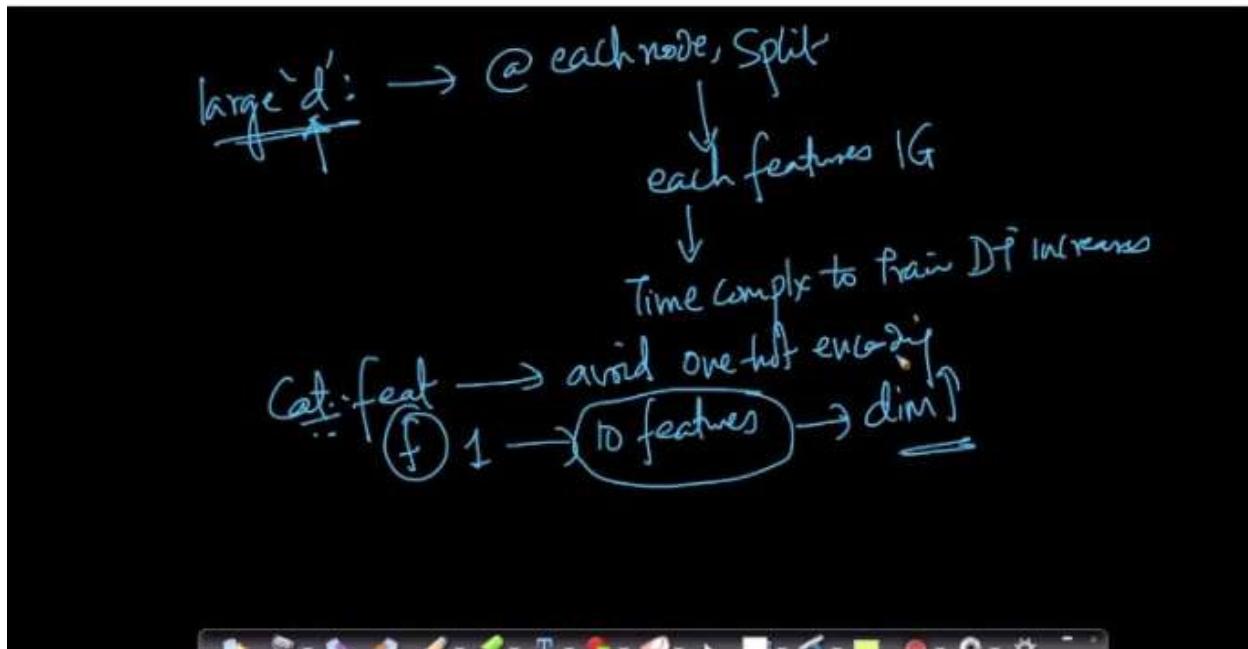
37.13 Cases



Timestamp 0:59

Let's look at various cases for our decision trees.

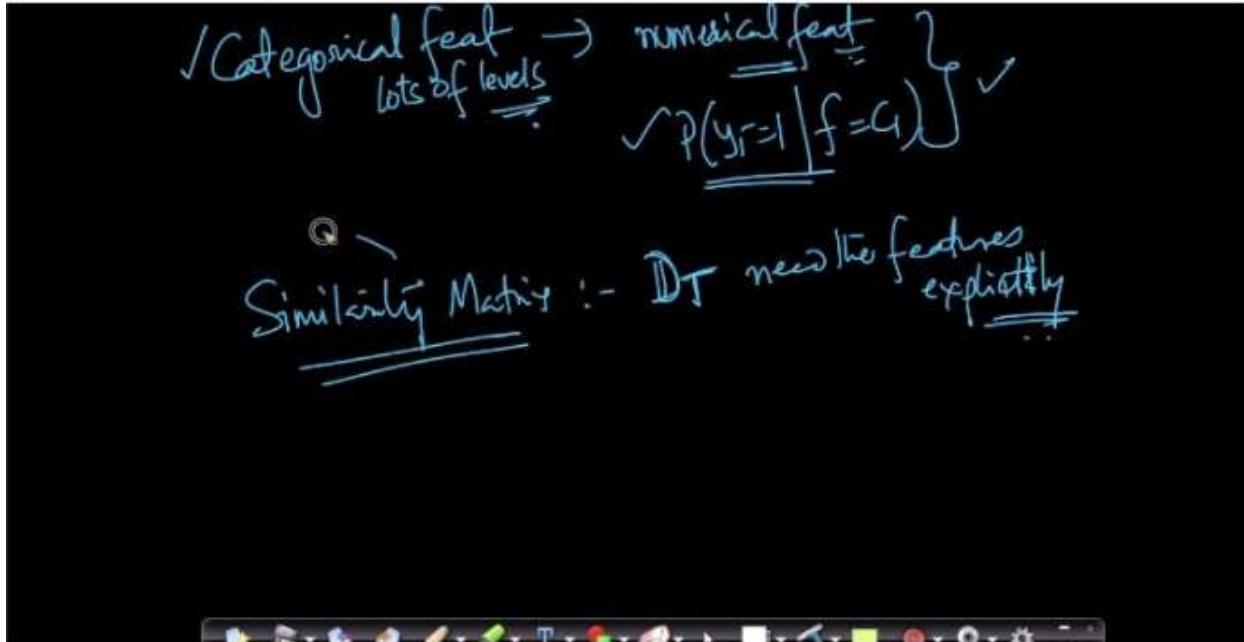
1. Imbalanced data: Decision trees are severely impacted by imbalanced data. It gets impacted because the entropy calculation or mse calculation is impacted by an imbalance dataset . We need to prevent it either by oversampling or undersampling.



Timestamp 2:22

2. Large 'd': If the number of dimensions increases then during each split we need to consider a large number of features and calculate their information gain/ mse. This can significantly increase the training time.

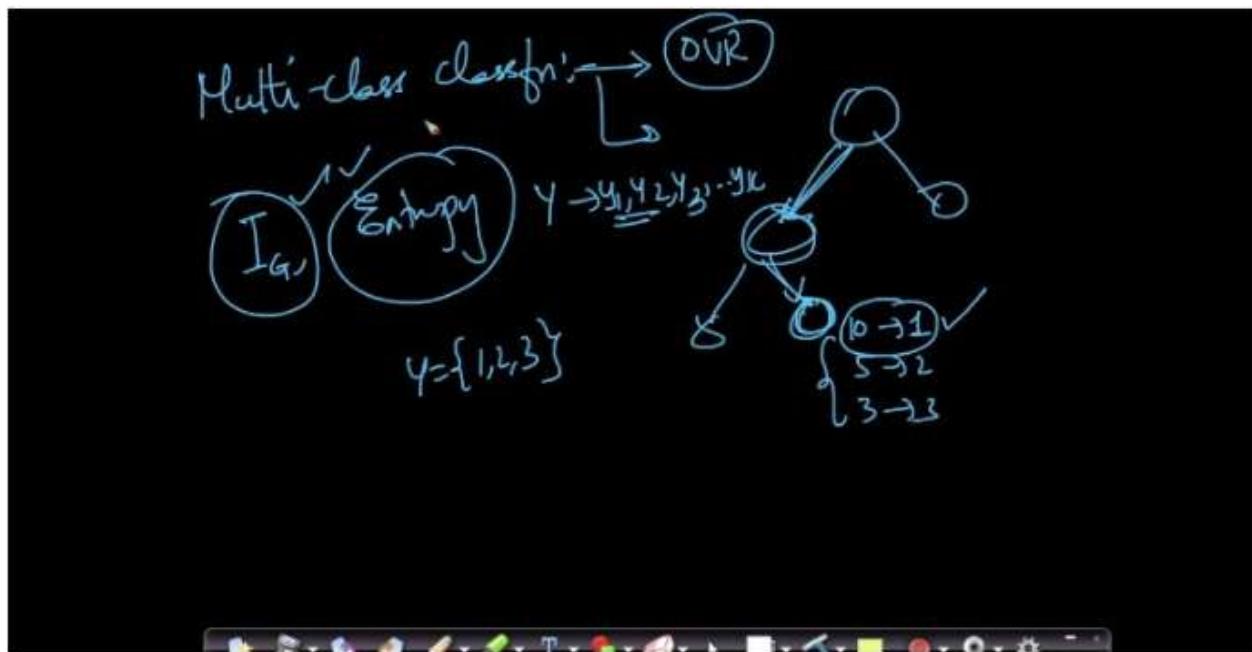
In decision trees we typically avoid doing One Hot Encoding(OHE) of categorical features if there are large no of categories, because it significantly increases the number of features for consideration leading to larger training time.



Timestamp 3:50

For categorical features having a large number of categories we typically convert them into numerical features as we have seen earlier.

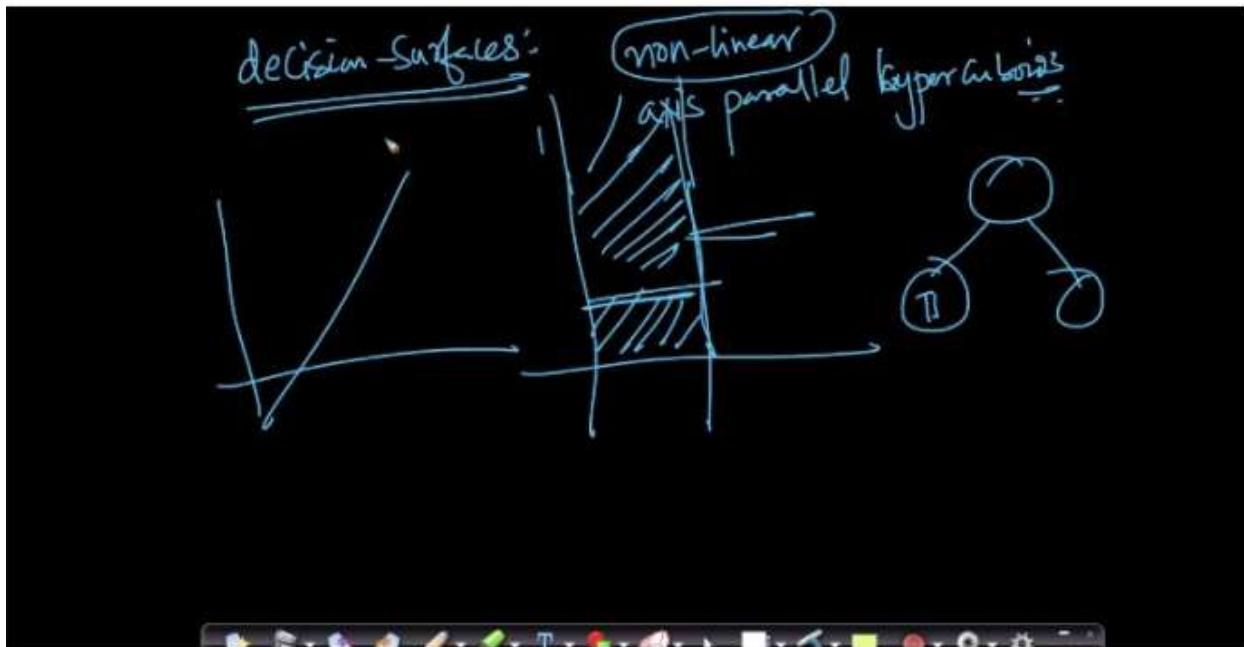
3. Similarity Matrix: Decision trees cannot work with similarity matrices because they need features explicitly to calculate entropy/mse.



Timestamp 5:13

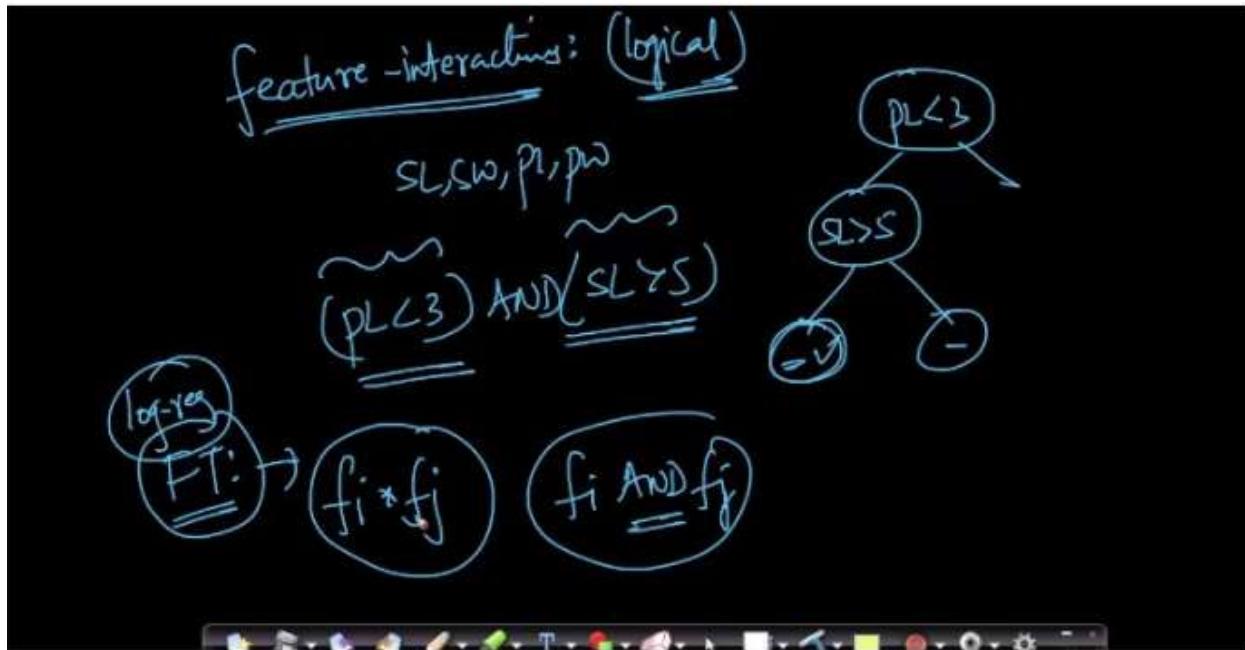
4. Multiclass Classification: Decision trees can naturally handle multiclass classification, we don't need one vs rest like techniques. Our criteria i.e. mse/entropy is applicable to random variables having more than two possible values. So they can be used effectively for multi class classification problems.

Now for the query point we can simply do a majority vote out of all possible classes as we did for binary cases.



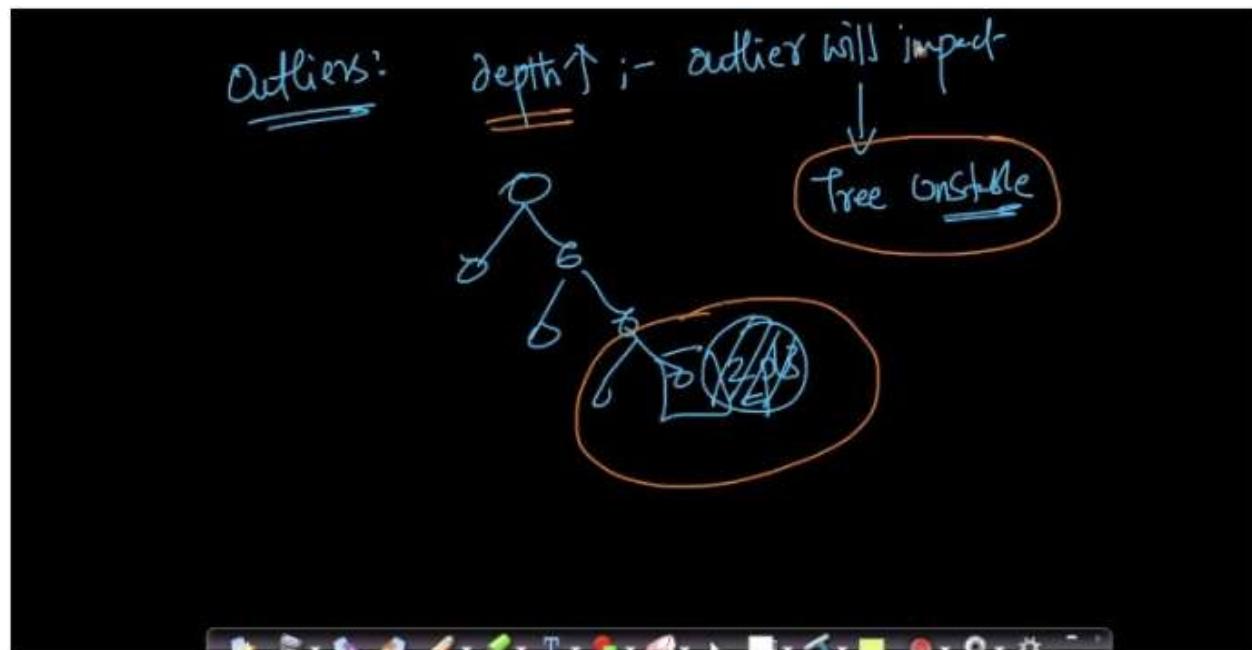
Timestamp 6:15

5. Decision Surfaces: Decision trees produce non-linear decision surfaces. As we have seen earlier, it produces axis parallel hyperplanes essentially breaking up the whole space into hypercubes and hyper cuboids.



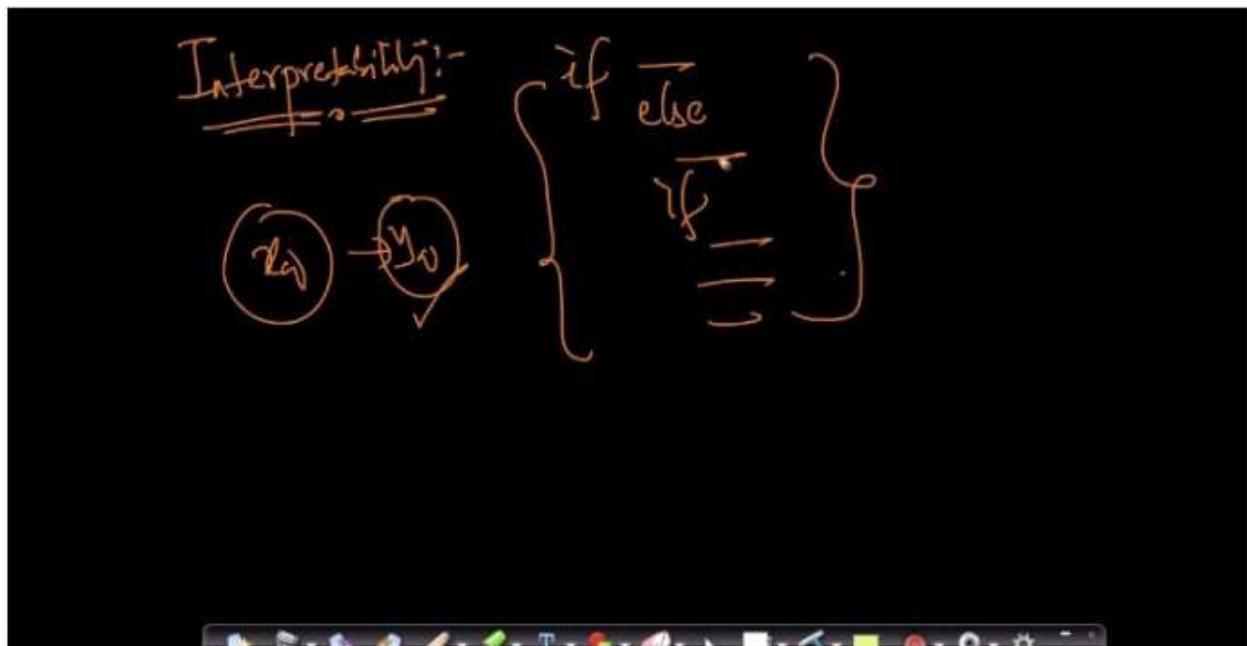
Timestamp 8:20

6. Feature Interactions: Decision trees have an inbuilt feature interaction mechanism. Say we have a tree as shown in the image above. Now, to reach the leftmost leaf we need to have a query point which has $PL < 3$ and $SL > 5$. Now these two features PL and SL together made an interaction for predicting a query. This was not possible in techniques like logistic regression where we needed to manually transform features.



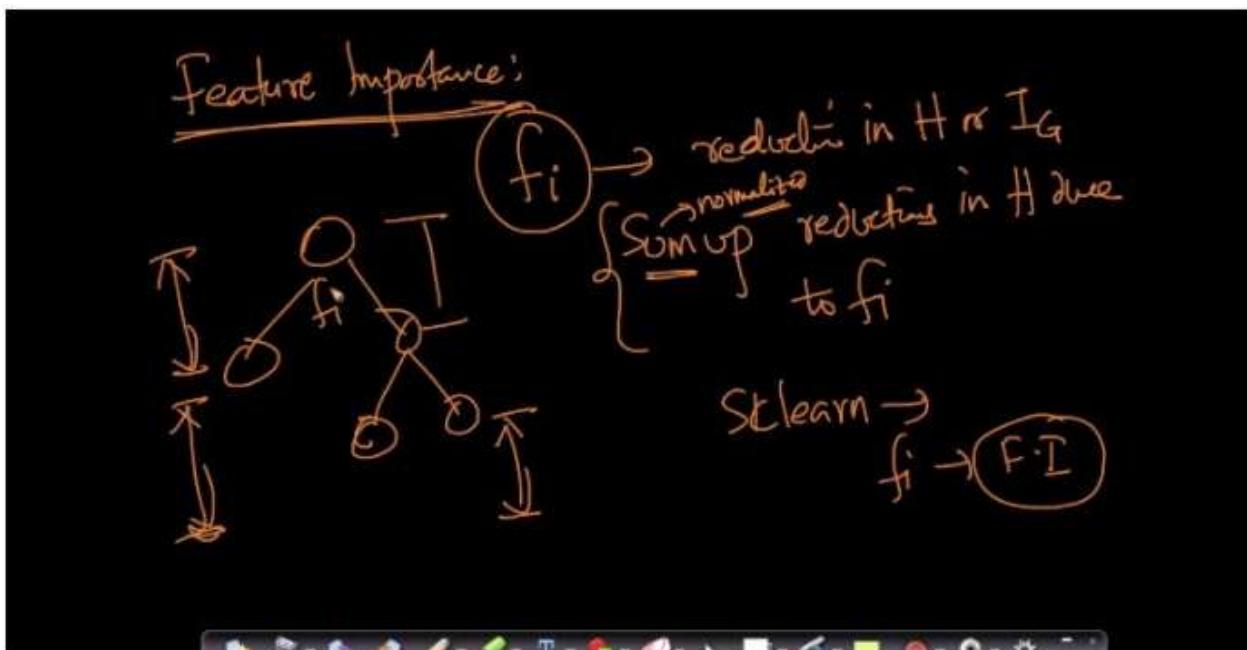
Timestamp 9:25

7. Outliers: Outliers can impact Decision trees especially if we have a tree of large depth. Trees of large depth can fit to outliers making it unstable.



Timestamp 10:11

8. Interpretability: Decision trees are highly interpretable if the depth is reasonable. Everything in a decision tree can be written in the form of nested if else statements. If the depth of the tree increases then interpretability reduces.



Timestamp 11:50

9. Feature Importance: Feature importance can be easily calculated in decision trees. The most important features will be the ones which cause the largest decline of entropy/gini impurity. Libraries like sklearn use this similar technique to calculate feature importance. It maintains a dictionary for each feature and calculates the total reduction caused by a feature in the tree.

37.14 Code Samples

We can see the scikit learn's implementation of Decision Tree here =>

<https://scikit-learn.org/stable/modules/tree.html>

38.1 Ensembles

Ensemble - Definition

A model which uses multiple models to obtain a better predictive performance than the performance obtained from any one of the constituent models is called an Ensemble model.

There are 4 types of mostly used ensemble models. They are

- a) Bagging
- b) Boosting
- c) Stacking
- d) Cascading

Using these ensembles, we can build very powerful and high-performance models.

The key aspect of all the ensemble models is “The more different the constituent models are, the better we can combine them.”

38.2 Bootstrapped Aggregation (Bagging) Intuition

Let us assume we are given a dataset 'D' of 'n' data points. Let it be denoted as ' D_n '.

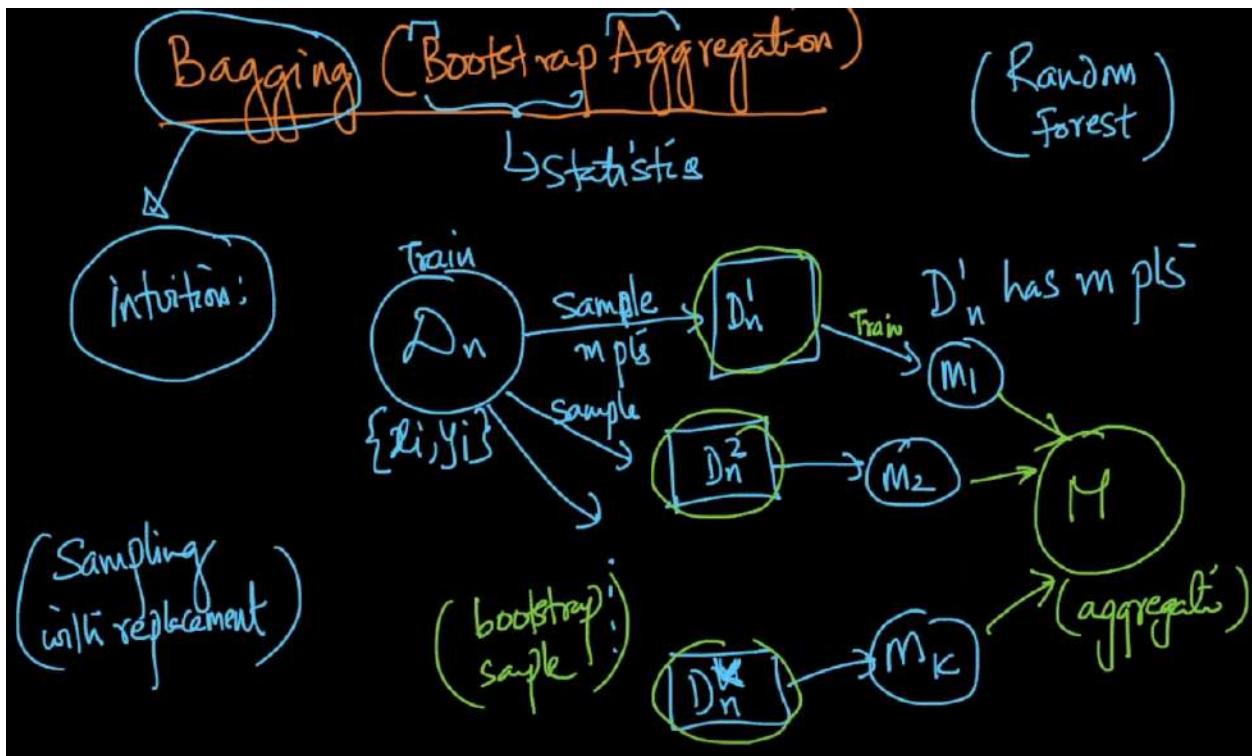
Let us pick 'k' samples of size 'm' each and let those samples be denoted as ' D_m^1 ', ' D_m^2 ', ' D_m^3 ', ..., ' D_m^k ' respectively.

$D_m^1 \rightarrow$ Sample with 'm' data points used to build a model ' M_1 '.

$D_m^2 \rightarrow$ Sample with 'm' data points used to build a model ' M_2 '.

$D_m^3 \rightarrow$ Sample with 'm' data points used to build a model ' M_3 '.

$D_m^k \rightarrow$ Sample with 'm' data points used to build a model ' M_k '.



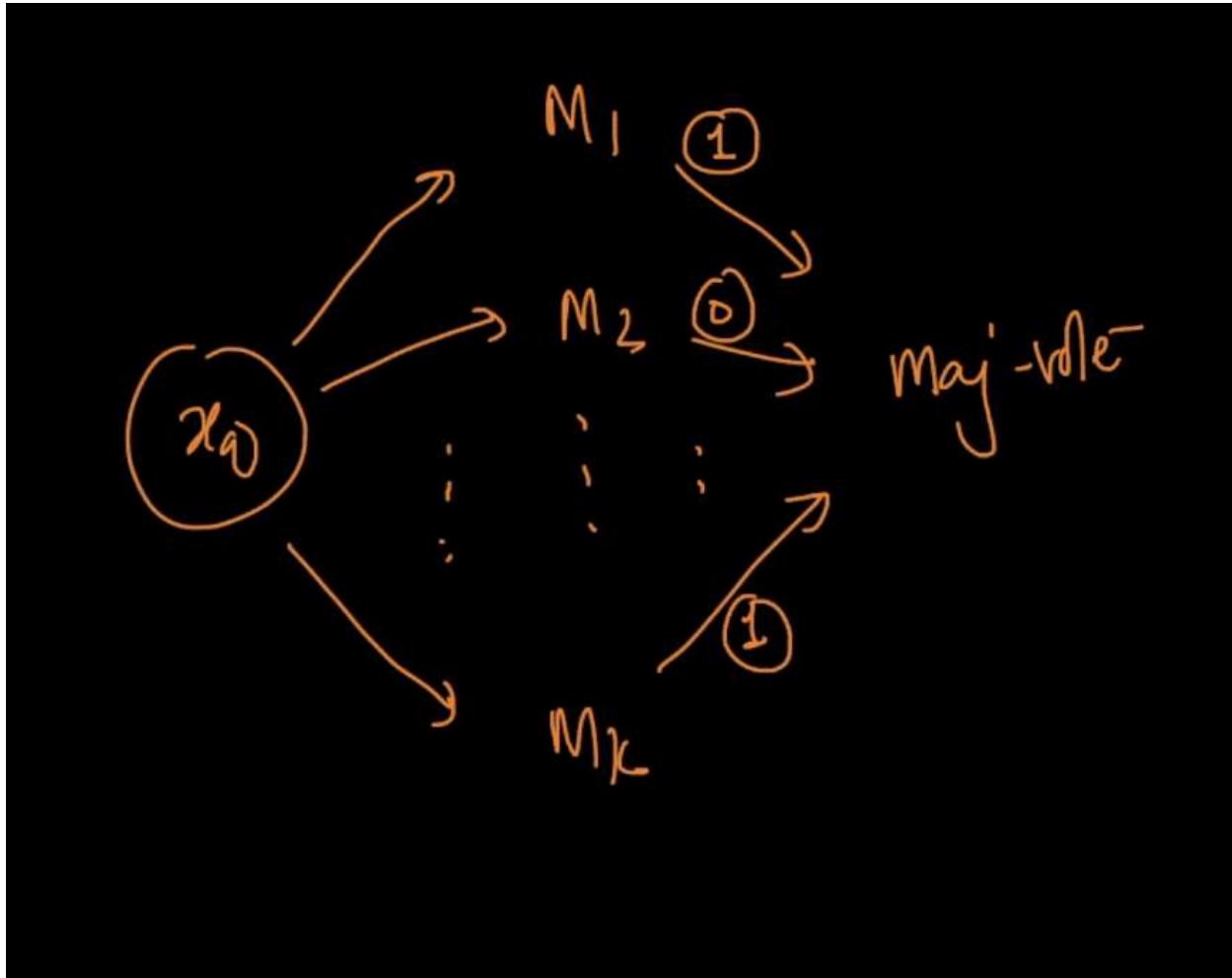
Here each model (ie., M_1 , M_2 , M_3 , ..., M_k) is built using random samples of size 'm' each($m < n$). Here it is sampling with replacement.

For every model ' M_i ' is built using ' D_m^i ' of size 'm'. It means each model ' M_i ' has seen a different subset of data.

All the samples created so far are called **Bootstrap samples**. Now all these models trained on the bootstrap samples must be combined into a larger model. This process is known as **Aggregation**.

A typical aggregation operation used is the **majority vote** (in the case of classification) and **computing the mean/median**(in the case of regression).

In the case of classification, each bootstrap sample of data is used in training the model and the majority vote is taken.



So if a query point ' x_q ' is given, it is passed through all the constituent models, and each model classifies this query point into one of the classes, and the majority vote will be taken among these predicted class labels and the result is assigned to this query point ' x_q '.

In the case of Regression, a query point ' x_q ' is passed through all the constituent models, and each model computes the output for this point. The mean/median of all these outputs is computed, and is assigned as the output value to the point ' x_q '.

Here is a key point in Bootstrapped Aggregation. If there is a change in the train dataset ' D_n ', then as the samples are picked randomly, the change in the samples will be less, thereby leading to the least amount of changes in the models trained by these bootstrap samples, and the final aggregation result is least affected.

This implies that Bagging can reduce variance in a model. Because of the way the model was designed, Bagging can reduce the variance in a model without impacting the bias. So the generalization error could be reduced as it is dependent on both the bias and the variance using the below equation

$$\text{Generalization Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

If each of these base models (ie., $M_1, M_2, M_3, \dots, M_k$) is a low bias, high variance model, then with the use of Bagging, we will continue to have low bias, but with a reduction in the variance. This reduction in the variance is because of the combination of **sampling** and **aggregation**.

Here even if there is a huge difference in ' D_n ' as we are randomly splitting the ' D_n ' into samples, only a few models get affected, thereby the variance gets affected.

So ultimately bagging says that if we take a bunch of high variance, low bias models and combine them using bagging (ie., bootstrap sampling + aggregation), then it gives a low bias, reduced variance model.

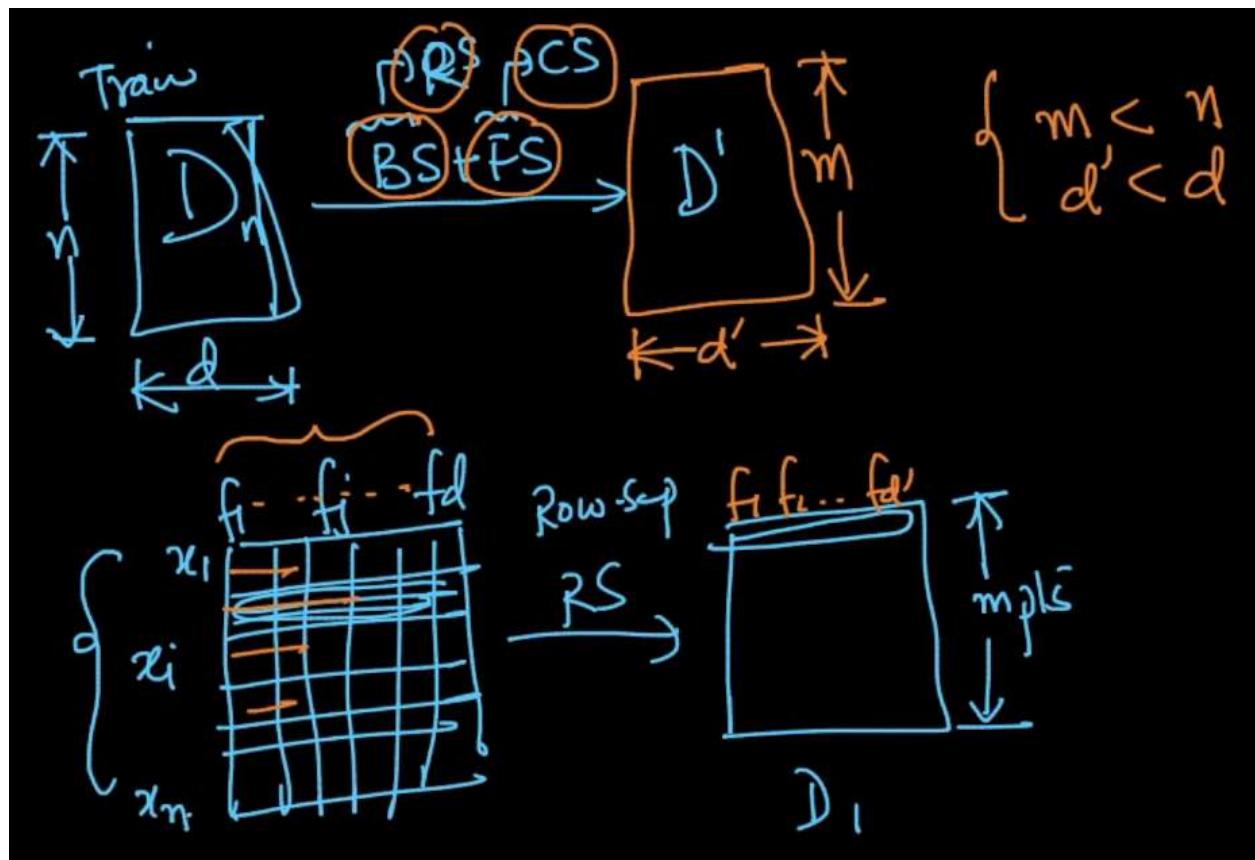
One of the examples with Low Bias, High Variance models is the Decision Tree with Large Depth. One of the very popular bagging techniques using the decision trees with large depth as the base models, and combining them is called **Random Forest**.

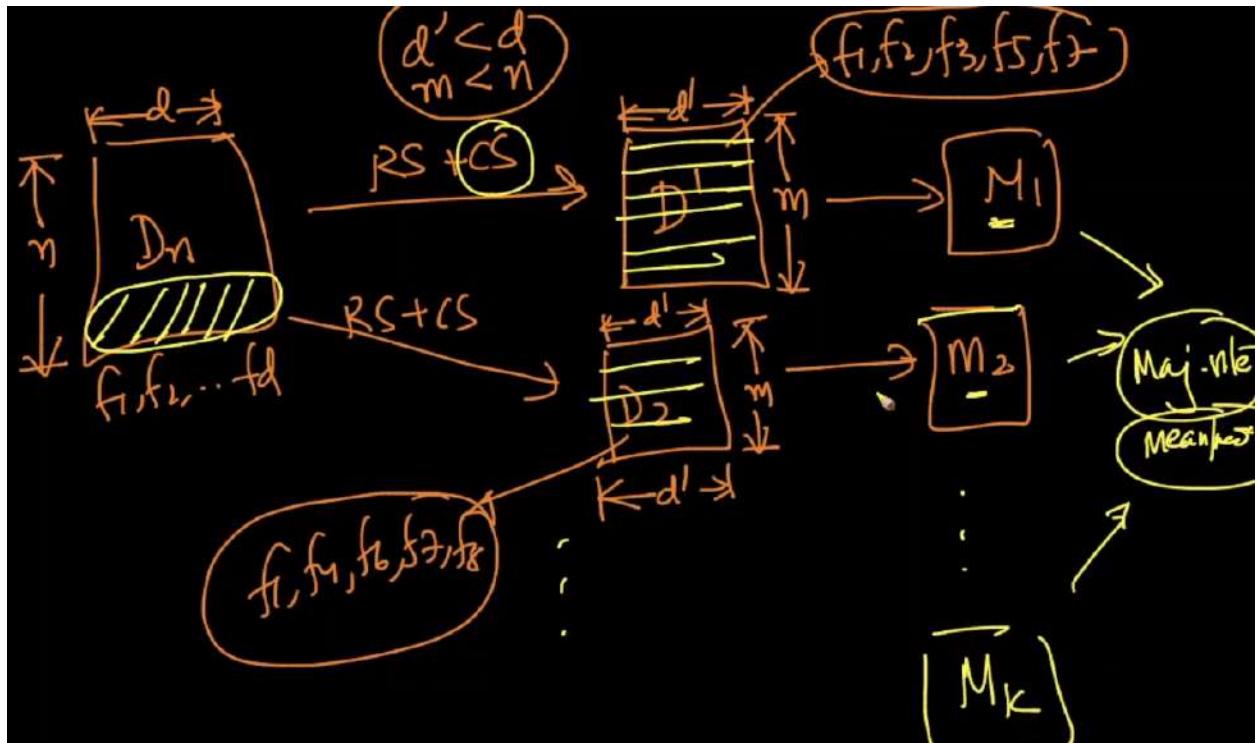
38.3 Random Forest and their construction

The word 'Random' in Random Forest means random sampling with replacement. The word 'Forest' means a group of trees. So
Random Forest = Decision Trees (as Base Learners) +
Bagging on top of the Base Learners +
Column Sampling (also called Feature Bagging)

Here Bootstrap sampling is also called Row Sampling. Feature Sampling is also called Column Sampling.

Let us assume we have a dataset ' D_n ' with ' n ' data points and ' d ' dimensions.

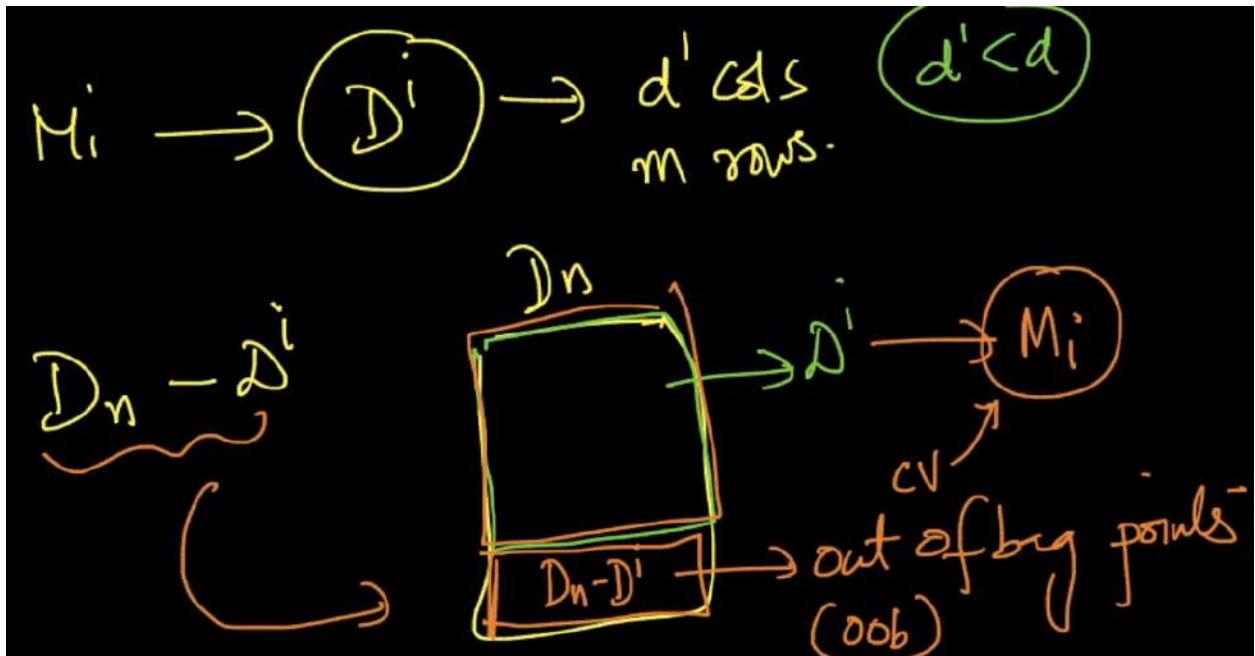




We have to pick a sample of 'm' points($m < n$) separately and of ' d' dimensions($d' < d$), and use it in training the intermediate models.

In Random Forest, all the intermediate models M_i 's are the decision trees of reasonable depth. The data points and the features in each intermediate model are not the same. As the datasets used for training these intermediate models are different, the models also will be different.

The Base Learners in Bagging should be the Low Bias, High Variance models. Hence we are choosing Decision Trees as the base learners in Random Forest. So in Random Forest, due to Bootstrap Sampling and Aggregation, the variance gets reduced. Each intermediate dataset contains different data points and also different features.



Out of the dataset ' D_n ', when random sampling is used and ' D_m ' dataset is extracted, this intermediate dataset can be used to train the base learning model. And the remaining data points (ie., $D_n - D_m$) are used for cross-validation on the intermediate model and result in the out-of-bag error.

$D_m \rightarrow$ In Bag Samples

$D_n - D_m \rightarrow$ Out of Bag Samples

38.4 Bias-Variance Tradeoff

Random Forests tend to have Low Bias because their Base Learners are the models with Low Bias to start with. The bias of the overall model is the same as the bias of the individual base learners.

Model (M) = Aggregation($M_1, M_2, M_3, \dots, M_K$)

If the number of Base learners increases, the variance decreases.

If the number of Base learners decreases, the variance increases.

So ' K ' is a hyperparameter that has to be tuned carefully.

We have another 2 hyperparameters. They are Column Sampling Ratio and Row Sampling Ratio.

Column Sampling Ratio(CSR) = d^l/d

Row Sampling Ratio(RSR) = m/n

As the row sampling ratio and the column sampling ratio decrease, the in-bag data sets become more and more uncorrelated, and by using such datasets for training the base learners, the variance of the aggregate model will decrease.

So we have 3 hyperparameters here to tune(ie., ' K ', 'RSR', and 'CSR'). But in general, 'CSR' and 'RSR' are kept constant, and only ' K ' is used in cross-validation.

Q) How does the variance of the overall model decrease with an increase in ' K '?

Ans) When the number of base learners is small(say $K=1$), then we have one base model which is highly overfitted. So the overall model with $K=1$ is overfitted.

As ' K ' increases, we have many models built on various slices of data each learning a different aspect of the data. So the overall model after the majority vote is less likely to overfit than a single model, as each model is slightly different from the others.

38.5 Bagging - Train and Run time complexity

In Random Forest with 'K' base learners

Train Time: $O(n \log(n) * K * d)$

But when we have multi-cores we can build each learner on each core. We call it **Trivially Parallelizable**.

When we have large amounts of data with a reasonable number of features, we can build trees faster. But if the dimensionality is high, then it becomes difficult.

Runtime: $O(\text{depth} * K)$

The depth in the base learners of the Random Forest model should be large(roughly 10 to 20).

Space Complexity: $O(\text{Decision Trees structure} * K)$

It means we are storing the entire decision tree structure of all the 'K' base learners.

Note

As the video lecture 38.6 is on the code discussion, we aren't giving any notes on it. We suggest you go through the video lecture, and if you have any queries, please feel free to post them in the comments section.

38.7 - Extremely Randomized Trees

In Random Forests, we have seen aggregation and randomization in Row and Column Sampling. But in extremely randomized trees, we have one more level of randomization. That is randomization in selecting the numerical feature values as the threshold for performing a split, in the base learners.

In Decision Trees and Random Forests, in order to perform a split on the basis of the values of the numerical features, we first sort the values and consider each value as the threshold, and compute the Information gain. That particular threshold at which we get the maximum information gain is used in performing a split at that level in the decision tree.

In Extremely randomized trees, instead of considering all the values as the threshold, only a few values are picked randomly as thresholds, and information gain is computed for each threshold. Whichever threshold is responsible for the maximum information gain, would be chosen as the optimal value to perform a split.

Random Forest = Row Sampling + Column Sampling + Aggregation

Extremely Randomized Trees =

Row Sampling + Column Sampling + Aggregation + Randomization in selecting the thresholds from real values features.

Even in Random Forests, we are using Randomization in row and column samplings, as a way to reduce variance. In extremely randomized trees, we use randomization in row and column samplings and selecting a threshold value. Extremely Randomized trees tend to reduce variance better than random forests, but the bias might slightly increase. Extremely randomized trees are much more useful when we have real-valued features.

Q) How do extremely randomized trees reduce more variance when compared to random forests?

Ans) The base learners in Random Forests have higher chances of overfitting(ie., high variance) when compared to the base learners of extremely randomized trees, as the extremely randomized trees are using only a subset of possible values.

Both random forests and extreme trees use majority voting to reduce the overfitting, but the final model of extreme trees tends to be less overfit, as their base learners are relatively less overfitting when compared to the base learners of random forests.

Note

Extremely randomized trees can also handle categorical features as we can pick the random splits from a set of discrete values that the feature can take.

If a categorical feature has very few categories, we need not worry about the time complexity of evaluating all the possible values for a split. In such a case, extremely randomized trees work the same way as random forests.

Extremely randomized trees are very helpful when we have many real-valued features (or) categorical features with many categories.

38.8 Random Forest: Cases

Random Forest = Decision Trees (as base learners) + Row Sampling + Column Sampling + Aggregation

High Dimensionality

The decision trees cannot handle high dimensionality. Also, the decision trees are not recommended for use, when there are categorical features with a huge number of categories.

All the cases of the decision trees are applicable for random forests also, except a few. In all those cases where Decision Trees are not recommended for use, the Random Forests are also not recommended for use.

Bias-Variance Tradeoff

The Bias-Variance Tradeoff of Random Forest is not the same as that of the decision trees. In decision trees, the bias-variance tradeoff depends on the depth of the tree, whereas in random forests, the bias-variance tradeoff depends on the number of base learners.

Feature Importance

In Decision Trees, a feature ' f_i ' is said to be important on the basis of the overall reduction in entropy (or) Gini impurity because of this feature at various levels in the decision tree.

In Random Forest, a feature ' f_i ' is said to be important on the basis of the overall reduction in entropy (or) Gini impurity at various levels of each of the base learners. (ie., if a feature is important in most of the base learners, then it is important in the whole random forest model).

Except in these two cases, in the rest of the cases, both Random Forests and Decision Trees work the same.

38.9 - Boosting Intuition

In the case of Bagging, we have seen

Bagging = (High Variance, Low Bias Base models) + Randomization(in Column and row sampling and picking the threshold in ERT) + Aggregation

Here in the case of Boosting,

Boosting = (Low Variance, High Bias Base models) + Additive Combining (Reducing Bias while keeping the variance low)

Our main intention is to reduce the generalization error either using Bagging or Boosting. Same as Bagging, Boosting also allows us to build both Regression and Classification models.

Core idea of Boosting

Let us assume, we have the training data set $D_{Train} = \{x_i, y_i\}_{i=1}^n$.

Here the base learners are the High Bias and Low Variance models. One example of such a model is a Decision Tree with shallow depth. As these base learners have a high bias, the training error will be high.

Stage-0

Train the base learner ' M_0 ' with ' D_{Train} '. For every point ' x_i ', compute $error_i = y_i - h_0(x_i)$

Stage-1

Now consider a base learner ' M_1 ' and the training data for it will be $\{x_i, error_i\}_{i=1}^n$. Let the output of the model ' M_1 ' be $h_1(x)$.

Let $F_1(x)$ be the model at the end of stage 1.

$F_1(x) = \alpha_0 h_0(x) + \alpha_1 h_1(x) \rightarrow$ (weighted sum of 2 base models $h_1(x)$ and $h_0(x)$)

Stage-2

Here we have to train the model ' M_2 ' with $\{x_i, error_i\}_{i=1}^n$. Here $error_i = y_i - F_1(x_i)$

The model at the end of stage 2 is

$$F_2(x) = \alpha_0 h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x) \rightarrow (\text{weighted sum of 3 base models } h_2(x), h_1(x) \text{ and } h_0(x))$$

Similarly, at the end of stage 'K', the final model is

$$F_k(x) = \sum_{i=0}^k \alpha_i h_i(x)$$

This model $F_k(x)$ is the additive weighted model.

$h_i(x)$ is trained to fit the residual error at the end of the previous stage.

So every base learner $h_i(x)$ is trained on $\{x_i, \text{error}_i\}_{i=1}^n$ to fit the residual error at the end of the stage (i-1).

The final model $F_k(x)$ ends up having a low residual error. As the residual error reduces, the bias of the model also decreases. Hence the objective of Boosting is fulfilled.

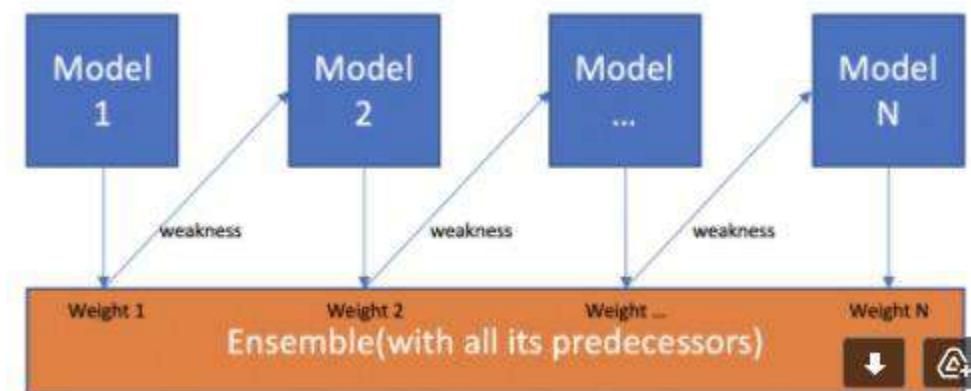
The most commonly used Boosting algorithms are

- a) Gradient Boosting Decision Tree (GBDT)
- b) Ada Boosting (Also known as Adaptive Boosting).

Ada boost is mostly used when we are working with the images data.

Note

Model 1,2,..., N are individual models (e.g. decision tree)



We cannot train all the base learners in Boosting in parallel, like in Bagging. We are trying to predict y_i 's in the first model, and each subsequent model is trying to fix the errors by combining the previous models.

Boosting could easily overfit. Hence we use either of the two tricks to avoid overfitting.

- a) Using High Bias base learners
- b) Applying Regularization by shrinkage

Boosting could overfit if the number of base learners is high. We can avoid overfitting even when 'K'(number of base learners) is large by reducing α_i 's. (where α_i 's are the weights given to each model)

38.10 - Residuals, Loss Functions, and Gradients

In Boosting, we have seen the final model with $K+1$ base learners, and the final model at the end of stage 'K' is given as

$$F_K(x) = \sum_{i=0}^K \alpha_i h_i(x)$$

The residual at the end of stage 'K' is given as

$$\text{error}_i = y_i - F_K(x)$$

To solve any problem of Machine Learning, we have to minimize the loss function using an optimization problem. The loss function is dependent on the class labels y_i 's and the final model we had at the end of stage 'K'. (ie., $F_K(x)$)

Let us assume we are working on a regression problem using Boosting. Then the loss function is given as

$$L(y_i, F_K(x_i)) = (y_i - F_K(x_i))^2$$

If the loss function 'L' is of the form $(y_i - z_i)^2$, then

$$\frac{\partial L}{\partial z_i} = \frac{\partial}{\partial z_i} (y_i - z_i)^2 = 2*(y_i - z_i)*(-1) = -2*(y_i - z_i) \text{ (where } z_i = F_K(x_i))$$

So now the value of $\frac{\partial L}{\partial F_K(x_i)}$ is given as

$$\frac{\partial L}{\partial F_K(x_i)} = -2*(y_i - F_K(x_i))$$

Let us send the negative sign to the left-hand side, then the equation becomes

$$-\frac{\partial L}{\partial F_K(x_i)} = 2*(y_i - F_K(x_i)) \rightarrow (1)$$

$\frac{\partial L}{\partial F_K(x_i)}$ → Gradient

$(y_i - F_K(x_i))$ → Residual/Error

So from equation (1), if we exclude the number '2', we can say that Negative Gradient is nearly equal to the residual.

(ie., $-\frac{\partial L}{\partial F_K(x_i)} \approx (y_i - F_K(x_i))$). Hence the negative gradient is also called **pseudo-residual**.

So when we train a base learner at the end of the stage 'i', then

$$\text{error}_i = (y_i - F_{i-1}(x))$$

So in the optimization, instead of using the residual, we can use the negative gradient(which is the pseudo residual). The main reason for using pseudo-residual, in place of residual is, if we use residual we can minimize any loss function, as long as it is differentiable. Because of this, the Boosting algorithms are so powerful.

Models like Random Forests could not minimize all types of losses(because it can only minimize the entropy loss), whereas Boosting models like Gradient Boosting, can minimize any loss, just because we are using the pseudo residuals, in place of normal residuals.

38.11 - Gradient Boosting

Let the training data be $\{(x_i, y_i)\}_{i=1}^n$.

'M' → number of iterations (ie., the number of base learners).

Let $L(y, F(x))$ → Differential loss function

Steps of Algorithm

- 1) Initialize the model with a constant value.

$$F_0(x) = \arg\min_y \sum_{i=1}^n L(y_i, y)$$

Here we have to find a constant 'y' that could minimize $\sum_{i=1}^n L(y_i, y)$.

Here 'y' that minimizes $F_0(x)$ is the mean of y_i 's.(In case of regression)

- 2) For $m = 1$ to M ,

- a) Compute the pseudo residuals,

$$r_{im} = -[\partial L(y_i, F(x_i))/\partial F(x_i)]_{F(x)=F_{m-1}(x)} \quad (\text{for } i = 1, 2, 3, \dots, n)$$

r_{im} → Pseudo residual for the m^{th} stage of the i^{th} point.

- b) Fit a base learner $h_m(x)$ to pseudo residuals (ie., train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$)

- c) Compute multiplier ' γ_m ' by solving the following one dimensional optimization problem.

$$\gamma_{im} = \arg\min_y \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

- d) Update the model

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

- 3) Output $F_M(x)$

Note

One of the great properties of Gradient Boosting is that we can pick any differentiable loss function(L) and any base model $h_i(x)$ and still be able to optimize the loss function using Gradient Boosting. To achieve this, we need to introduce the loss function in the construction of ' γ_{im} '.

The beauty of Gradient boosting is that the base model could be any model(Decision Trees, Logistic Regression or SVM) and is independent of the loss function. Shallow Decision Trees with high bias are a very popular option making them GBDTs. The loss function is independent of the base model we use, and it should be differentiable.

38.12 - Regularization by Shrinkage

The formulation of the final model in Gradient Boosting is given as

$$F_M(x) = h_0(x) + \sum_{m=1}^M \gamma_m h_m(x) \quad (\text{where } M \rightarrow \text{Number of base models})$$

As the number of base models increase, variance tends to increase, bias tends to decrease and leads to overfitting.

Shrinkage

Earlier we have seen the model at the end of stage 'M' is given as

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

In the case of Shrinkage, the above formula is slightly modified. A new parameter 'v' gets multiplied to ' $\gamma_m h_m(x)$ ' in the above equation and the new equation becomes

$$F_m(x) = F_{m-1}(x) + v\gamma_m h_m(x) \quad (\text{where } 0 < v \leq 1)$$

In shrinkage, we have the learning rate which is denoted as 'v' as a hyperparameter.

If 'v' increases, then the model overfits. It is better to use Grid Search Cross-Validation and find the optimal values of 'M' and 'v' for building GBDT.

Here shrinkage(v) is a hyperparameter that helps us avoid overfitting. It is one of the hyperparameters that we can tune to avoid overfitting of Gradient Boosting models, which are prone to overfitting. It reduces the output generated by each of the base models through multiplication as $0 < v \leq 1$, thereby avoiding the base models to overfit on the train data.

Note

' γ ' is the weight we give to the M^{th} model and this allows us to give weight to each base model based on how useful that base model is in reducing the overall loss. This can be thought of as weights similar to those weights we give to each feature in the case of Logistic Regression. The weights here are given to the base models, but not to the individual features. The weights (γ_m) are determined by solving an optimization problem at each iteration of Gradient Boosting.

38.13 - Train and Run time complexity

Train Time:

For one Decision Tree: $O(n \log(n) d)$

For 'M' base learners: $O(n \log(n) d M)$

GBDT is not easily parallelizable like Random Forest. GBDT takes more time to train, than Random Forest, though the time complexity is the same.

Runtime:

For one Decision Tree: $O(\text{depth})$

For 'M' base learners(for $h_m(x)$): $O(\text{depth} * M + m)$

Space Complexity:

$O(\text{storing each tree} + \gamma_m)$

Note

As the video lecture 38.14 is on the documentations of Gradient Boosting Classifier and XGBoost models, we are not providing any notes for it. Please go through the video lecture, and if you have any queries, please feel free to post them in the comments section under the video.

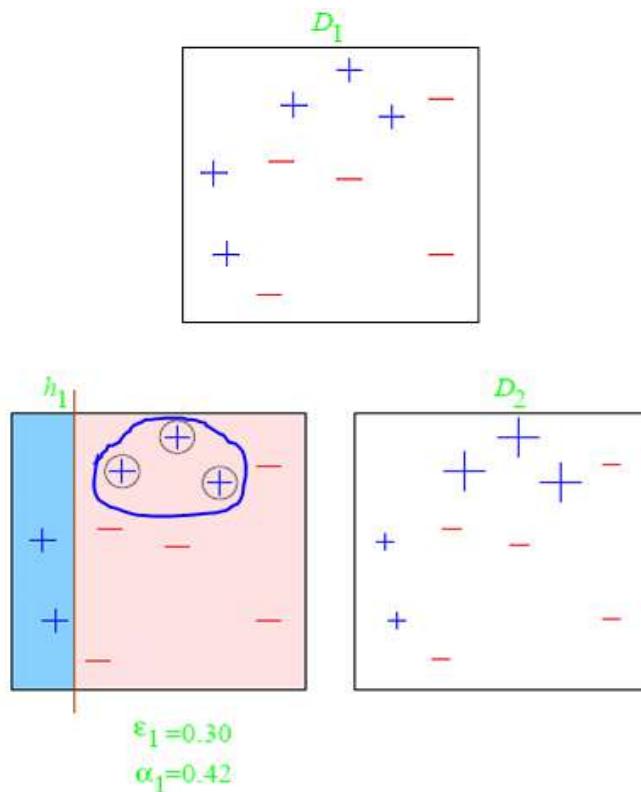
38.15 - AdaBoost: Geometric Intuition

AdaBoost is one of the Boosting algorithms. It is similar to Gradient Boosting, but there are a few differences between them. AdaBoost is typically mostly used in image processing applications and computer vision based tasks.

Please go through the link below for the example that was discussed here.

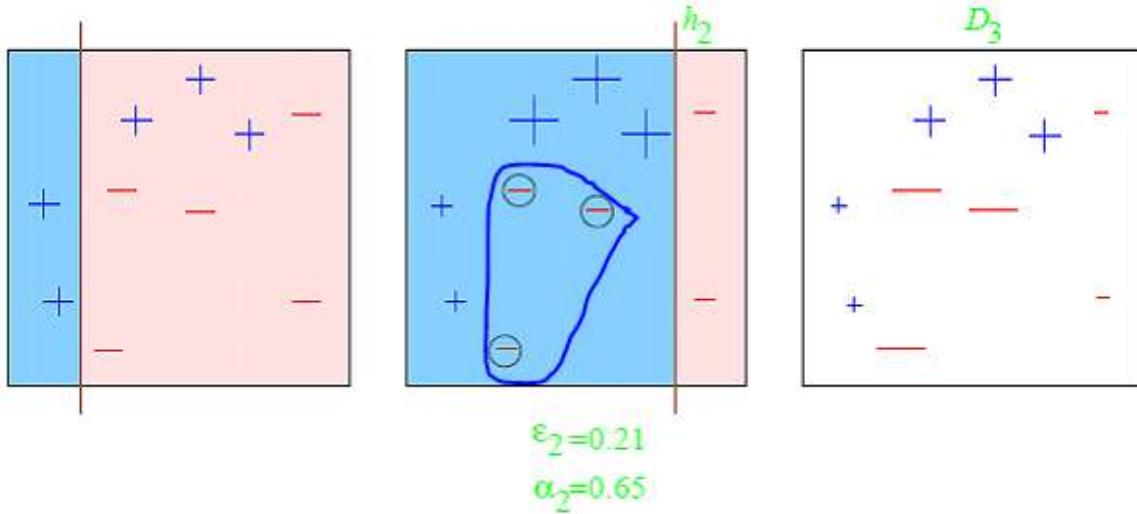
<https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=lectures.boosting>

In Adaboost, same as the other Boosting models, the base learners are of high bias and low variance. So if we assume the decision trees with a depth of 1 as the base learners, then the decision surface would be a straight line parallel to the 'Y' axis. So when the input data(let's say D_1) is given to the base learner, we do get a few misclassifications.



When we look at the classification made in ' h_1 ', 3 positive points are misclassified. We shall now either perform upsampling of the misclassified points (or) assign more weightage to the misclassified points and then train

the next base learner model using this updated/modified dataset. This time, it gives another decision surface that is a little better when compared to the decision surface obtained in the previous base learner.



In the predictions obtained from ' h_2 ', we could see all the positive points are classified correctly, but there are 3 negative points that are misclassified. So now we have to add more weightage to these 3 negative points and again pass this modified data as input to the next base learner. This way it continues until the 'K' stages and will get the best model at the end (with minimal amount of misclassifications).

This model is called AdaBoost because, at every stage we are adapting to the errors that were obtained in the previous stage, and are giving more importance to the points that are misclassified.

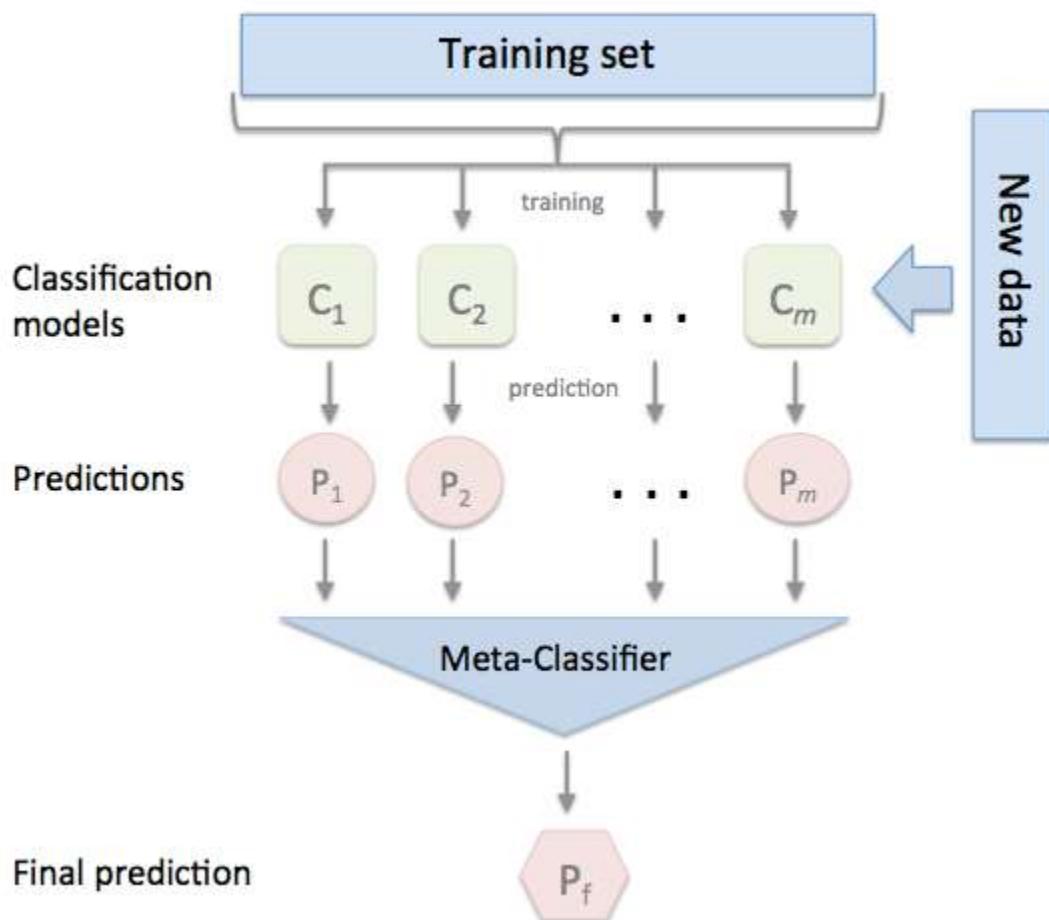
The core idea behind Adaboost is to adapt to the changes in the previous stage and give more weightage to the misclassifications in the next stage.

38.16 - Stacking Models

Stacking is an ensemble model that combines multiple classification models via meta-classifier. In Boosting we have seen that each base learner was built by taking the output of the base learner in the previous stage, into consideration. But here the individual classifiers are trained parallelly and are independent of each other. No model has nothing to do with the output of the other model.

Please go through the below webpage for an example of Stacking Classifier.

http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/



In this web page, we can see there are ' m ' different classification models. The Stacking Classifier is almost similar to the Bagging model, but with a few changes.

In Bagging models, all the base learners are of high variance and low bias. But here in Stacking, we generally go with the best fit models as the individual classifiers(ie., no high variance and no high bias).

Let us assume there are ‘n’ data points in our dataset and there are ‘m’ individual classifiers used in stacking, then if we denote our dataset as $D_n = \{x_i, y_i\}_{i=1}^n$, then

We have to first pass the training data as input to each of these ‘m’ individual classifiers. For each point (x_i, y_i) , let the output obtained from each of these individual classifiers be denoted as $h_{1i}(x), h_{2i}(x), h_{3i}(x), \dots, h_{mi}(x)$. So now, when a point (x_i, y_i) passes through ‘m’ individual classifiers, it gets ‘m’ outputs. Similarly all the ‘n’ data points are passed through the ‘m’ classifiers, and each point gets ‘m’ outputs. So the total number of outputs will be $n*m$.

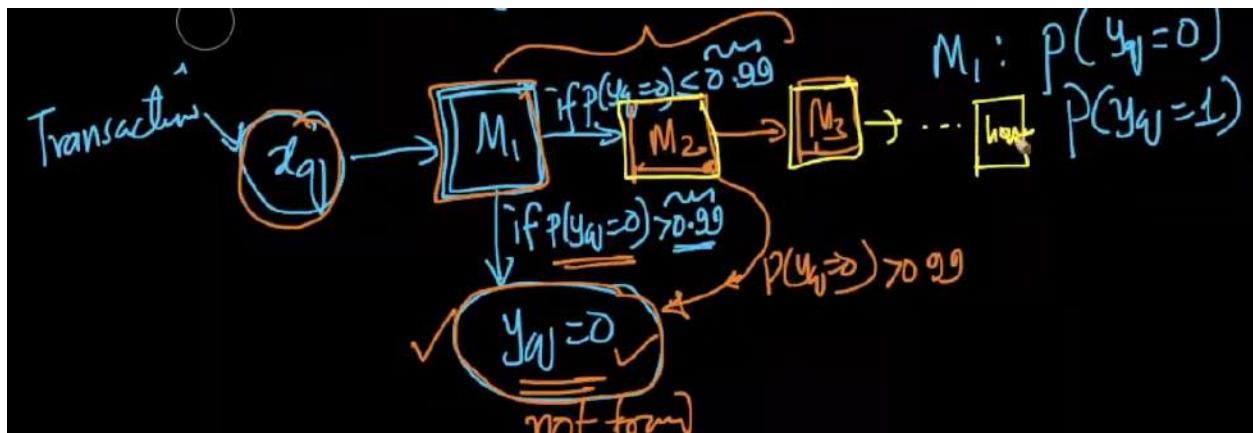
Now the meta-classifier mentioned in this example could be again any one classification model(like Logistic Regression, SVM, DT, etc). So now the same data points should be passed to the meta-classifier, but not directly. The outputs obtained from the ‘m’ classifiers for each data point are concatenated into an m-dimensional vector form. So in this way, we should concatenate the ‘m’ outputs of all the ‘n’ data points, and get the result in the form of an $n \times m$ matrix.

This $n \times m$ matrix along with the class labels y_i 's will be passed as an input to the meta-classifier and the output of the meta-classifier will be the final predicted class label for a given query point.

38.17 - Cascading Classifiers

So far we have learned about Bagging, Boosting and Stacking ensembles. Let us now learn about another ensemble technique called Cascading.

Let us consider the problem of predicting whether a given credit card transaction is fraudulent. Let us denote the class labels of the fraudulent cases as 1, and those of the non-fraudulent cases as 0.



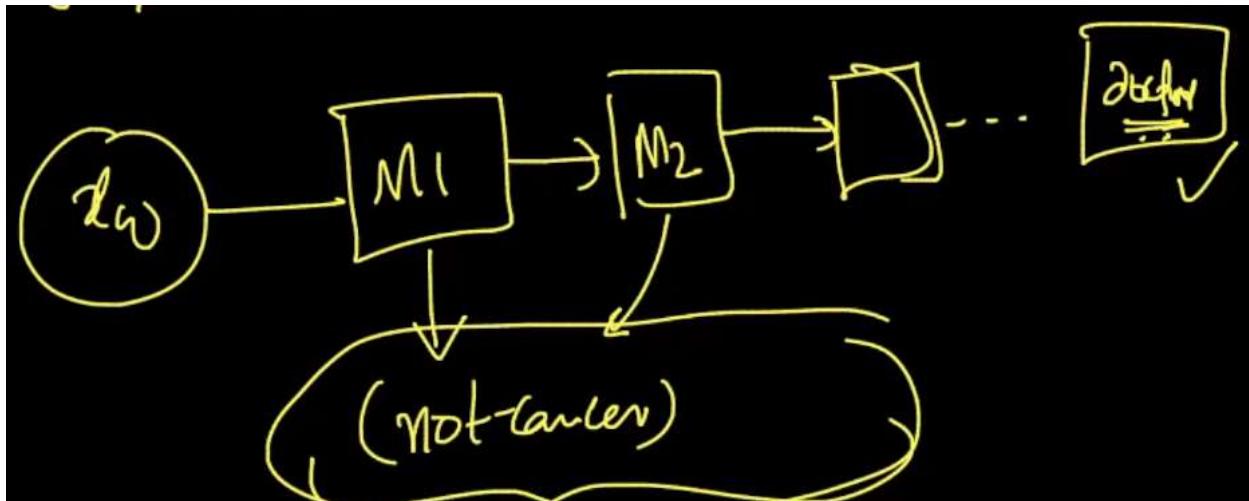
For any query point ' x_q ', our model ' M_1 ' predicts the value $P(y_q=1)$ and $P(y_q=0)$. As this is a highly critical study and we don't want to take a risk, let us assume, we have set the threshold of $P(y_q=0)>0.99$. So for a given transaction, only if the value of $P(y_q=0)>0.99$, then only we confirm it as a non-fraudulent case, otherwise, we confirm it as a fraudulent case.

But in case, if we do not just want to make a decision on the basis of the prediction of one model, if we want to make judgement only after multiple models give the same result, then in such cases, let us assume we shall add the models ' M_2 ' ad ' M_3 '.

So if ' M_1 ' yields $P(y_q=0)>0.99$, then we confirm it as a non-fraudulent case. But if $P(y_q=0)\leq 0.99$, then instead of directly confirming it as a fraudulent case, we can again pass the query point through multiple models. If all those models give $P(y_q=0)>0.99$, then we can confirm it as a non-fraudulent case. Otherwise, we confirm it as a fraudulent case.

Cascading models are typically used when the cost of making a mistake is high. Even if a small mistake of 0.1% could severely impact the business, in such cases, we employ the ensemble models.

One more case study where we can employ the cascading models is in the problem of predicting whether a given patient has cancer or not, on the basis of the data from his/her medical tests. Here even a small mistake of 0.1% could severely affect the life of the patient.



Working of a Cascading Model

Whenever we pass the whole training dataset ' D_{Train} ' as an input to the model ' M_1 ', the model predicts $P(y_q=0)>0.99$ for a few points and let all these points combinedly be denoted as D^I .

So now when we have to pass the data as an input to the model ' M_2 ', then instead of passing the whole ' D_{Train} ', we have to exclude the data points of D^I and pass the remaining data points. (ie., $D_{M1} = D_{Train}-D^I$)

So again the model ' M_2 ' predicts $P(y_q=0)>0.99$ for a few points and let all these points combinedly be denoted as D^{II} . So now we pass the data points $D_{M2}(D_{M1}-D^{II})$ as input to the model ' M_2 ' and so on. All those points which got the predictions as $P(yq=0)>0.99$ are excluded(because we already have made a clear decision on them as non-fraudulent cases) and the remaining points are passed as input to the next models in the sequence.

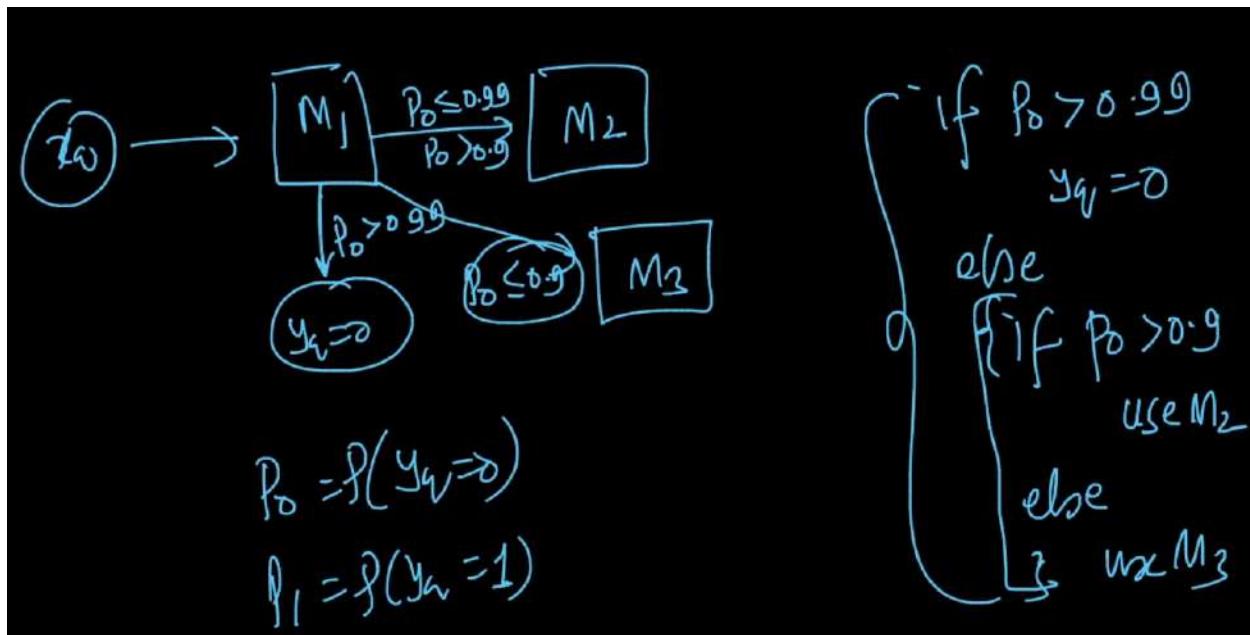
Note

We pass only the data points x_i 's as input to each of the individual models in the cascading models. In the beginning, if each point in ' D_{Train} ' has ' d ' dimensions (ie., $x_i \in R^d$), then there would be no changes in the

dimensions of the data for the other models in the sequence. **We do not carry the output of previous models to the next models as input.** Please do not get confused here. If our initial dataset has 1000 dimensions, then appending the output(ie., predicted y_i 's) of ' M_1 ' to the 1000-dimensional data, and sending the new 1001-dimensional data as input to ' M_2 ' is not correct. We do not carry forward the outputs of previous models, as the inputs to the next models in cascading classifiers.

Note

Also, it is not mandatory to have only one model at each stage in cascading. We can have multiple models at one stage, depending on a few if-else conditions we can choose which points to be sent to which model as inputs. Below is an example of it.



In this example, out of all the points which got the predictions $P(y_q=0) \leq 0.99$ by the model ' M_1 ', we are again dividing them into two groups(one with $P(y_q=0.9) \leq 0.9$ and the other with $P(y_q=0) > 0.9$). Those points with $P(y_q=0) \leq 0.9$ are sent as input to the model ' M_3 ' and those points with $P(y_q=0) > 0.9$ are sent as input to the model ' M_2 '.

45.1 Introduction

Featurization/Feature Engineering is the most important aspect of Machine Learning. Featurization is about converting the data of one type into a numerical vector form. Feature Engineering is about modifying the features so that they make the machine learning models perform well at prediction.

Out of the various featurization techniques available for the text data, the most commonly used techniques are

- a) Bag of Words (BOW)
- b) Term Frequency - Inverse Document Frequency (TF-IDF)
- c) Average Word2Vec
- d) TF-IDF Weighted Average Word2Vec

For the categorical features, the most commonly used featurization techniques are the mean response time and one-hot encoding.

Feature Engineering is the process of using domain knowledge to extract the new variables from the raw data, which makes the machine learning algorithms work better. Feature Engineering is a difficult task as it is domain-specific.

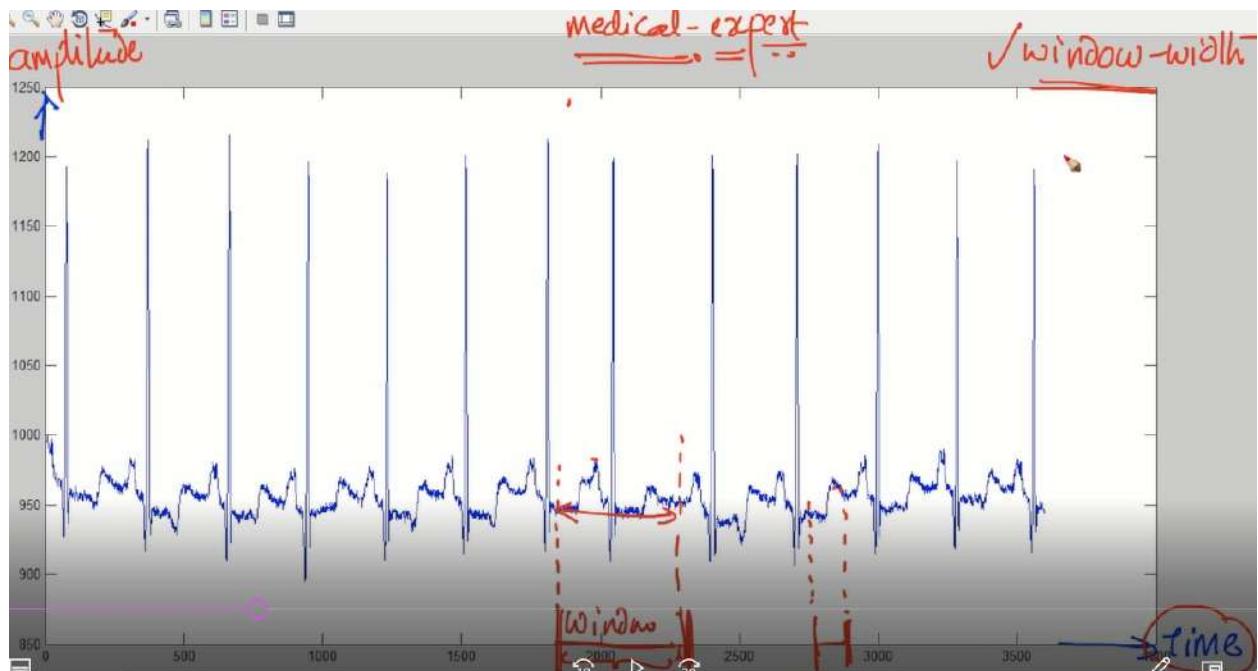
Feature Selection is to select some useful, relevant features among the features we generate/find/have in our data.

Note

If we are using a linear model like Logistic Regression or Linear SVM, then we perform feature engineering to convert the non-linearly separable data into a linearly separable format, to make our models work better. In a nutshell, we are trying to transform our features to better suit our models and hence design better overall models.

45.2 Moving Window for Time Series Data

Moving window is the simplest featurization technique applied to the time series data. Let us take the example of ECG/Medical data, which measures the heartbeat. We have to convert this time series data into numerical vector form, and for that, we have to sit with the medical/domain expert.



Some of the standard featurization techniques that are performed on the data, that is picked from this window are

- a) Mean, Standard Deviation
- b) Median, Quantiles
- c) Max, Min, Max-Min, Max/Min (or) local minima/maxima
- d) Zero-Crossing (checking how many times, the signal has crossed zero).

All these featurization techniques are domain-specific. Before going with any featurization technique, we should seek expert advice.

Example

Let us consider the problem of predicting cardiac arrest in the next 10 minutes.

Steps of Featurization

- 1) Define Window Width
- 2) Find out the features in this window that are useful in predicting heart-attack.
- 3) After getting the unimportant features, take the combination of those features as ' x_i ' and the response binary variable as ' y_i '. (Here the response variable indicates cardiac arrest occurs or not)

For any problem, we have to do some domain-based research in order to decide which features are important for the task and which featurization technique has to be applied.

Q) Why is this approach called the Moving Window approach?

Ans) This approach is called the Moving Window approach because a window of defined size and shape, is moved over the data, the moving distance is equal to the width of the window.

Q) How to determine the width of the window? Is it mandatory for all the windows to be of the same size?

Ans) In case if we are taking multiple windows into consideration, we need to take all the windows of the same width. But we might get overlapping windows where one window overlaps over another in case of having multiple windows of different widths.

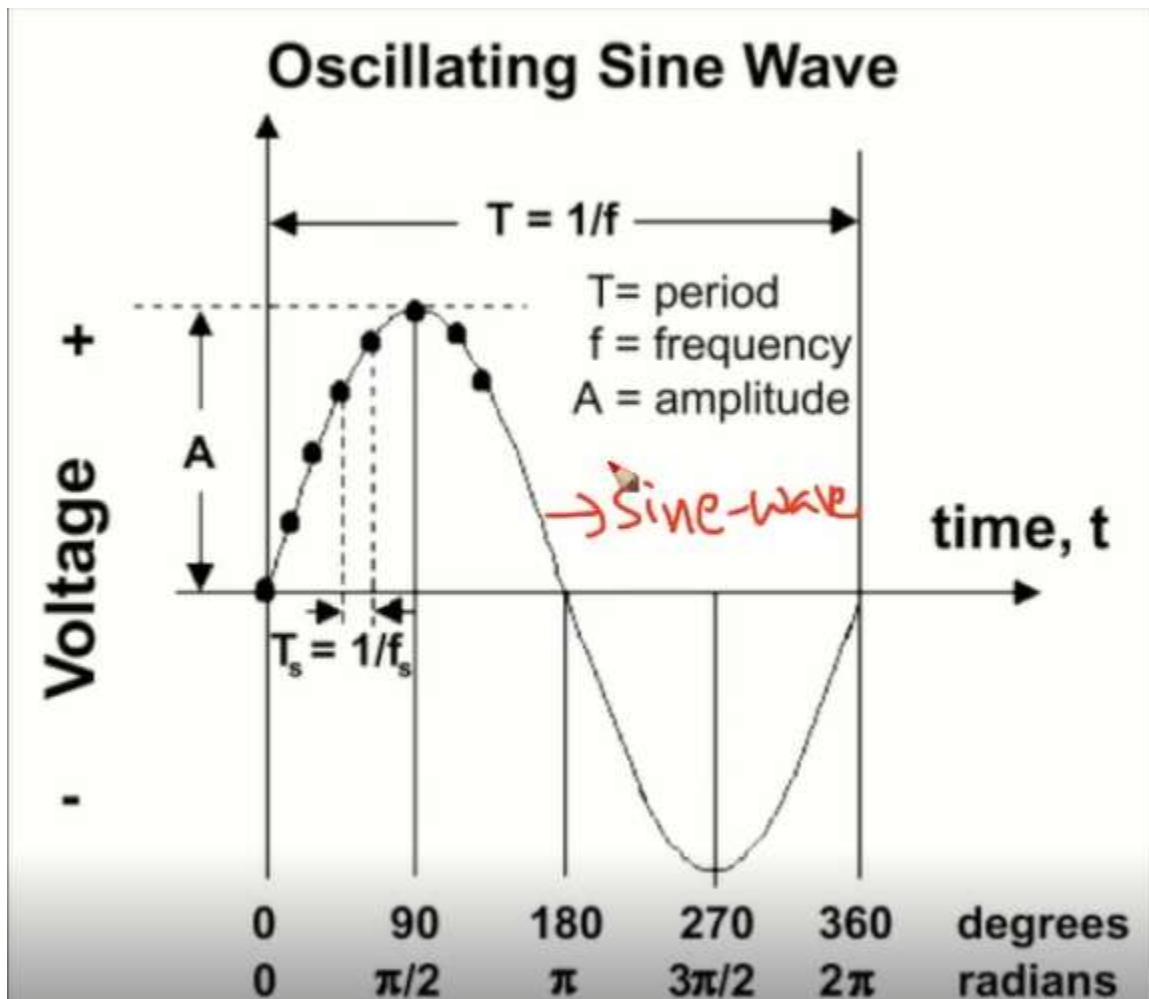
Choosing windows of different widths would lead to some implementation challenges like

- (i) How to determine the width of each window?
- (ii) If we have just one width for all the windows, we can treat it as a hyperparameter. If we have multiple windows, the number of hyperparameters would increase and it would be very hard to tune all these hyperparameters.

In practice, there is going to be no benefit in having a variable-sized window.

45.3 Fourier Decomposition

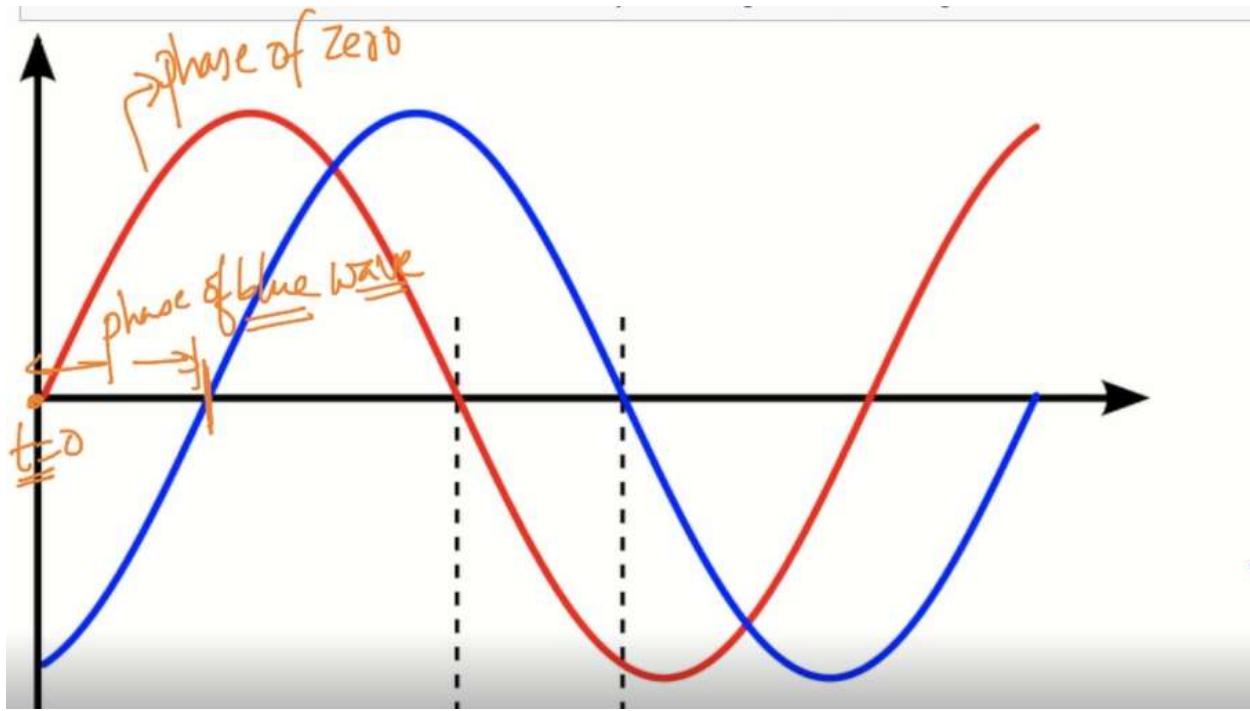
One of the most popular ways to represent the time series data is using the Fourier transform.



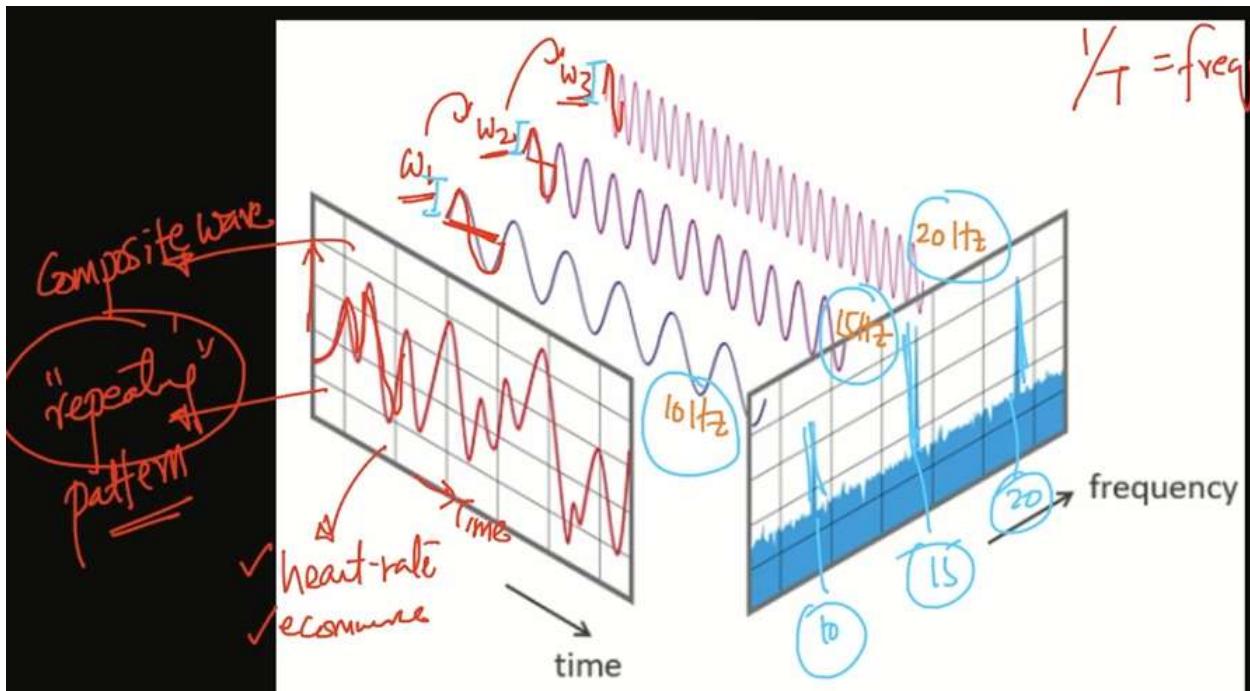
The number of oscillations per second is called **Frequency**.

The time taken to complete one cycle is called the **Time period**.

The height of a signal from '0' is called **Amplitude**.



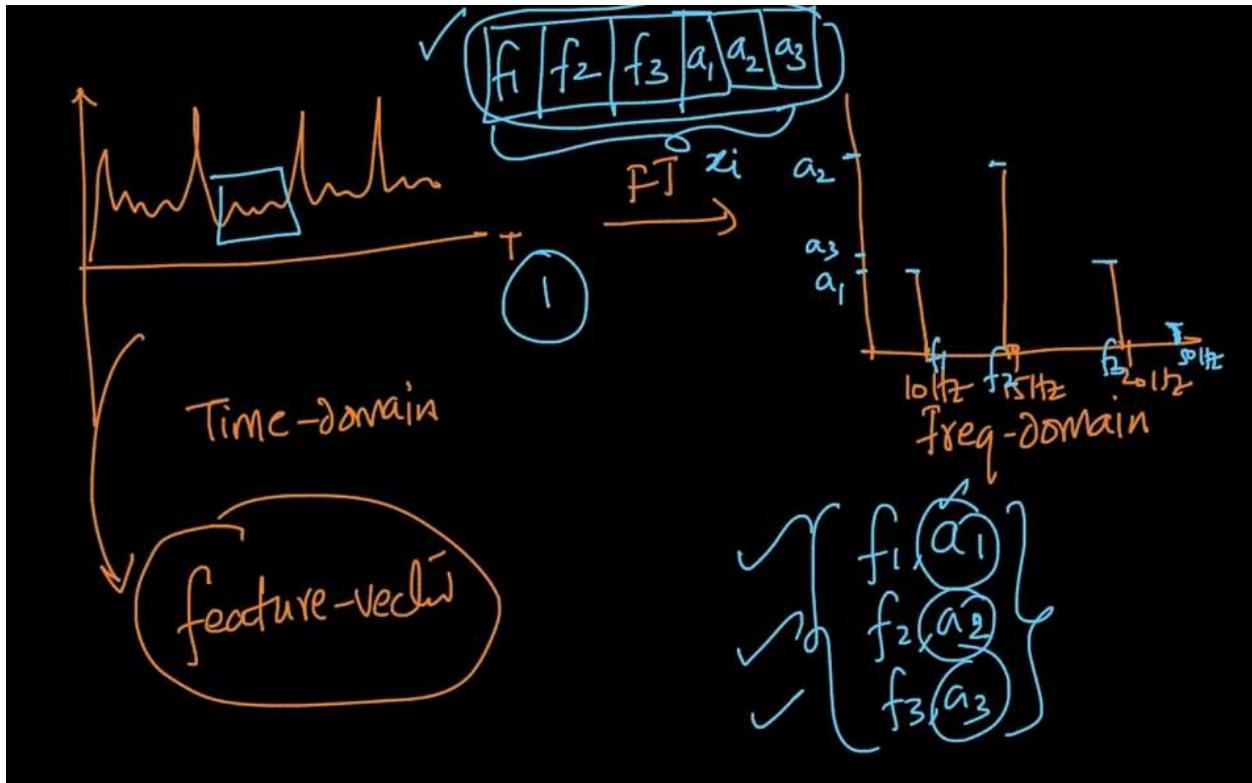
If the given data is in the form of a composite function/wave, we can break it into a sum of multiple sine waves. It applies to repeating patterns.



So after obtaining the frequency domain representation of the input time signal, we then have to obtain the numerical vector in the form

f_1	f_2	f_3	a_1	a_2	a_3
-------	-------	-------	-------	-------	-------

This is the **Fourier representation** (or) **Fourier representation** of the time series data. Fourier representation is always preferred when the given time series data has repeating waveforms. The Fourier transform breaks the given composite signal into multiple waves (each wave has a different time period and frequency)



The representation of the signal in Frequency domain by applying the Fourier Transform is necessary if there are repeating patterns in the given input Time Series signal. In such cases, the frequencies are very important.

45.4 Deep Learning Features: LSTM

Deep Learning models take lots of data as input and learn the best featurization techniques for the given data automatically. These features developed using the deep learning models are called Deep Learnt Features. These features are almost the best features.

Deep Learning models give the best results and features when the input data is in time series, text, or image formats. Deep Learning models should be used only when we have lots of data. It is not recommended to use the Deep Learning models if we have a small amount of input data.

Q) Why don't we use deep learning models instead of other models, when we have lots of data?

Ans) When we have lots of data, the deep learning models tend to work better than any other classical ML model. When we have the data in the form of images, videos, audio, or if we are working on complex tasks like machine translation of the text from one language to another, etc, the deep learning models work better. Based on the features we have, simple models like Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, etc work fairly well.

If the model interpretability is important in the problem, then using the classic ML algorithms is better, instead of using the deep learning models.

If we are working on low latency applications, where if a query point ' x_q ' is given, the time taken to give the response ' y_q ' should be of the order of milliseconds, then as the evaluation time of the deep learning models is high, it is better to go with classical ML algorithms.

Q) How are the deep learning models different from the classical ML models?

Ans) In a deep learning model, we featurize the data using all of the initial layers and the final classification is done by the last layer which is often a Logistic Regression (or) an extension to it (or) a similar method. So the deep learning models perform feature engineering and classification (or) regression in a single model.

While in classical ML algorithms, we featurize the data using techniques like Fourier Transforms and then input the transformed features to the model, and then classification (or) regression is performed.

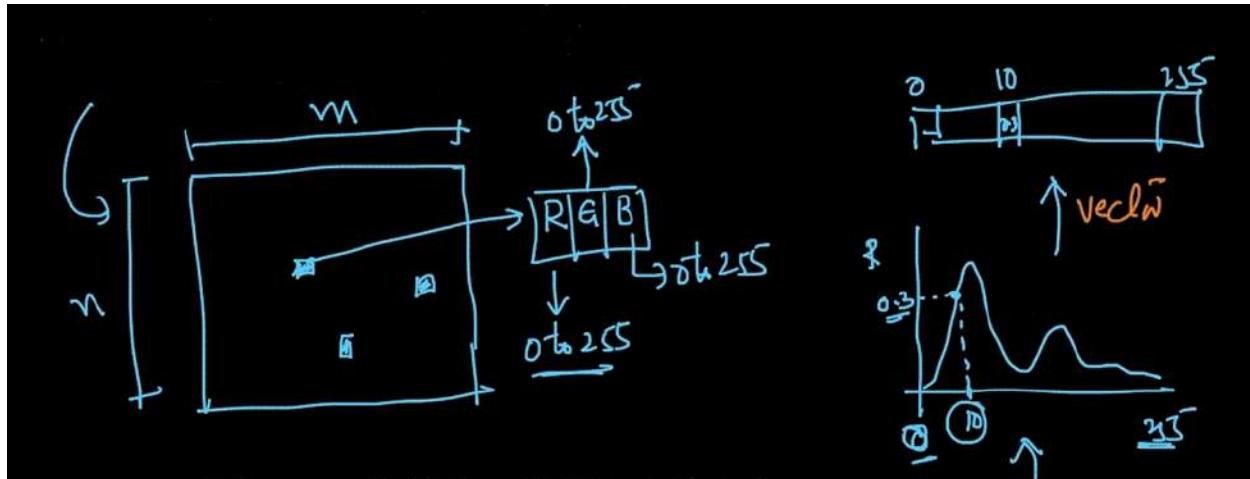
The decision of choosing a deep learning model or a classical ML model depends on the real-world problem we are solving, the problem constraints, and the requirements of the problem to be fulfilled.

45.5 Image Histogram

There are two types of Image Histograms. They are

- a) Color Histograms
- b) Edge Histograms

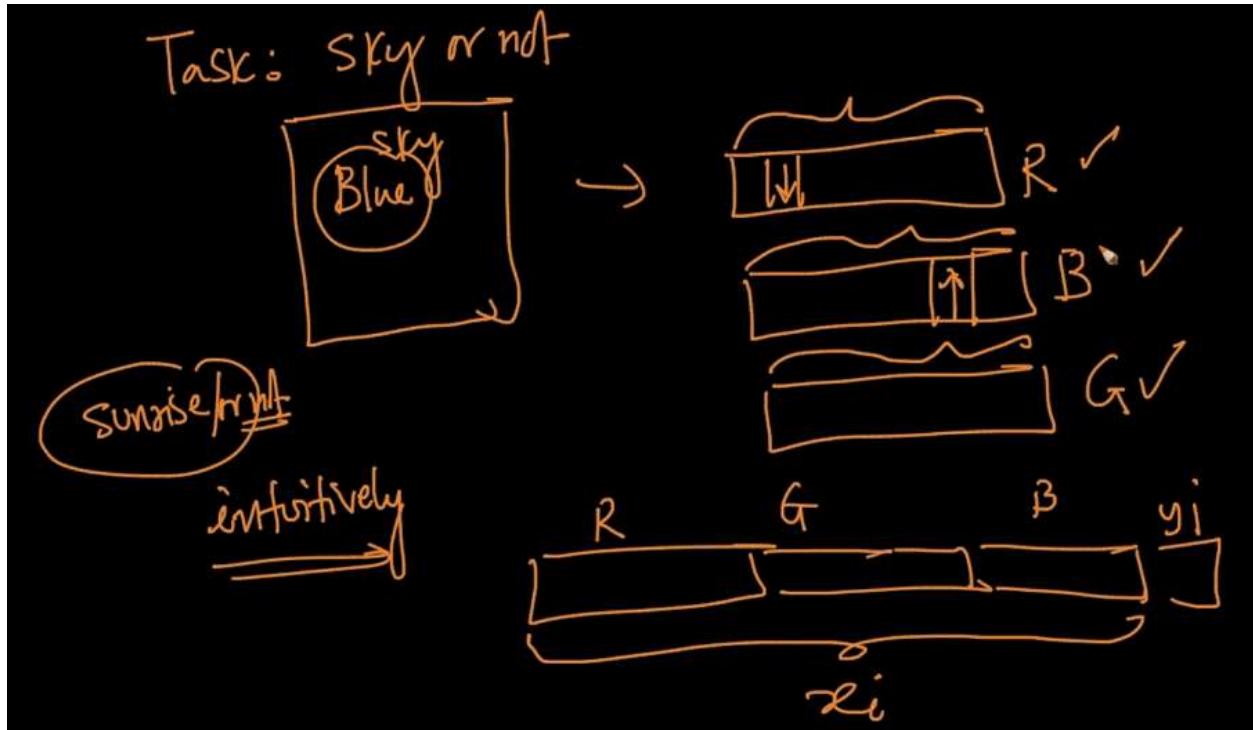
a) Color Histograms



Every image consists of pixels and every pixel consists of 3 values (R,G,B). There are a total of $n \times m$ pixels in the image. (where $n \times m$ is the dimensionality of an image)

We have to take the first component values(ie., values associated with red color) of all the pixels in the image, and have to plot the histograms with all possible values(ie., from 0 to 255). The possible values(ie., 0 to 255) are on the 'X' axis and their corresponding frequencies are on the 'Y' axis, and this histogram will be converted into a 256-dimensional vector, with the frequencies as the vector components.

Similarly, we have to convert the histograms of the green and the blue colors into 256-dimensional vectors each. For every color image, we get three 256-dimensional vectors. We then have to combine these three 256-dimensional vectors(ie., concatenation) and use the obtained 768-dimensional vector ($3 \times 256 = 768$) for data modeling.

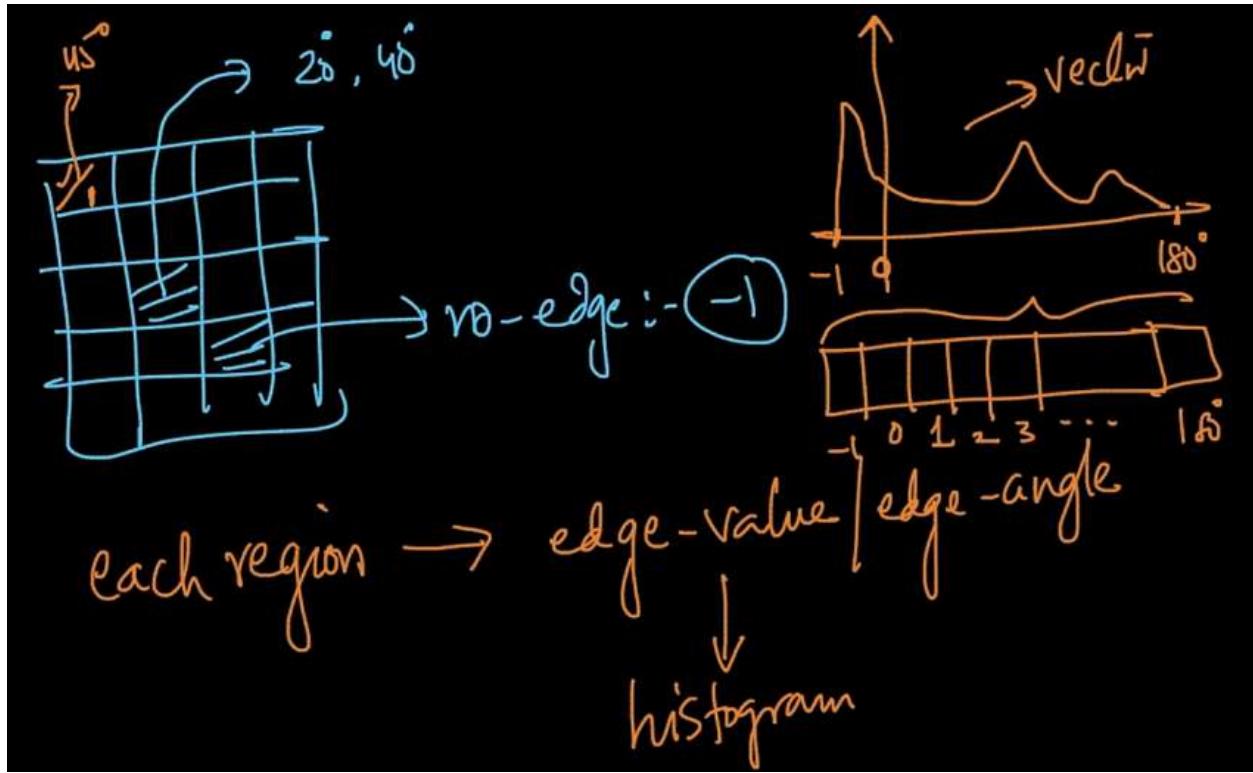


The color histogram can recognize only the colors, but not the shapes like square, rectangle, circle, etc. In order to recognize the shapes, we have another type of histogram called **Edge Histogram**.

b) Edge Histograms

Here we have to convert the image into a grid and in each cell, we have to check for an edge. If there is no edge in a cell, then the value in that cell becomes -1. In case, if we have multiple edges in a cell, then the edge with a larger angle is assigned to this cell.

So we get the edge angle/value to each of the cells in the grid. Now we have to plot a histogram with degree values (0° to 180°) on the 'X' axis, and their corresponding frequencies of angles on the 'Y' axis.



Note

While plotting the edge histograms, we have to also add the value '-1' on the 'X' axis, which indicates there is no edge present in the given cell. We then have to convert this histogram into a 182-dimensional vector(The values 0° to 180° and it also includes '-1')

45.6 Keypoints: SIFT

Scale Invariant Feature Transform (SIFT) is a very popular technique for detecting the objects in an image. It can also be used in featurizing the images.

The SIFT algorithm mostly detects the key points (which are mostly the corners) in a given image. For every key point, it creates a 128-dimensional vector for each of the key points. The collection of images that we use to compare the features for a new query point/image, is called **Database Image**. SIFT is extensively used in Image Search.



The images could be identified easily, even if the size of the image changes. This property is called **Scale Invariance**. Similarly, the images could be identified easily, if they are rotated. This property is called **Rotational Invariance**. SIFT technique possesses these two properties. If we need to compute the features using SIFT, we have to go with the **OpenCV** library, which has a Python interface.

Note

Key points in the SIFT technique are those pixels in which we have the objects. For example, if we have an image of a white wall with a door/window, then those pixels that cover the door/window are called **key points**, whereas the other pixels aren't taken into consideration.

Most image processing applications tend to use grayscale images over the color images, as the grayscale images tend to have much less information to process, and also it is simpler to handle the grayscale images when compared to the color images.

As we know there are 3 channels in an RGB image, and only one channel in a grayscale image, we end up having fewer parameters if we use grayscale images, when compared to the RGB images.

Q) Why do we get only a 128-dimensional vector as an output in SIFT when compared to any other number of dimensions?

Ans) SIFT is specifically a 128-dimensional vector that summarizes (or) describes a 16X16 window patch. The SIFT is obtained by dividing the 16X16 window into 4X4 bins. (Here each bin has the dimensions 4X4. When we divide, we get 16 bins)

Each bin has 8 orientation bins (or) channels. So the dimensionality becomes $4 \times 4 \times 8 = 128$.

45.7 Deep Learning Features: CNN

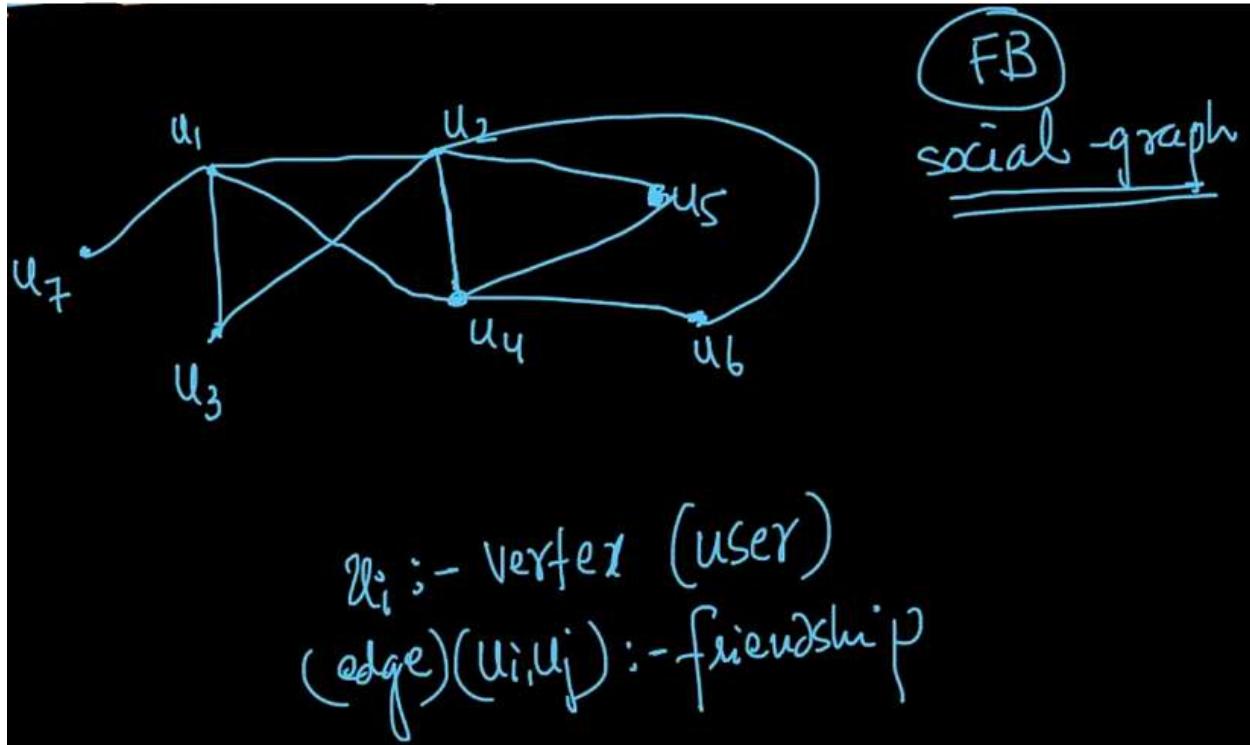
For featurizing the time series data, we have the LSTM technique. For featurizing the image data, we have the CNN technique. If we give lots of data to the CNN model, it will automatically detect the best featurization in the data. To solve a problem, the default technique is CNN (or) a variation of it in most of the image-based applications

45.8 Relational Data

If the data is stored in the form of tables, then we call the data as relational data. If the entire data is stored in multiple tables, then we have to pick the features that make some sense in the context of the problem we are solving, combine them and then perform data analysis. We also can do the same not only in SQL but also in Python (using Pandas library). The features that are picked in this way are called **Relational Features**, and these features will change when our problem changes.

45.9 Graph Data

Let us consider the problem of recommending friends for a user ' u_i '. If two users ' u_i ' and ' u_j ' are already friends, then we have to predict the label as '1', otherwise, we have to predict the label as '0'.



$u_1, u_2, u_3, \dots, u_7$ are the **vertices** and the connections between these vertices are called the **edges**.

Let us assume we now have to recommend friends for ' u_4 '. As ' u_4 ' is connected to almost all of them, we are left only with ' u_3 ' and ' u_7 '.

Pair Mutual Friends

$(u_3, u_4) \rightarrow u_1$

$(u_4, u_7) \rightarrow u_1, u_2$

For example, we can design features like

$f_1 \rightarrow$ Number of Mutual Friends

$f_2 \rightarrow$ Number of paths distance (ie., the number of ways to reach from one vertex to another)

We can design many more features like this, and these features are called **Graph-Theoretic** (or) **Graph-Based** Features.

These kinds of graphs which have the users as the vertices and the friendships/connections as the edges are called **Social Graphs**.

For all the users ' u_i ' and ' u_j ', if there exists friendship/connection between ' u_i ' and ' u_j ', then $(u_i, u_j) = 1$, otherwise $(u_i, u_j) = 0$.

If the number of mutual friends (or) number of paths distance between ' u_i ' and ' u_j ' are more, then there is a high probability for these two users to become friends.

45.10 Indicator Variables

Converting the features indicator variables is one of the techniques of feature engineering. Deciding whether to apply this technique is problem-specific.

Example 1

If we have ‘height’ as a feature, and it is a real-valued feature, then

- a) We can keep it as it is and go ahead with data modeling (or)
- b) We can convert it into an indicator variable as

if height > 150cm → class 1

height <=150cm → class 0

Here ‘height’ is a binary indicator variable. Mostly the indicator variables are binary.

Example 2

If a given feature is categorical, let’s say the feature ‘country’ is a categorical feature, then it can be converted into an indicator variable as

```
if country == 'India' OR country == 'USA':  
    return 1  
else:  
    return 0
```

Note

The usage of appropriate feature engineering techniques is problem-specific. Featurizing the data appropriately is a creative aspect of ML.

45.11 Feature Binning

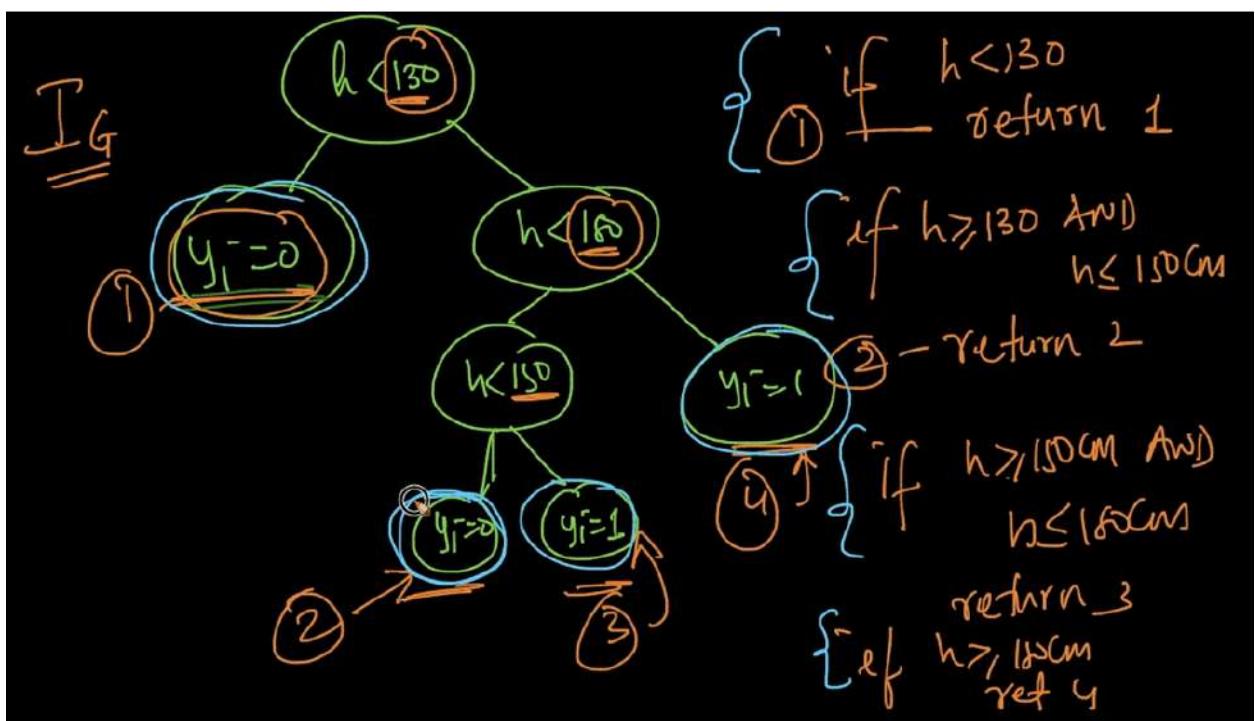
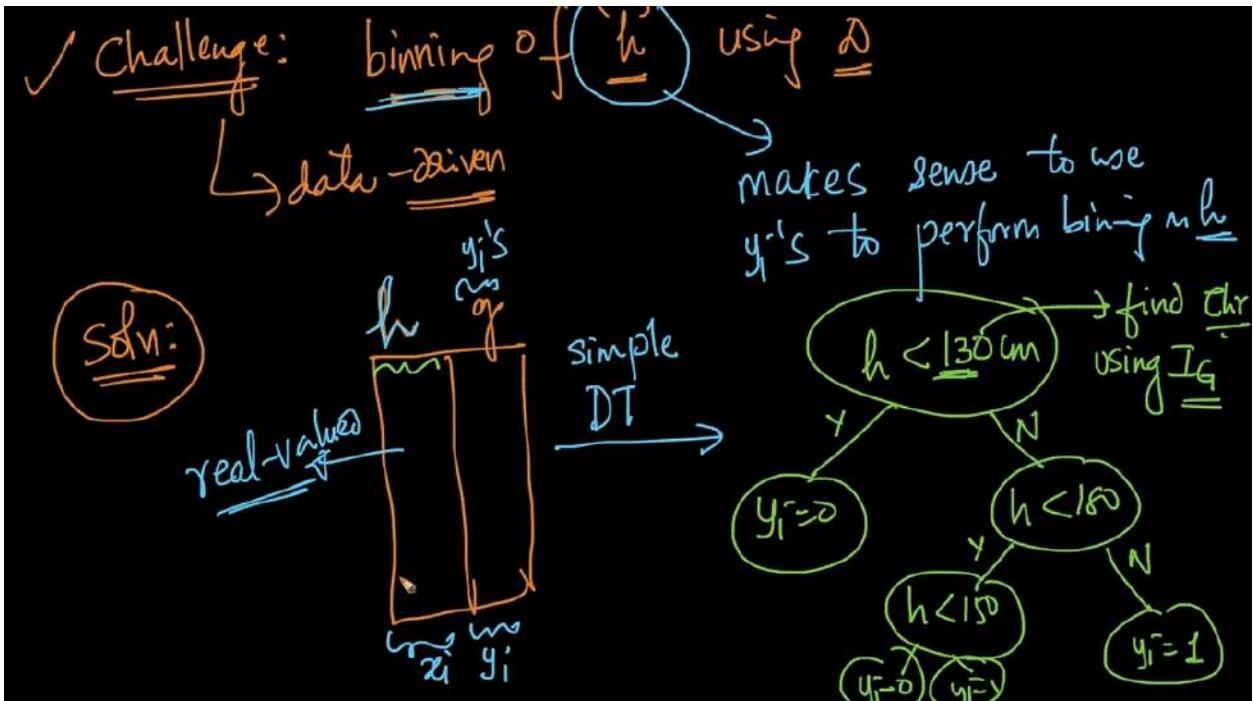
Feature Binning is a logical extension of the indicator variable technique.

Example

Let us take the same height example that was discussed in the previous section, but the conditions get slightly changed when we apply Feature Binning.

```
if height < 120cm:  
    return 1  
if height < 150cm AND height >= 120cm:  
    return 2  
if height < 180cm AND height >= 150cm:  
    return 3  
if height >= 180cm  
    return 4
```

Feature Binning is also called **Feature Bucketing**. It divides the available data values into buckets. In the above example, the values 120cm, 150cm, and 180cm are called the **Thresholds**. Finding out the right threshold is very problem-specific.



45.12 Interaction Variables

Creating interaction variables is also a form of feature engineering.

Example

Let us consider the example of predicting gender on the basis of height, weight, hair length, and eye color.

If $\text{height} < 150\text{cm}$ AND $\text{weight} < 60\text{kg}$, this is called the 2-way interaction feature.

The other examples of interaction are

- a) $\text{height} * \text{weight}$
- b) $\text{sqrt}(\text{height}) * \text{weight}$
- c) $(\text{height} < 150\text{cm}) \text{ AND } (\text{weight} < 60\text{kg}) \text{ AND } (\text{hair length} < 75\text{cm}) \rightarrow$
3-way logical interaction feature.

Q) Given a task, how do we find good interaction features?

Ans) The first option we have is the Decision Trees. Let us assume we have 4 features: height, weight, hair length, eye color, and the output variable is gender.

Here we are coming up with the decision trees to find the new features, and these features will be useful to predict 'y'. Decision Trees in this context are used to find the interaction between the variables and figuring out the best set of features.

Here we are using a Decision Tree as a method to perform Feature Engineering.

Goal of Feature Binning/ Indicator Variables/ Interaction Variables

The goal is to engineer new features using the existing ones to better predict the class label. We typically add the new interaction feature to the original set of features and build a model. We typically never throw out the features and the data, unless we are really sure they are absolutely useless.

45.13 Mathematical Transforms

If ‘x’ is a single numerical feature, there is a type of feature engineering where we could apply mathematical operations on ‘x’, and this type of feature engineering technique is called **Mathematical Transform**.

Examples of Mathematical Transforms

- a) $\log(x)$, e^x
- b) $x^{1/2}$, $x^{1/3}$
- c) x^2 , x^3 , x^4 , (polynomial)
- d) $\sin(x)$, $\cos(x)$, $\tan(x)$,....

The best feature transform for any problem is problem-specific and domain-specific.

Note

If ‘x’ follows a power-law distribution, then applying logarithm on ‘x’ makes sense. For tree-based algorithms like RF, DT, and GBDT, we need not perform the box-cox/log feature transform as these algorithms simply use thresholds to split the datasets.

In practice, RF and GBDT are very popular as they do not need simple feature transforms to work well.

45.14 Model Specific Featurizations

Example 1

Let us assume we have a feature ' f_1 ' which follows a power-law distribution, and we want to apply logistic regression (works on the basis of Gaussian Naive Bayes which assumes every feature in the given dataset follows gaussian distribution), in such a case, as Logistic Regression assumes the features to follow the Gaussian distribution, and the feature ' f_1 ' follows a power-law distribution, in order to make the logistic regression model work better, we have to transform ' f_1 ' from power-law distribution form to Gaussian distributed form, by applying logarithm on ' f_1 '.

Example 2

Let us assume we have the features $f_1, f_2, f_3, y \in R$. From the domain knowledge, let's say 'y' is a linear combination of f_i 's. Let it be in the form $y = f_1 - f_2 + 2f_3$.

In this context, linear models like Logistic Regression would be appropriate. Models like DT, RF, etc couldn't work well on linear combinations.

Example 3

Let us assume from the domain knowledge, our class label 'y' is dependent on the interactions between the features ' f_1 ' and ' f_2 '. For example, predicting the gender of a person depends on the interactions between height and weight. In this case, the behavior of these interactions among the men and the women is different and models like DT/RF/GBDT are the perfect models to be used to solve this problem.

In this case, linear models like Linear SVM, Logistic Regression would not work well and the interactions could be easily caught by DT/RF/GBDT.

45.15 Feature Orthogonality

The more different/orthogonal the features are, the better the models would be. Let us assume we have 3 features ' f_1 ', ' f_2 ' and ' f_3 ', and the response variable 'y'.

- a) If each of the features ' f_1 ', ' f_2 ' and ' f_3 ' are highly correlated with 'y', and these features are highly correlated with each other, then combining these 3 features and using them, will have the least overall impact on the model we build to predict 'y'.
- b) If each of the features ' f_1 ', ' f_2 ' and ' f_3 ' are highly correlated with 'y', and these features are not correlated with each other, then combining these 3 features and using them, will have a huge overall impact on the model we build to predict 'y'.

If we want to create a new feature ' f_4 ', then it must be highly correlated with the response variable 'y', but not correlated with the input features ' f_1 ', ' f_2 ', and ' f_3 '. In such a case, adding ' f_4 ' to the model will have a huge impact.

Note

Two vectors are orthogonal to one another if their dot product is zero. A set of vectors is an orthogonal set if each distinct pair of vectors in the set have a dot product of zero. In two dimensions, it means the vectors are perpendicular to one another. Since the correlation of two random variables is zero, if the covariance is zero, according to this definition uncorrelatedness is the same as orthogonality. Independent variables are usually given as sequences of numbers, for which orthogonality is naturally defined by the dot product.

Note

We often use domain knowledge to design the features that are well correlated with the errors which often involves a lot of data analysis. Additionally, we can try and build feature interactions, mathematical transforms, etc which are well correlated with the errors to improve our model.

Q) How to design a feature ' f_4 ' that is highly correlated with 'y', and least/not correlated with ' f_1 ', ' f_2 ', and ' f_3 '?

Ans) For all the points, we have to compute the error

$$e_i = y_i - y_i^{\wedge}$$

We have to design the feature ' f_4 ' such that it is highly correlated with the error (e_i). After designing ' f_4 ', we have to build a model with all the four features. This is similar to the concept of gradient boosting, where we compute a new feature (ie., error), add it to the model, and make predictions. We again compute the error, add it again to the model and make predictions. This way it continues.

But we should ensure that our model doesn't overfit, as the chances for overfitting are more in gradient boosting. In this scenario, we say ' f_4 ' is orthogonal to ' f_1 ', ' f_2 ', and ' f_3 '.

45.16 Domain-Specific Featurizations

For example, if we want to predict a heart attack using the ECG data, it is always important to do some research and study the existing featurization techniques designed by the doctors/healthcare specialists.

For any domain, first of all, we have to do some research on the existing featurization techniques and try to build new ones on top of the existing ones instead of building everything again from scratch.

45.17 Feature Slicing

Let us consider the example of predicting the gender given the features are height, weight, hair length, eye color, country. Let's say the possible values for the 'country' column are 'India' and 'USA'.

If 80% of the data points have 'country' value as 'India', and the remaining 20% of the data points have 'country' value as 'USA', then if we build a model 'M' on it, then the model seems to perform better on data associated with 'India' in most of the cases and perform worse on data associated with 'USA' in most of the cases.

When we make predictions of the class labels on the test data, we do encounter a few errors in prediction. If the error rate in the case of points with minority 'country' value is very high when compared to the case with majority 'country' value, then the only strategy that could work is to split the train data 'D' into ' D_{Train} ' and ' D_{USA} ' and build 2 separate models.

Using this idea of slicing the data on the basis of the feature values, we need to build separate models for each feature value and compute the performance for each model.

In order to perform feature slicing, two conditions are to be satisfied.

- a) Each category of a feature should have different behavior.
- b) There should be a sufficient(decent) number of points in each slice.

Q) Is Feature Slicing applicable for the continuous numerical features also?

Ans) Yes, it can be applied to continuous numerical features. But feature slicing is proved to be more useful when applied to the categorical features.

Q) Do we have to apply feature slicing on every categorical feature in the dataset?

Ans) We do not separate the data for every categorical feature and create separate models. We'll check the error distribution for every value of the categorical feature. If the error distributions for each value are different, then we will go for the splitting of data, based on the categorical column values and we train multiple models.

46.1 Calibration of Models: Need for Calibration

Let us consider a 2-class classification problem (ie., $y_i \in \{0, 1\}$). Let $D_{Tr} = \{x_i, y_i\}$ be the training data. Let us now train a model on this training data, and let the model function be denoted as $f(x)$.

For any given query point ' x_q ', the output is given by $y_q = f(x_q)$. So in the task of classification, the output ' y_q ' sometimes may or may not be the probabilistic estimate (ie., $P(y_q=1|x_q)$).

For example, in Naive Bayes,

$$P(y_q=1|x_q) \propto P(y_q=1)^{\pi_{i=1}^d} P(x_{qi}|y=1) \quad \dots (1)$$

$$P(y_q=0|x_q) \propto P(y_q=0)^{\pi_{i=1}^d} P(x_{qi}|y=0) \quad \dots (2)$$

We can say,

a) $y_q = 1$, if $P(y_q=1|x_q) > P(y_q=0|x_q)$

b) $y_q = 0$, if $P(y_q=1|x_q) < P(y_q=0|x_q)$

So when we look at (1) and (2), we can say the terms on the left side of the ' \propto ' symbol are proportional to the terms on the right side, but not exactly equal. So when an input ' x_q ' is given, and if the output is ' y_q ', then this ' y_q ' may or may not be the actual $P(y_q=1|x_q)$ (or) $P(y_q=0|x_q)$.

Need for Calibration

Let us assume we are working on a binary classification task, and are using Log-Loss as the metric to assess the model performance. The log-loss formula for a binary classification task is given as

$$\text{Log-Loss} = (-1/n) * \sum_{i=1}^n [y_i * \log(p_i) + (1-y_i) * \log(1-p_i)], \text{ where}$$

$n \rightarrow$ total number of points

$$p_i = P(y_i=1|x_i)$$

We need the actual probability(p_i) value, as the log-loss is determined using the actual probability(p_i) value. If the actual probability(p_i) value is wrong, then the whole log-loss value goes wrong.

The process looks like as shown below

$$x_q \rightarrow f(x_q) = y_q \rightarrow y_q \rightarrow \text{calibration model} \rightarrow P(y_q=1|x_q)$$

All the models do not provide probabilistic estimates. Some models give poor estimates of the class probabilities, and some of them do not support the probability predictions (SVM is the best example, which gives

only the class labels, but not the probabilistic estimates).

The calibration module allows us to better calibrate the probabilities of a given model (or) to add support for probability prediction.

Calibration is just building one more model on top of the existing model to correct the class probabilities. In calibration, we are simply modifying the outputs of the previous model slightly, to convert them into the class probabilities.

The probability outputs by the models like logistic Regression, Decision Trees, Naive Bayes, etc are often not well-calibrated which can be observed by plotting the calibration plot. Hence we use calibration as a post-processing step to ensure that the final class probabilities are well-calibrated. If the calibration plot looks like a 45-degree straight line, then it means there is no need to apply calibration to the model. In such a case, we can skip the calibration step.

Calibration plots are used only in binary classification tasks. For a multi-class classification task, we first have to convert the task into multiple binary classification tasks(using the one-vs-rest approach), and then apply calibration on top of each binary classification model.

Calibration is a mandatory step, if we want probabilistic estimates as the output, as calibration corrects these probabilities. Also if we are using log-loss as the performance metric, which needs the values of $P(y_i|x_i)$, the calibration is a mandatory step to be followed.

46.2 Calibration Plots

Let us assume we have a model 'f' that was trained on the training dataset ' D_{Train} '. Let us consider our cross-validation dataset is given as

$$D_{cv} = \{x_i, y_i\}.$$

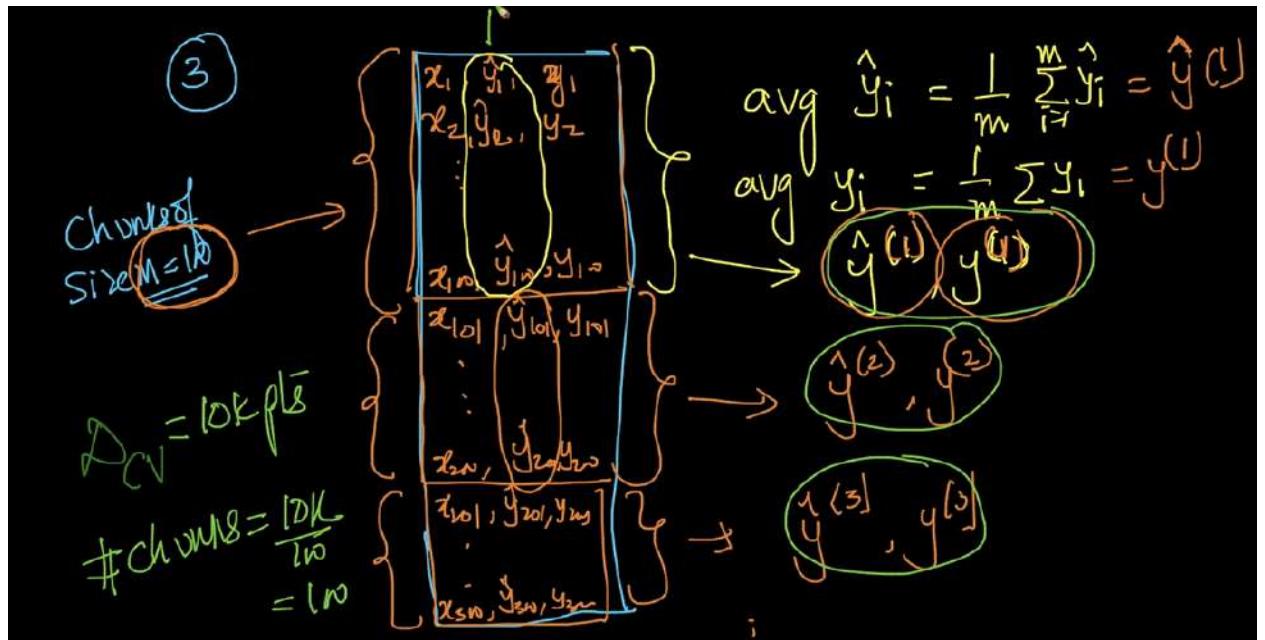
For each data point ' x_i ' in ' D_{cv} ', we have to make the class label prediction using the model 'f', and let the predicted class label be denoted as ' \hat{y}_i '. The actual class labels are denoted as ' y_i ', and $y_i \in \{0, 1\}$.

Steps of Construction

- 1) For each point (x_i, y_i) in ' D_{cv} ', we have to predict ' \hat{y}_i ' = $f(x_i)$, and tabulate the values as

x_i	y_i	\hat{y}_i

- 2) Sort this table in the increasing order of ' \hat{y}_i '.
- 3) After sorting, break the table into chunks of size 'm' each. For each chunk, we have to calculate the average ' \hat{y}_i ' and average ' y_i '.



$$\text{Average } \bar{y}_i = (1/m) * \sum_{i=1}^m y_i \quad \dots \quad (1)$$

$$\text{Average } \bar{y} = (1/m) * \sum_{i=1}^m \bar{y}_i \quad \dots \quad (2)$$

The averages for these chunks are denoted as $(\bar{y}^{(1)}, \bar{y}^{(1)})$, $(\bar{y}^{(2)}, \bar{y}^{(2)})$, $(\bar{y}^{(3)}, \bar{y}^{(3)})$,

Let us assume there are 'n' data points in ' D_{cv} ' and the size of each chunk is 'm', then

Number of chunks = n/m .

Now the data used for calibration is denoted as ' D_{calib} '.

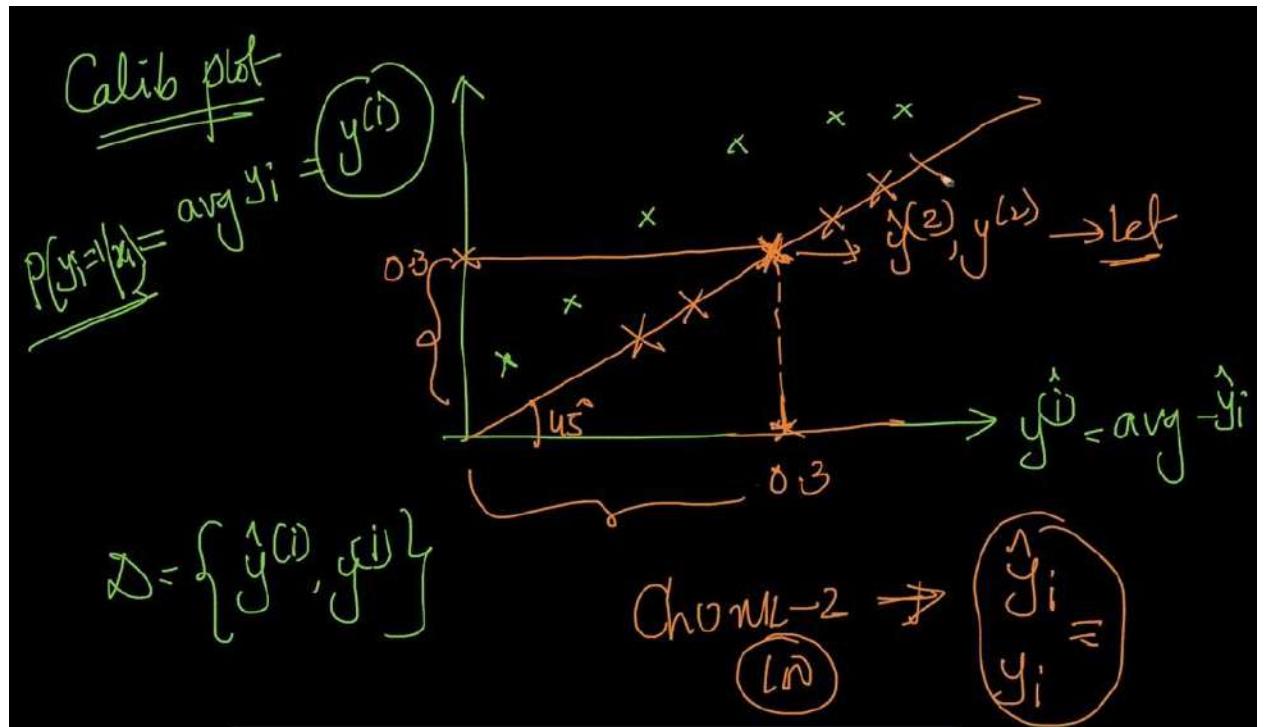
$$D_{calib} = \{\bar{y}^{(i)}, \bar{y}^{(i)}\}$$

Here $\bar{y}^{(i)}$, $\bar{y}^{(i)}$ are the averages obtained in (1) and (2) for each chunk.

In the calibration dataset ' D_{calib} ',

$y(i)$ represents $P(y_i=1|x_i)$, and $\bar{y}^{(i)}$ represents **average of $f(x_i)$'s**

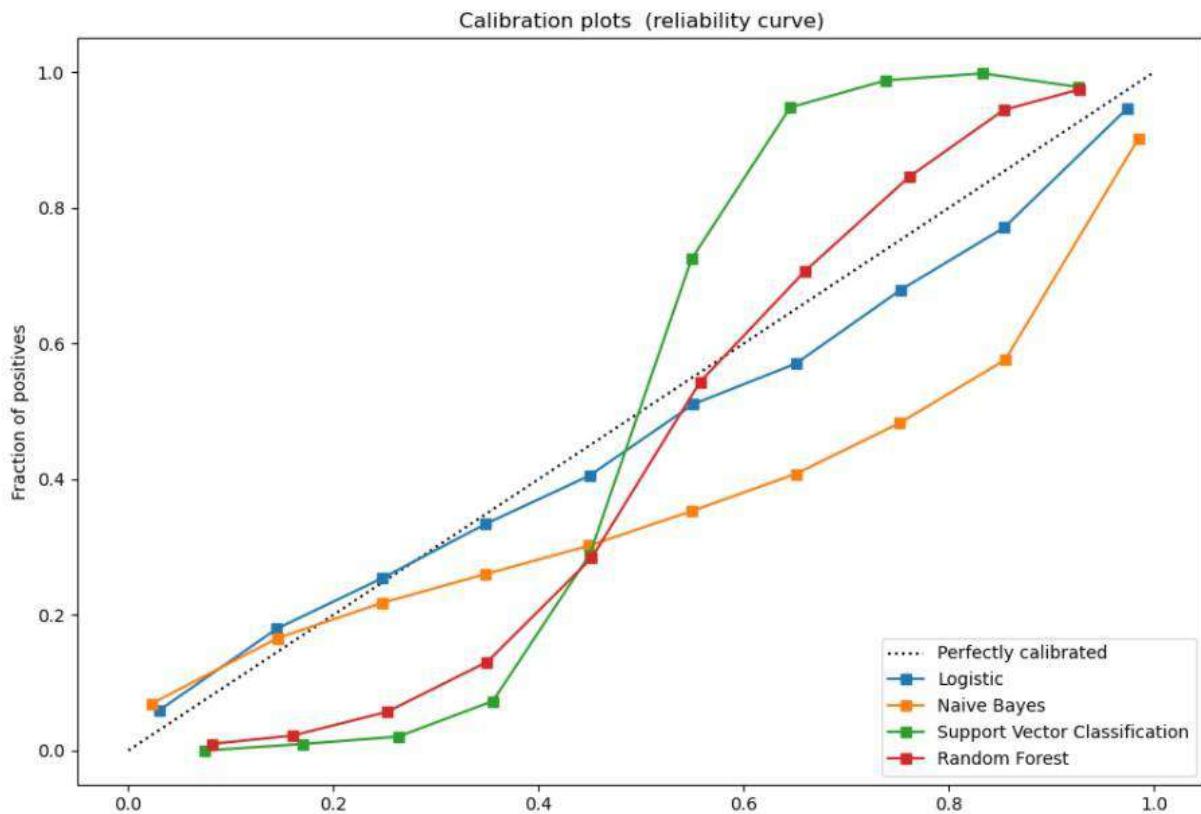
- 4) Now we have to build a 2D scatter plot using the calibration dataset ' D_{calib} ', with $\bar{y}^{(i)}$ values on the 'X' axis, and $\bar{y}^{(i)}$ values on the 'Y' axis. This plot is called the **calibration plot**.



If the plot looks like a 45-degree straight line, then it means $\text{avg}(\bar{y}^{(i)}) = \text{avg}(\bar{y}^{(i)})$ for all the chunks. An ideal model that doesn't require calibration will have all the points on a 45-degree straight line. But in reality, we do not get 45-degree straight lines. We mostly get

the plot in the form of curves, and all these curves are monotonic in nature.

We can see the comparison of the calibration curves of various ML models below



Note

The calibration plots can not be used to decide which model is better. Calibration plots only tell us how well y_i 's are calibrated for different ML models. In order to decide which model is better, we have to take any one of the performance metrics into consideration, depending on the problem we are solving.

46.3 Platt's Calibration/Scaling

One of the simplest techniques to perform Calibration is **Platt's Scaling or Sigmoidal Scaling**.

We have a calibration dataset ' D_{calib} ' = $\{y^{(i)}, y^{(i)}\}$. Now the task is to predict $y^{(i)}$, when $y^{(i)}$ is given.

$$y^{(i)} = f(x_i)$$

$$y^{(i)} = P(y_i=1|x_i)$$

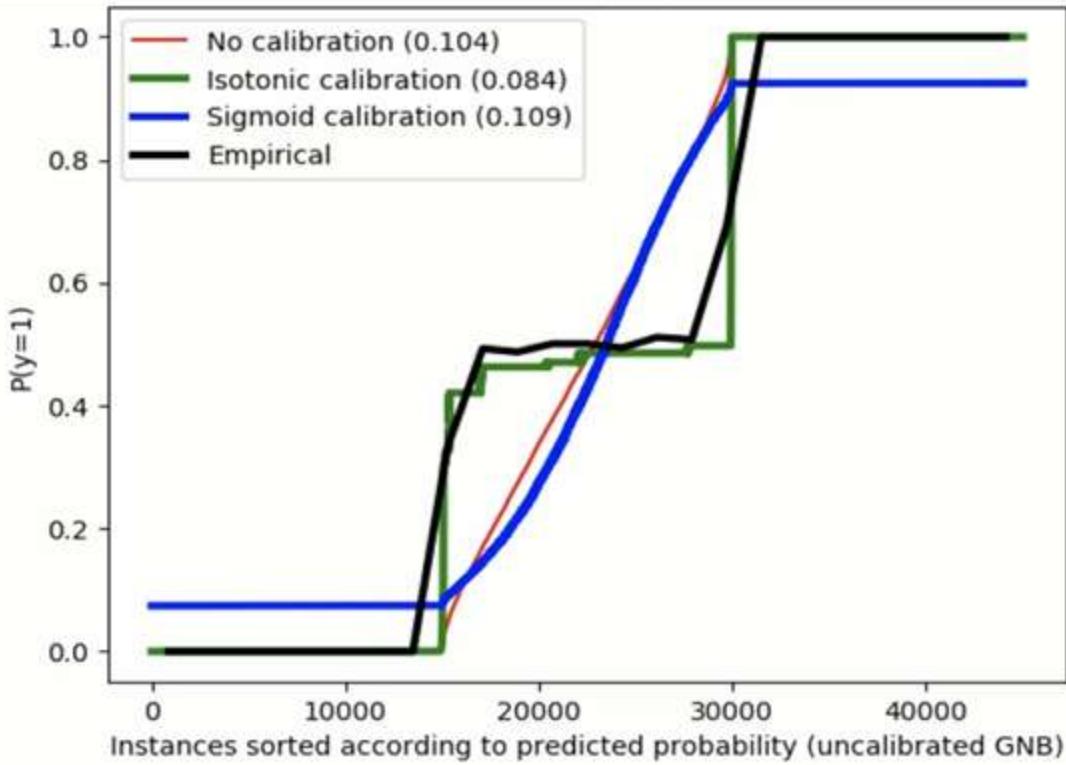
When we look at the comparison of the calibration plots for different ML models in the previous video lecture, we can observe that the calibration curves look similar to sigmoid functions.

If $y = \text{sigmoid}(x)$, then we can say that 'y' is defined as sigmoid of 'x'. Similarly, when we look at the comparison of the calibration plots, for different ML models, we can observe that the curves look similar to sigmoid curves, and $y^{(i)}$ can be obtained by applying a sigmoid function over $y^{(i)}$. Or else we can say that $y^{(i)}$ can be defined as a sigmoid function in terms of $y^{(i)}$. So Platt's scaling says

$$P(y_q=1|x_q) = 1/(1+\exp((A*f(x_q))+B))$$

In the above formula, given ' D_{calib} ', we shall replace $P(y_q=1|x_q)$ with $y^{(i)}$ and $f(x_q)$ with $y^{(i)}$, and find the best values of 'A' and 'B' by solving an optimization problem.

We have a disadvantage with Platt's scaling. Let us look at the plot below



The black curve represents the empirical data (ie., the points in ' D_{calib} '). The blue curve represents Platt's scaling calibrated data. This empirical data curve (ie., calibration curve) is nowhere similar to a sigmoid function. Platt's scaling works properly when the calibration curve is similar to a sigmoid curve. Platt's scaling fails when the calibration curve doesn't look similar to a sigmoid curve.

In case, if the calibration curve doesn't look similar to a sigmoid curve, then we have an alternative type of calibration, and that is called **Isotonic Regression**.

In the above plot, the green curve data represents isotonic regression, which is looking almost similar to the empirical data.

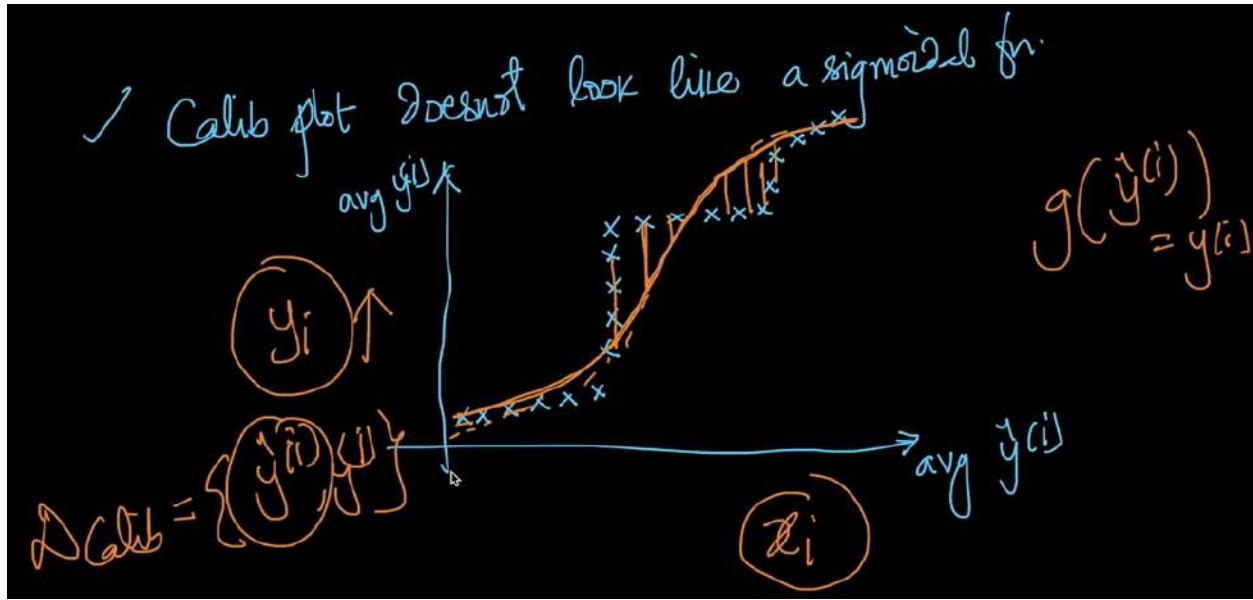
Note

If the calibration plot looks similar to a sigmoid curve or S-shape, then Platt's/Sigmoidal Scaling calibrates the data much better. Whereas if the calibration plot doesn't look similar to a sigmoid curve(if it is in different shapes), then Isotonic Regression calibrates the data better.

Platt's scaling is simpler, whereas Isotonic Regression is complex and requires a lot more data(otherwise it may overfit), but can support reliability diagrams with different shapes. Isotonic Regression is non-parametric, whereas Platt's scaling is a parametric approach.

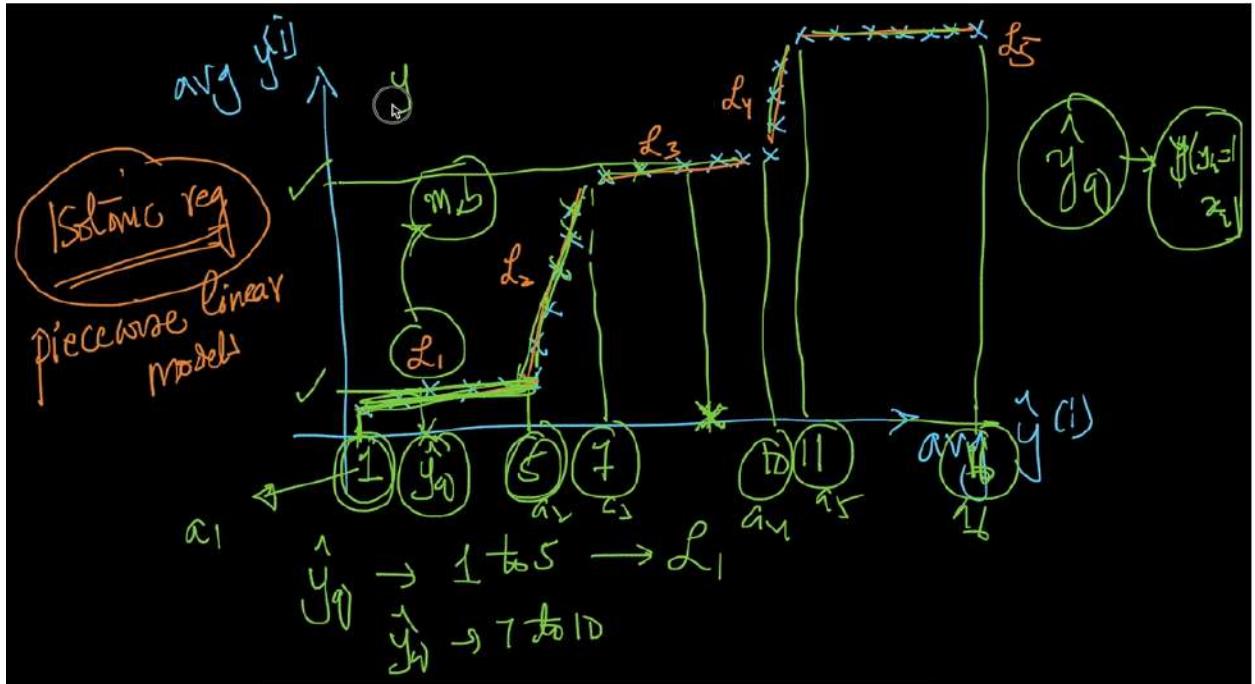
46.4 Isotonic Regression

Isotonic Regression is a very popular technique that works even if the calibration plot is not of sigmoidal shape. Let us assume the calibration plot is given to us as shown in the below figure in cross symbols.



For this kind of calibration plot, a simple sigmoidal function doesn't work as it is impossible to fit this data to a simple sigmoidal function. If we try to fit a sigmoidal function on this data, the errors will be very high.

The core concept of Isotonic regression is to break up this whole curve into multiple linear models (L_1, L_2, L_3, \dots). These individual models are called piecewise linear models.



If $y_q \rightarrow 1 \text{ to } 5 \rightarrow \text{Use } L_1$

If $y_q \rightarrow 5 \text{ to } 7 \rightarrow \text{Use } L_2$

If $y_q \rightarrow 7 \text{ to } 10 \rightarrow \text{Use } L_3$

If $y_q \rightarrow 10 \text{ to } 11 \rightarrow \text{Use } L_4$

If $y_q \rightarrow 11 \text{ to } 16 \rightarrow \text{Use } L_5$

The core idea here again is to solve an optimization problem to minimize the gaps between the predicted values and the observed values.

The whole optimization problem is about finding the values of thresholds, slopes, and intercepts of the linear lines used as the linear models. The points of calibration lie on these lines as closely as possible. This optimization works by enforcing a constraint that there should be no multiple lines. This optimization problem looks like an extended linear regression problem.

But with the Isotonic Regression, we need many more points than with Platt's Scaling. It is because, in Platt's scaling, we have only two parameters 'A' and 'B' whereas in Isotonic Regression, we have many parameters like the slopes and intercepts of each line. Hence we need more data points to estimate these parameters.

If we have a large dataset, when we split it, we can have a large amount of data points in ' D_{cv} ' and thereby in ' D_{calib} ', and then we can go for Isotonic regression.

If we have a smaller number of points in ' D_{calib} ', it is best to go with Platt's scaling.

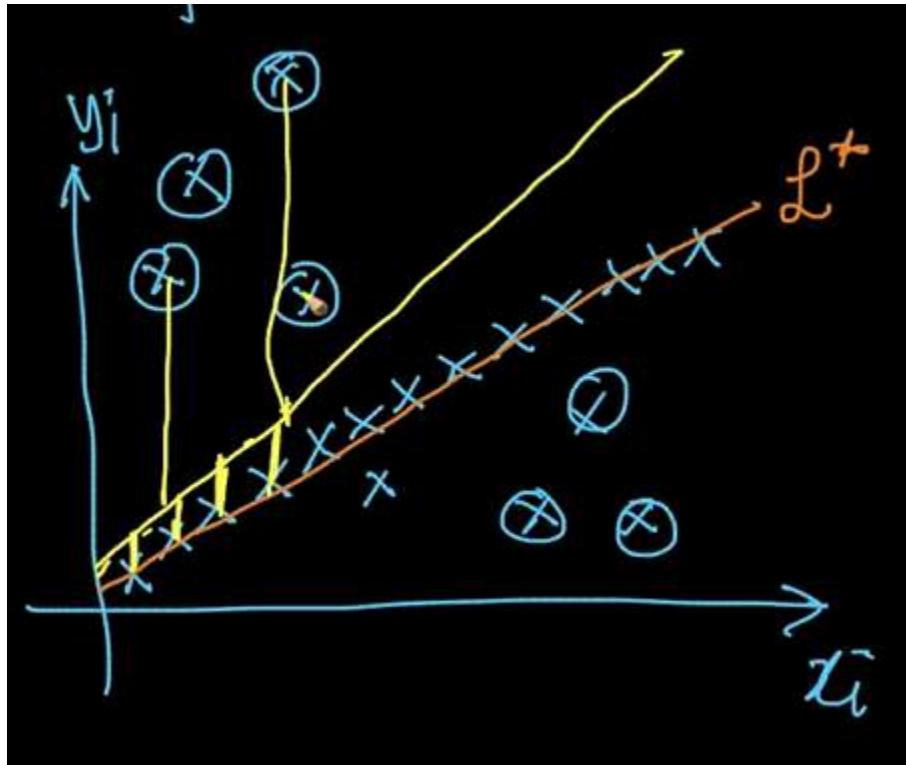
Note

As the video lecture 46.5 is all about the discussion on the code samples, we aren't providing any notes for it.

46.6 Modeling in the presence of Outliers: RANSAC

RANSAC explains how we can build a robust model in the presence of outliers. A robust model is a model that is not impacted by the outliers much.

Let us take linear regression as an example. The goal in linear regression is to find a line/plane that fits most of the points.



Let the ' D_{Train} ' be the training dataset for this problem. Here
 $D_{Train} = \{x_i, y_i\}$ where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}^d$

So if we train a model using ' D_{Train} ', then we will get a line as ' L_1 '. Linear Regression tries to minimize the distance of the points from the regression line ' L_1 '. (ie., $\min \sum_{i=1}^n (y_i - \hat{y}_i)^2$).

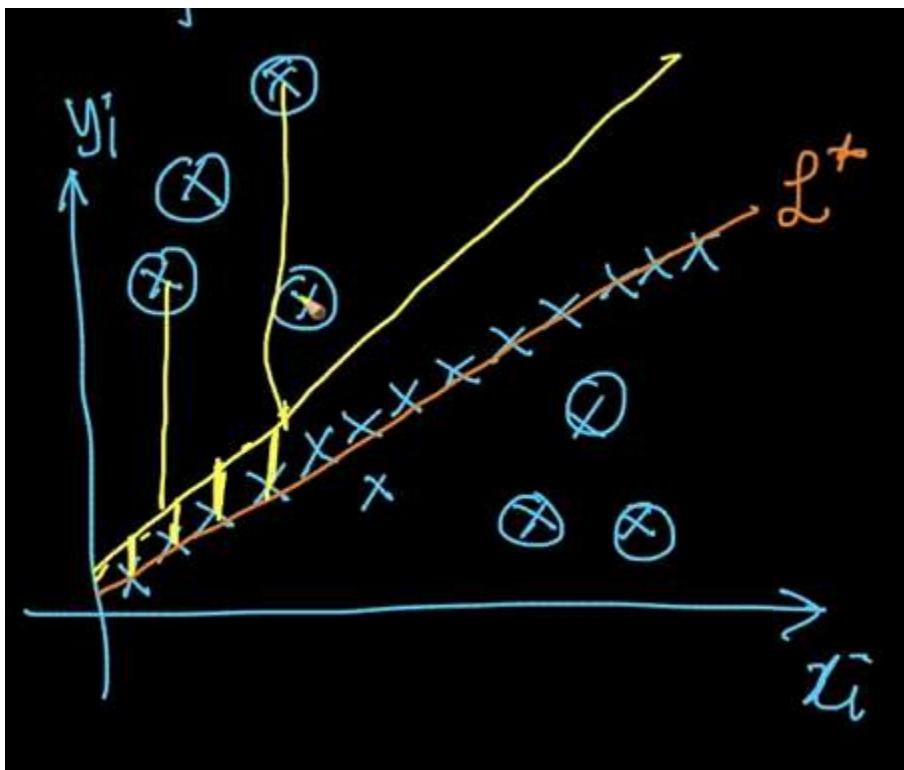
The more the extreme outliers are, the more the model pulls the regression line towards them. The regression line is pulled in that direction in which we have more extreme outliers. As this model is getting impacted by the outliers, we call it a non-robust model.

So in order to avoid the impact of outliers on the model, we have to go for the RANSAC technique.

RANSAC Procedure

- 1) Let us pick a sample dataset ' D_0 ' from ' D_{Train} ' through random sampling. Let us build a model (linear regression in this case) using the ' D_0 ' dataset and let it be ' L_0 '.

When we sample the points randomly, the probability of an outlier being a part of ' D_0 ', as the probability of an outlier being a part of ' D_0 ' reduces, the regression line is less impacted by the outliers. In this plot, the regression line has been moved to ' L_0 ' and is slightly impacted as we are building this model on a sample of data points from ' D_{Train} '.



- 2) Compute outliers based on ' L_0 ' by computing $(y_i - \hat{y}_i)$. Those points with higher values of $(y_i - \hat{y}_i)$ are considered outliers. Let these outliers be denoted as $O_0 = \{\text{outliers}\}$.

After computing the outliers, let's compute the next set of data $D_{Tr}^{-1} = D_{Tr} - O_0$.

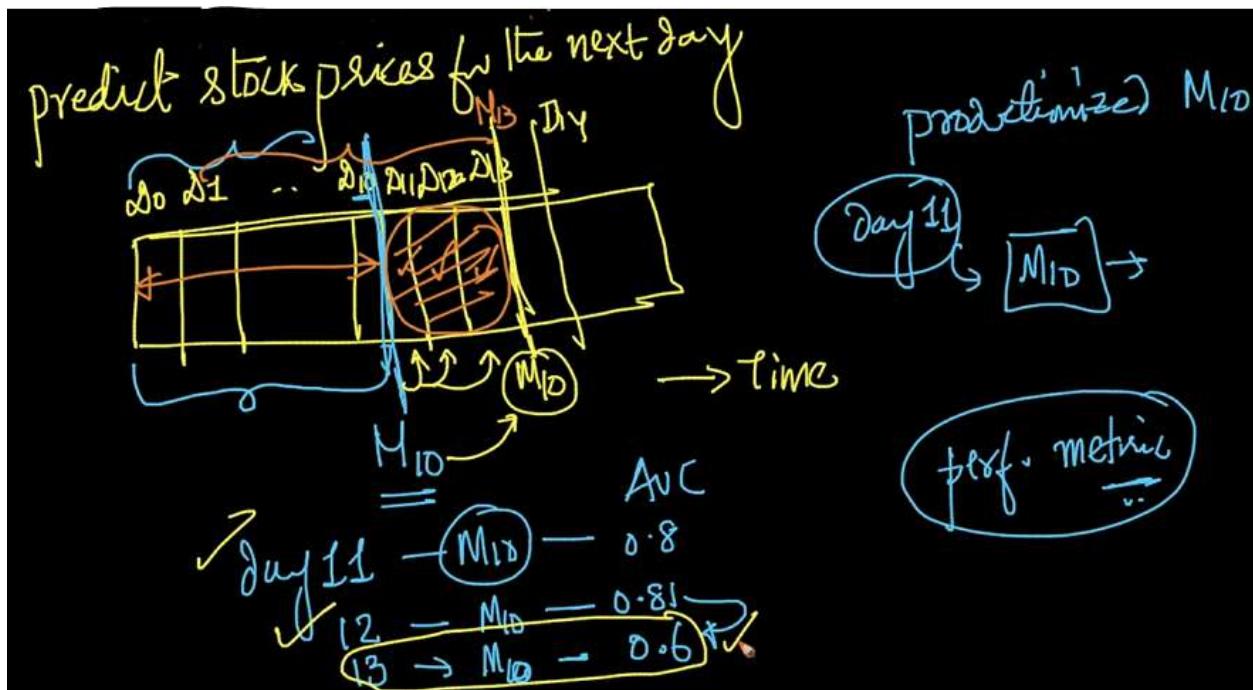
- 3) Now prepare a dataset ' D_1 ' by randomly sampling (D_{Tr}^{-1}). Fit the model on ' D_1 ' and this time let the regression line be ' L_1 ', and the outliers are ' O_1 '.

Let us find $D_{Tr}^2 = D_{Tr}^1 - O_1$. We have to repeat the process for L2, L3, L4,

In this way, at some point say ' L_i ' and ' L_{i+1} ' will look the same and there will be no outliers. Then this model is called the robust model, and this model is built using ' D_{Tr}^{i+1} ', and this model is free of all the outliers. This final model is denoted as ' L^* '.

46.7 Retraining Models Periodically

Let us assume we are working on the problem of predicting the stock prices for the next day. Every day we keep getting the fresh data, and the model is trained at the end of the day on the data so far we have (including the latest obtained data), and that trained data will be used in the production on the next day. For example, by the end of day 10, we'll have the model ' M_{10} ' trained on the data(let us denote it as D_{10}) we had till that day(including the latest data we got on the 10th day), and this model ' M_{10} ' will be in production on the 11th day.



On the 11th day, we get the new data and the predictions are made on the data we had till then including the new data obtained on the 11th day, using the model ' M_{10} '. Let us denote this data as ' D_{11} '. We get some value for the performance score. Again on the 12th day, we make predictions using the same model ' M_{10} ' on the ' D_{12} ', and we record the performance score. But on the 13th day, if we make predictions on ' D_{13} ', we observe a drop in the performance score. This drop in the performance score is because the nature/patterns in the newly obtained data (ie., the data obtained on the 13th day) is different from that of the existing data so far. In such a case, we see huge errors in the predictions made on ' D_{13} ' by the

model ' M_{10} '. In such a case, in order to minimize such huge errors, we retrain the model on the data ' D_{13} '.

Q) How to determine when to retrain the model?

Ans)

- a) If the cost of retraining is not so high, then we have to retrain the model on regular basis. If the cost of retraining is high, then day by day (or) periodically, if we notice the performance of the model to be dropping, then we have to retrain the model.
- b) If the dataset itself is changing regularly (such data is called **non-stationary data**), then we have to check if both the train and the test data follow the same distribution. If both the distributions are different, then we have to immediately retrain the model.

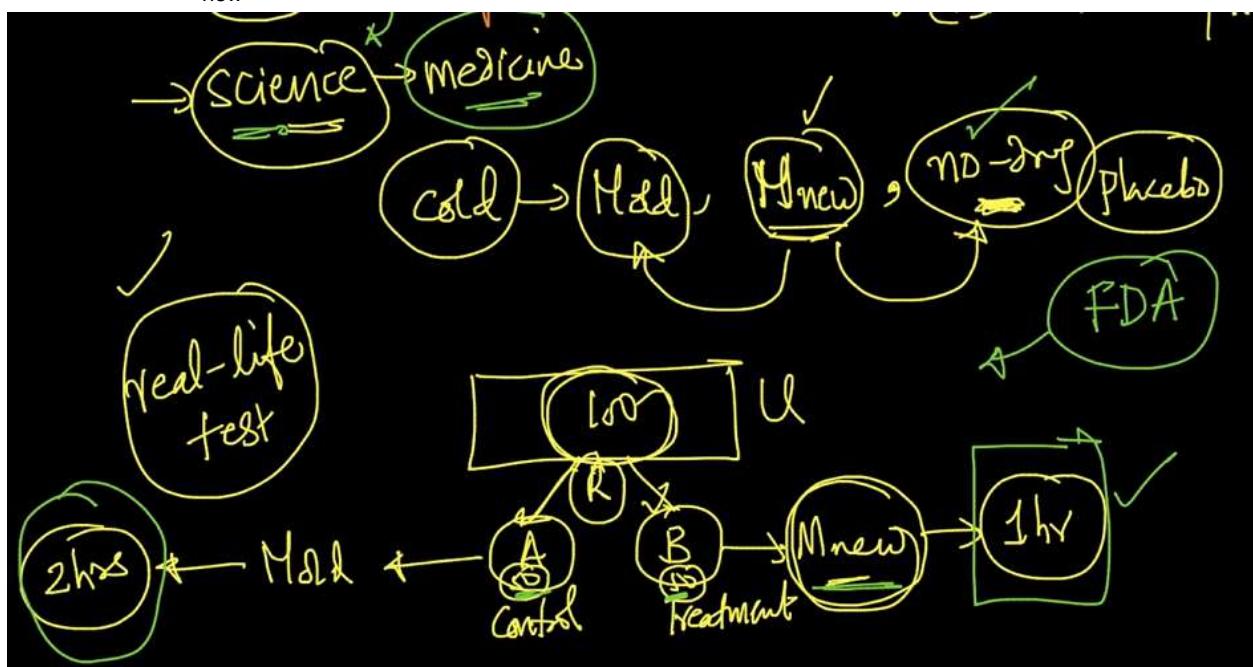
46.8 A/B Testing

A/B testing is an important phase in Machine Learning. It is also referred to as **Bucket Testing** (or) **Split run** (or) **controlled experiments**.

Example 1

Let us assume we have a medicine for a disease and let this medicine be denoted as ' M_{old} '. Let us assume a new medicine ' M_{new} ' has been introduced.

Let us assume we have a bunch of users, and these users are divided into two categories ('A' and 'B'). Let the users in category 'A' be given the old medicine ' M_{old} ', and the users in category 'B' be given the new medicine ' M_{new} '.



In case, if the medicine ' M_{old} ' takes 2 hours to cure the disease, and ' M_{new} ' takes 1 hour to cure, then the company that has manufactured ' M_{new} ' can claim the medicine ' M_{new} ' is working better, and it gets approved. This is the test process followed for approving a medicine.

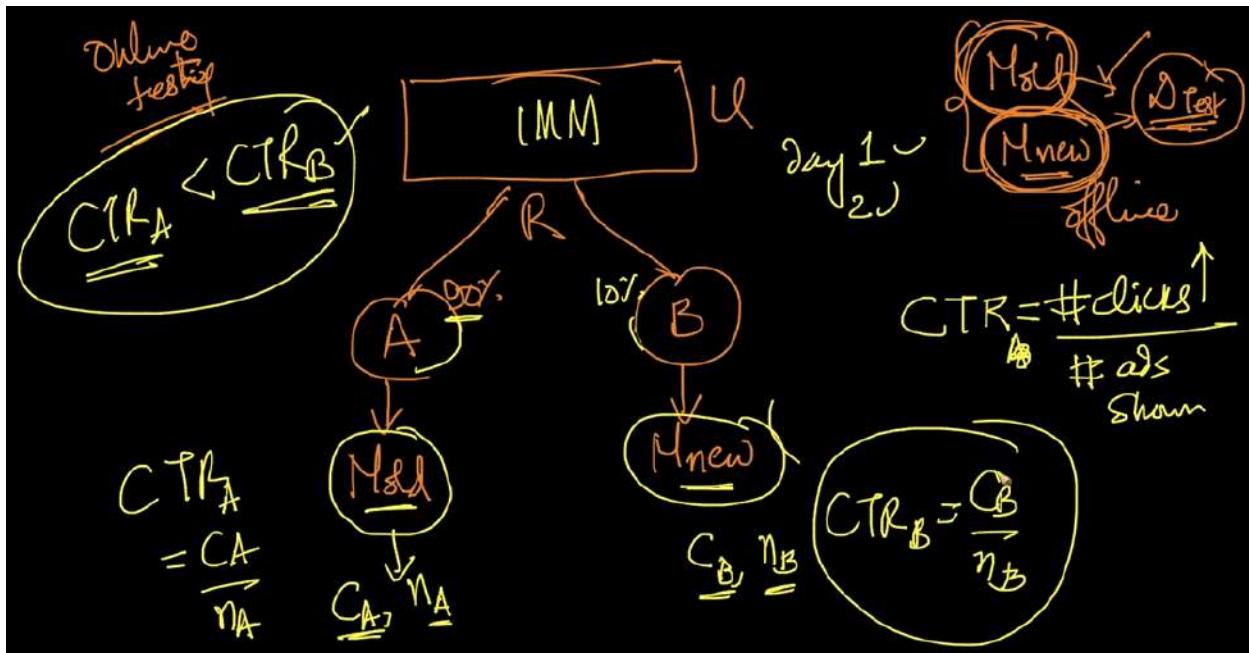
Let us now check how A/B testing is used in machine learning.

Example 2

We have to build a model for Google to predict which ad to show for a search query.

Procedure for A/B testing

There are a bunch of users and are randomly split into two groups 'A' and 'B'. Let's say 90% of the users are in 'A' and the remaining 10% of them are in 'B'. Let us also assume that we already have a model existing for this task, and let us denote it as ' M_{old} ', and we have a newly trained model and let us denote it as ' M_{new} '.



Let

c_A = Number of clicks from group 'A'

c_B = Number of clicks from group 'B'

n_A = Number of ads from group 'A'

n_B = Number of ads from group 'B'

The metric to compare these two models is the **click-through rate (CTR)**.

$CTR_A = (\text{Number of clicks we got from 'A'}) / (\text{Number of ads shown in 'A'}) = c_A / n_A$

$CTR_B = (\text{Number of clicks we got from 'B'}) / (\text{Number of ads shown in 'B'}) = c_B / n_B$

After running the models for a few days, if we find $\text{CTR}_A < \text{CTR}_B$ in most of the cases, then ' M_{new} ' that is deployed on group 'B' seems to perform better.

Initially, we have to split the users in a 90-10 ratio of old and new. If the performance of ' M_{new} ' is better at predicting which ads to show, when compared to that of ' M_{old} ', then we have to keep changing the ratios as 80-20, 70-30, 60-40, 50-50,..., 0-100.

Let us assume we have got the values as given below

$$c_A = 100, c_B = 2, n_A = 10000, n_B = 100, \text{ now}$$

$$\text{CTR}_A = c_A/n_A = 100/10000 = 0.01 (1\%)$$

$$\text{CTR}_B = c_B/n_B = 2/100 = 0.02 (2\%)$$

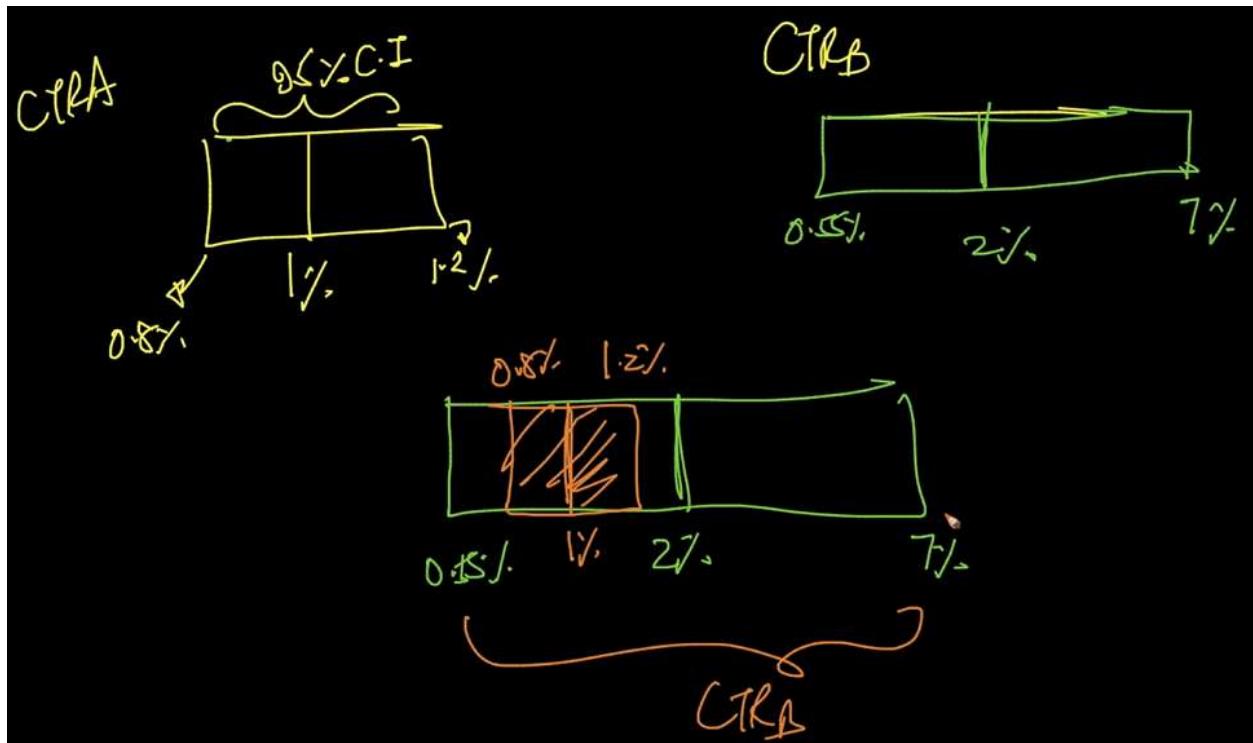
Here just by looking at the values of CTR_A and CTR_B , we generally make a conclusion that ' M_{new} ' is performing better than ' M_{old} '. But we are not supposed to make a conclusion like that because in ' CTR_A ', the value of ' n_A ' is very much larger when compared to ' n_B ' in ' CTR_B '. Those two values are not even nearby. Hence in such a case, we have to compute the confidence interval of ' CTR_A '.

For the above example (ie., with the values $c_A = 100, c_B = 2, n_A = 10000, n_B = 100$), let us say the

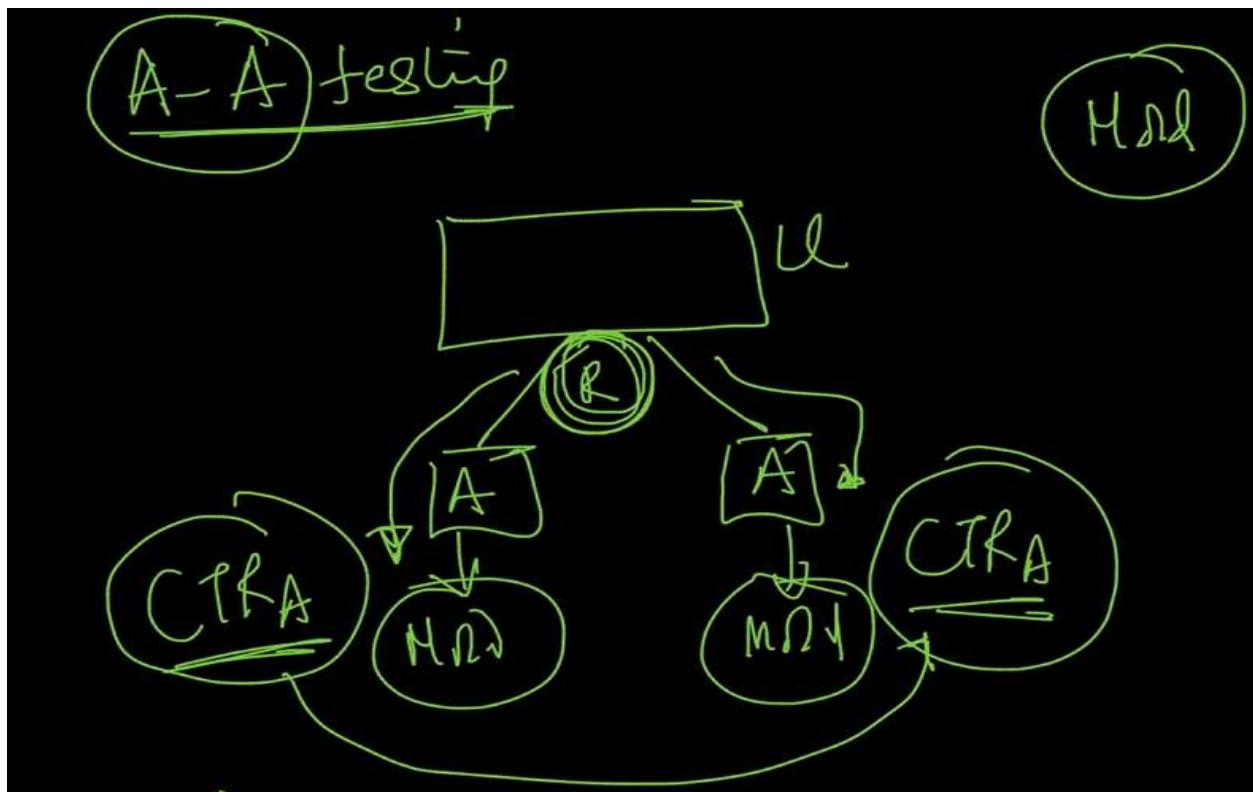
$$95\% \text{ confidence interval for } \text{CTR}_A = [0.8\%, 1.2\%]$$

$$95\% \text{ confidence interval for } \text{CTR}_B = [0.55\%, 7\%]$$

We have to check if both confidence intervals are overlapping. If they both overlap, then we couldn't say which model is performing better, and if they both do not overlap, then whichever confidence interval is wider, the model associated with it is considered to be the better performer.



A-A Testing

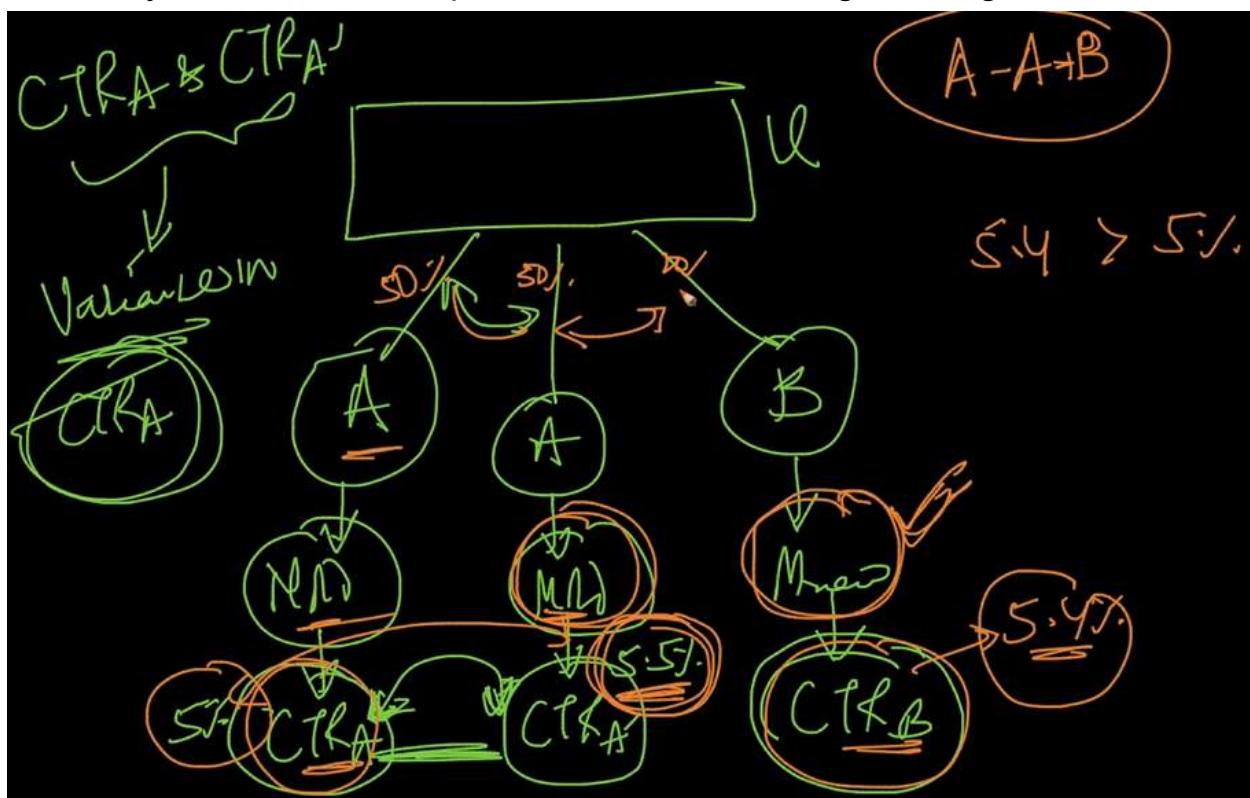


Let us assume we have a bunch of users and we are splitting them into 2 groups 'A' and 'A'. We have to run the old model ' M_{old} ' on both the groups and let the obtained CTR scores be denoted as CTR_A and CTR_A' respectively. Here CTR_A and CTR_A' are not the same because the data/users in one group are different from the data/users in the other group.

Typically used A-A Testing working procedure

Let the whole universe of users be divided into 3 groups. They are A, A, and B. Let us now train the model ' M_{old} ' separately on the users present in 'A' and the other 'A', and train the model ' M_{new} ' on the users present in 'B'.

Let the CTRs obtained be denoted as CTR_A , CTR_A' and CTR_B . Even though the same model was used 3 times, the data present in each of the groups is different. The difference between CTR_A and CTR_A' will tell us how differently the same model performs if there is a slight change in the data.



When we look at the CTR scores of these 3 models, we blindly should not conclude that the new model ' M_{new} ' performs better when compared to the old model ' M_{old} ', just because of $CTR_A < CTR_B$. This

difference in performance is due to randomness while splitting the users. So as the randomness changes, the CTR also changes. This is called **A-A-B testing**.

46.9 Data Science Life Cycle

1) Understand Business Requirements

Define the problem, understand the customer and their business for whom we are building the project.

The prime focus should be on working on what adds much value to the business and helps in making decisions to move the business in the best way.

Working on something that doesn't add much value to the business is of no use.

2) Data Acquisition

The process of data acquisition is mostly ETL. The processing is done mostly using SQL.

The major sources/storages of data used are the databases, data warehouses, log files, and Hadoop/spark systems.

3) Data Preparation

It involves data cleaning and data preprocessing. The first 3 phases are the time-consuming phases.

4) Exploratory Data Analysis

After the data preparation, we start plotting the data, visualizing the data using various visualization techniques. We also apply various statistical techniques such as Hypothesis testing, etc and at the end, we slice and dice the data by splitting the data and trying to understand what's happening, which feature is helping more to achieve our goal.

5) Modeling, Evaluation, and Interpretation

Modeling deals with building various models such as regression, classification, clustering, etc that perfectly suits the given business problem.

Evaluation deals with defining KPI(ie., all the performance

metrics for our models are defined in this phase for the best results) and adding huge business value.

6) Communicate Results

The results should be communicated in a clear and simple way to all the stakeholders, business customers, and top-level managers(ie., non-technical people)

The communication of results should be in such a way that should convince the business customers and make them believe, this is the best model for the given business problem.

7) Deployment

After getting approval from the higher-level management for the results that were communicated, we should go ahead with the deployment of the model.

Deployment deals with software engineering. This phase is worked out sometimes by the Machine Learning Engineers, sometimes by the software development engineers, and sometimes by the distributed system engineers.

8) Real-world Testing (A/B Testing)

The real-world testing deals with verifying the results of the data analysis, models whether all the results we have obtained are really meaningful in the production environment.

At the end of A/B testing, we have to measure the true business impact. The result of all the work done in the previous phases of this life cycle will become useful/useless on the basis of the measure of the true business impact.

9) Customer/Business

After achieving a good business impact, we have to go back to the business/customer and convince them with the experimental data and the solution we have. At the end of the day, it is our solution that has to add huge value to the business and satisfy the customer.

10) Operations

This phase deals with retraining the models, handling failures in the model, and defining an entire process of how to retrain the models and handle failures. This is called the **operationalization of models**.

11) Optimization

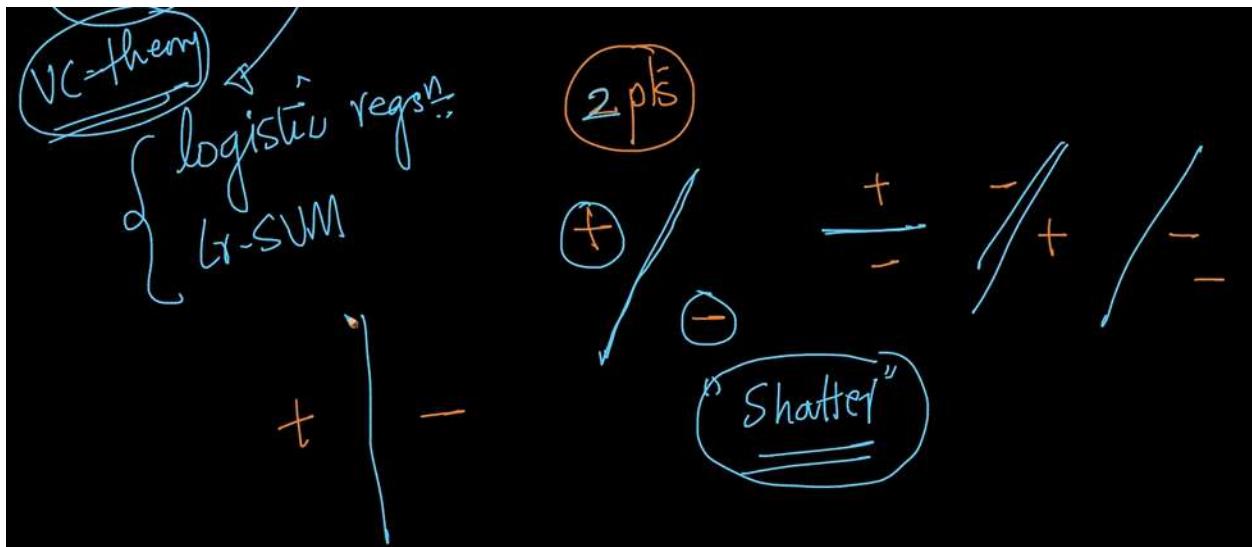
The model that was built so far is the first-cut model. In this phase, we go back and try to improve the models (by moving from one model type to another), acquiring more data, adding more features, and optimizing the production code. This phase is a continuously running phase.

46.10 VC Dimension (Vapnik Chervonenkis Dimension)

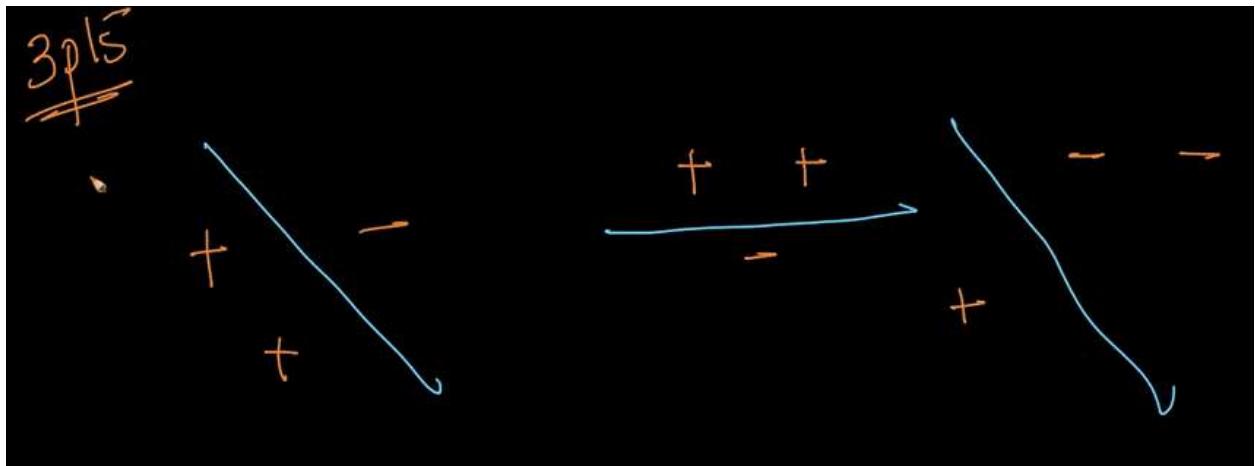
Is there a numeric value to find out how powerful is a class of models? (like how powerful the linear models are, how powerful RBF-SBM models are, how powerful the boosting models are, etc). This is an important aspect of theoretical machine learning.

The practical approach to find out which model is powerful is by taking a bunch of datasets, and building the models on them, and finding out the performances. The theoretical approach is by giving a numerical value to each of the models which indicates how powerful each model is.

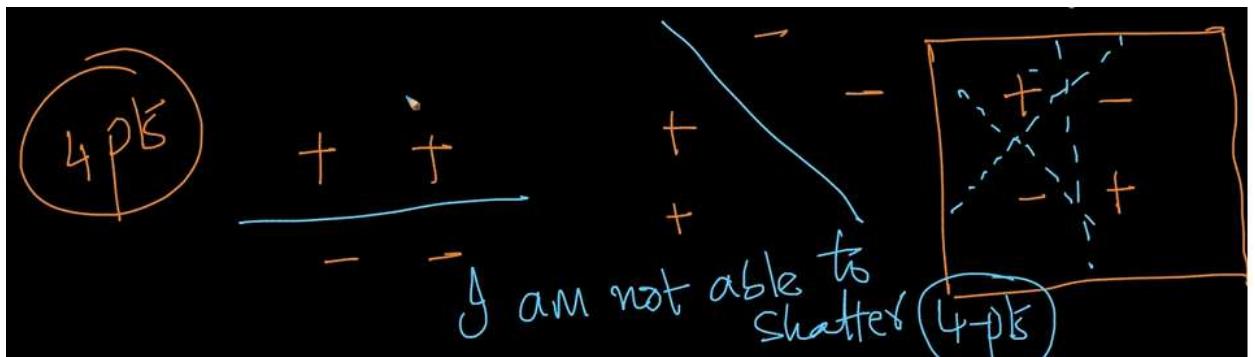
For example, let us consider the linear models (where line/plane/hyper plane are the decision surfaces). If we have only 2 points (1 +ve and 1 -ve), then irrespective of the way the points are shattered, we can create a decision surface easily.



Similarly when we have 3 points, we still can create a decision surface, irrespective of the way the points are shattered.



In case of 4 points, sometimes, it is possible and sometimes it is not possible to create a decision surface that perfectly classifies all the points.



There exists at least one alignment where these points cannot be separated perfectly.

So the VC-Dimension value for a linear model = 3

(ie., maximum of upto 3 points can be shattered by a linear model, for all possible alignments)

VC Dimension is a measure of the capacity of a space of functions that can be learned by a statistical classification algorithm. It is defined as the cardinality of the largest set of points that the algorithm can shatter.

In case of a Boosting classifier, if we have

Number of Base classifiers = T

Class of each of these Base classifiers = B

VC Dimension of each of these base classifiers = D

Then the VC Dimension of the set of all such classifiers = $T^*(D+1)^*(3*\log(T^*(D+1))+2)$

In RBF Kernel SVM,

VC Dimension = ∞

If $\text{VC-Dimension}(M_1) > \text{VC-Dimension}(M_2)$, then theoretically we can say that the model ' M_1 ' is power than ' M_2 '.

By the VC Dimension value, we can say RBF-Kernel SVM is so powerful, but it doesn't imply that RBF Kernel SVM is the best model in practice.

Use of VC Dimension

The VC dimension can predict a probabilistic upper bound on the test error of a classification model.

$P(\text{test error} \leq \text{train error} + \sqrt{(1/N)*[(D*(\log((2*N)/D)+1)) - \log(\eta/4)]}) = 1-\eta$

($0 \leq \eta \leq 1$)

Where

$D \rightarrow$ VC Dimension of the classification model

$N \rightarrow$ Number of points in the training dataset

The above formula is valid only if $D \ll N$. If ' D ' is larger, then the test error may be higher than the training error, and this happens due to overfitting.