PCA:

A) As FEATURES increases in a particular/group of models, CURSE to the DIMENSION takes place, due to which the model OVERFITS, hence LESS ACCURATE & Model performance degrades!

E.g.: House: There are lot of features one sees before buying but at once they could not decide.

B) Ways to remove CURSE of DIMENSION:

i) Feature Selection: select imp features!

ii) Dimensionality Reduction: PCA: feature extraction (decreasing dimensions)

C) Why dimension reduction?

i) Prevent --> curse of dimensionality

ii) Improve the performance of the model

iii) Visualize and understand the data!

D) Feature Selection:

x	y	x↑ y↑	x↓ y↑
(i/p)	(o/p)	x↓ y↓	x↑ y↓
-	-		
-	-	linear -	inverse linear -
-	-	relation	relation
-	-		

w.r.t x predict y

E) covariance (cov):

$\text{cov}(x, y) = \Sigma(x_i-\mu)*(y_j-v) / n$

when cov is +ve: linear relation

when cov is -ve: inverse linear relation

when cov is or near to 0: there's no relationship between x & y (scatter plot in circular way)

*highly +ve = super important features!

F) Pearson correlation (corr):

corr(x,y) / σx*σy: range = -1 to +1

more towards +1 the more +ve correlated x & y is!

more towards -1 the more -ve correlated x & y is!

more towards 0 no relationship between x & y is!

G) Example:

Housing Dataset:

There's 3 features: 2 Independent feature & 1 Output feature!

House Size	Fountain Size	Price
-	-	-
-	-	-

comparing HOUSE SIZE with the PRICE, we get two conditions:

i) big house, more price

ii) small house, less price

which shows, it's forming linear or inverse linear relation, cov will be -ve or +ve, corr from -1 to +1!

comparing FOUNTAIN SIZE with the PRICE, we get one condition:

i) that's, it does not affect a lot

which shows, there's no relation, cov & corr will be 0!

H) Feature Extraction:

Room Size	No.of Rooms	Price
-	-	-
-	-	-

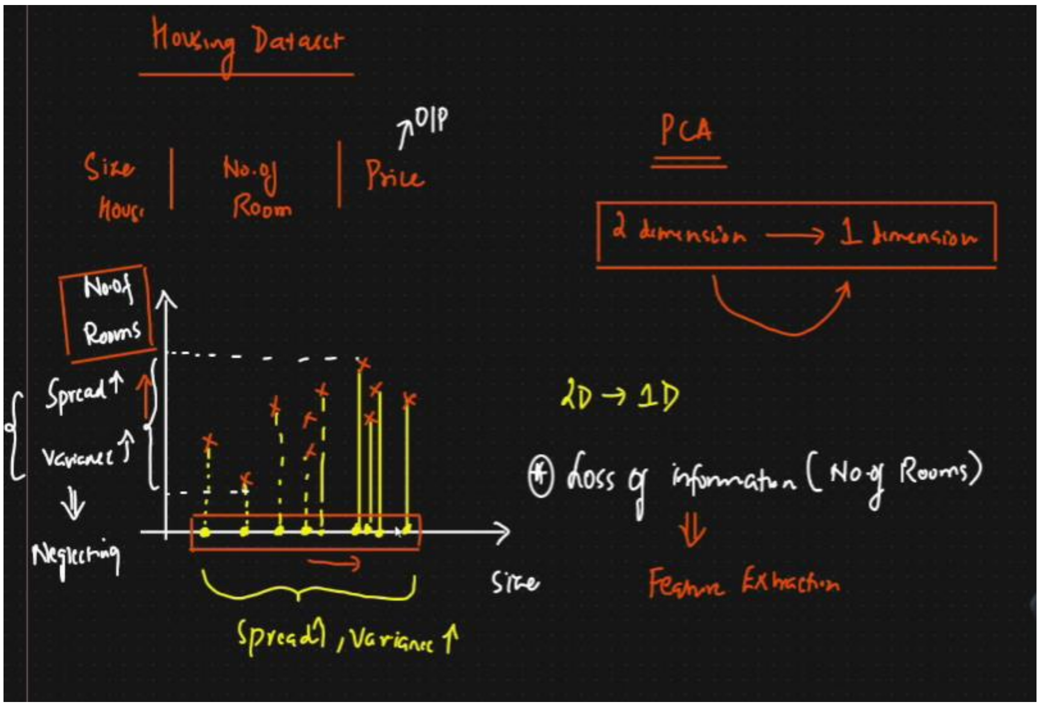
over here, both the independent features are important unlike above example

and so, we cannot perform feature selection instead will do feature extraction!

2 feature --> 1 feature (Dimensionality Reduction: PCA)

will perform transformation by which will combine both the independent feature (room size & no.of rooms), due to which we extracted and gonna have 1 feature, named: House size!

I) Geometric Intuition behind PCA (dimensionality reduction):

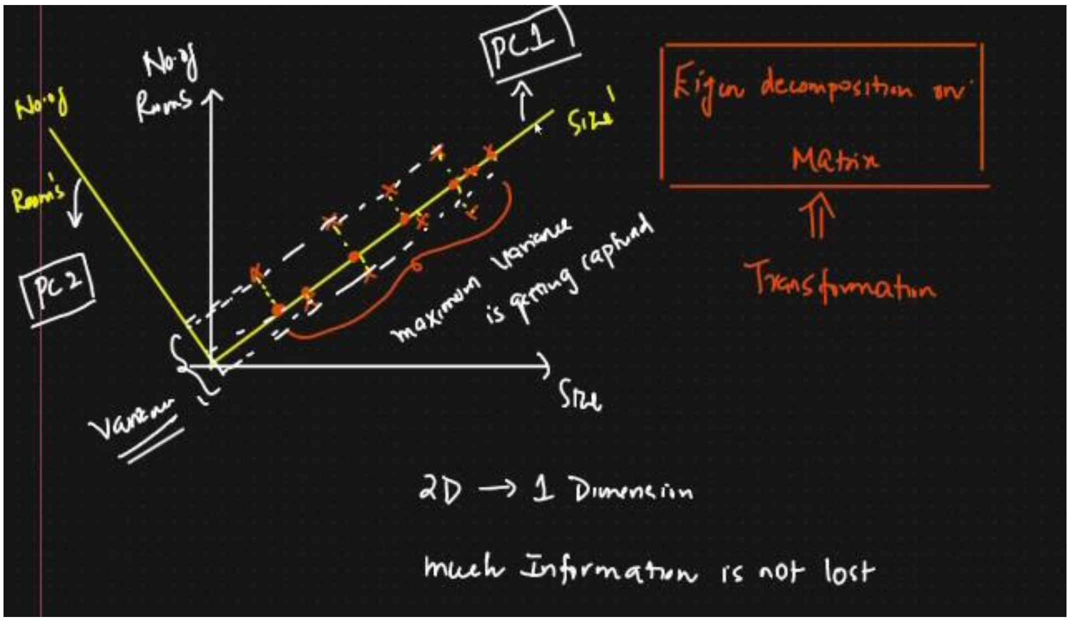


In the pic we can observe that, how big is the house? Is compared with the no.of rooms to extract valuable information!

And so, we plot some data, we find spread & variance increases! Later to convert 2D to 1D, extend the line on 1 of the plots!

As we do that, we have successfully converted 2D into 1D; but have lot many information as well from the no.of rooms & so our model won't perform well!

So, to avoid loss of the information, will perform a transformation named: eigen decomposition on matrix!


No of Rooms
PC1
Size'
Eigen decomposition on Matrix
Transformation
No of Room's
PC2
maximum variance is getting captured
Size
Variance
2D → 1 Dimension
much Information is not lost

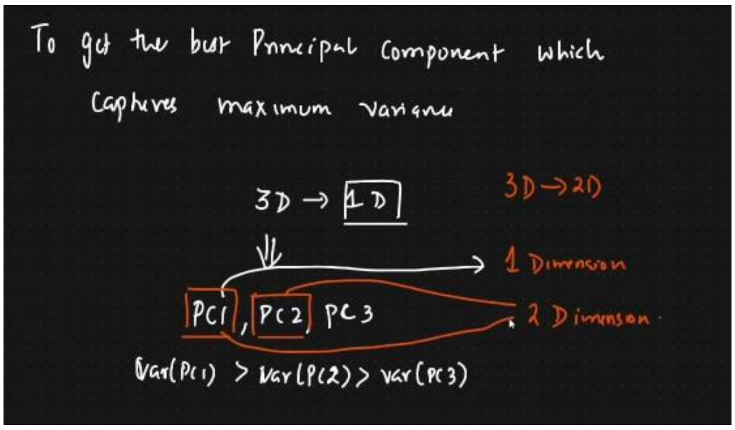
Later will project the information and a new axis gets formed & spread will be less whereas maximum variance is been captured & loss some information!

2D to 1D is successfully done!

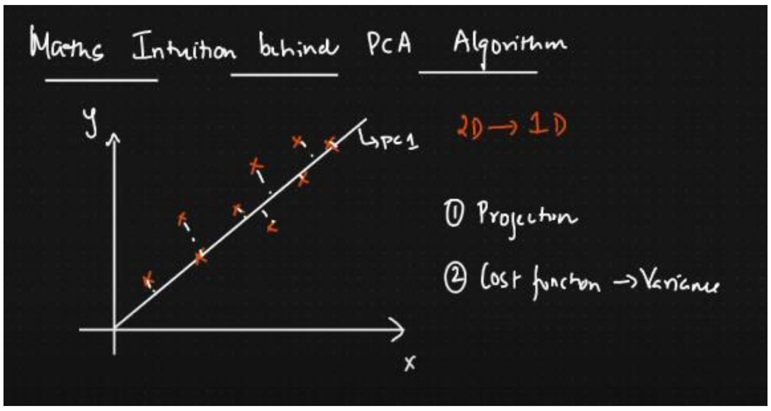
The 2 projected lines will be denoted by PC1 & PC2, PC1 will capture more amount of information than PC2.

Final goal will be to get PCA with the maximum variance with less loss of information!

J) 3D to 1D:


To get the best Principal Component which captures maximum variance
3D → 1D
3D → 2D
1 Dimension
PC1, PC2, PC3
2 Dimension
Var(PC1) > Var(PC2) > Var(PC3)

K) Maths intuition behind PCA algorithm:

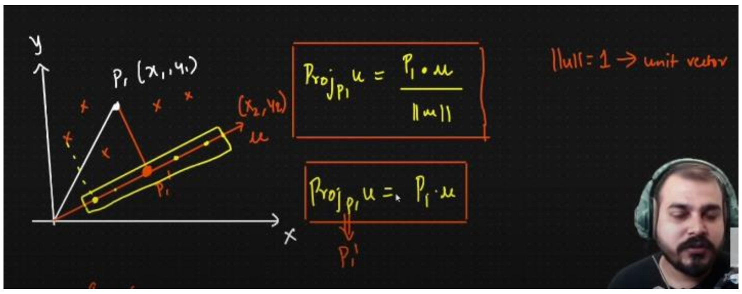

Maths Intuition behind PCA Algorithm
y
x
PC1
2D → 1D
① Projection
② Cost function → Variance

To verify, whether it's best or not, will do:

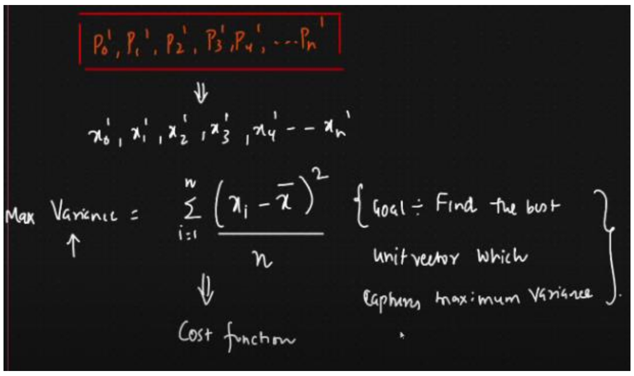
1) Projection &
2) Cost function → related to variance
3) And the goal: get maximum variance!

Steps:

1) P1(x1, y1) → point 1 → a vector
2) µ → unit vector
3) project P1 to µ, we get P1'
4) Later, will apply the formula & understand what's P1', refer below figure!


y
x
$P_1 (x_1, y_1)$
(x_2, y_2)
u
P_1'
$\text{Proj}_{P_1} u = \frac{P_1 \cdot u}{\|u\|}$
$\|u\| = 1 \rightarrow$ unit vector
$\text{Proj}_{P_1} u = P_1 \cdot u$
P_1'

5) Likewise, will get n no.of projections! → P0', P1', P2', P3',.., Pn'; called as scalar values!
6) Scalar values are talking about the distance from the origin till a particular point!
7) Once we have the scalar value, it become easy to compute the variance using variance formula which helps us to find out the best unit vector (µ) which captures maximum variance. → cost function!


$P_0', P_1', P_2', P_3', P_4', \ldots P_n'$
$x_0', x_1', x_2', x_3', x_4' \ldots x_n'$
Max Variance = $\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}$
Cost function
{Goal: Find the best unit vector which captures maximum variance}

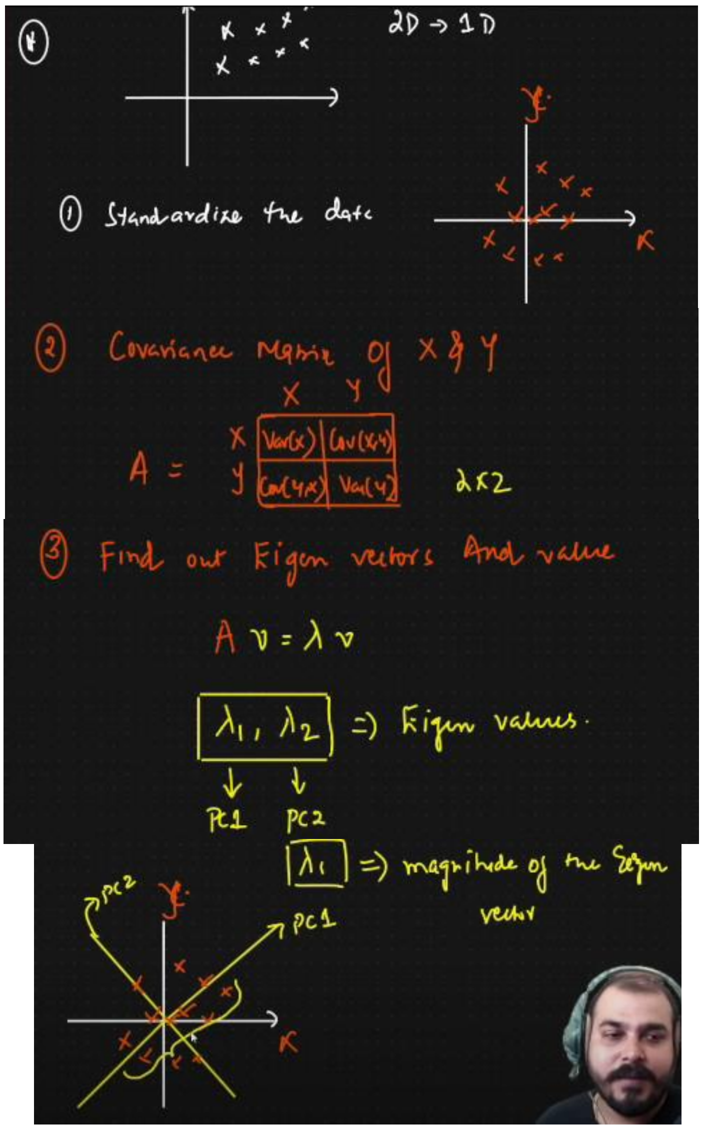
This method is bit lengthy & so we cannot keep on doing this for the larger dataset to get a perfect unit vector. Instead, will use Eigen decomposition (Eigen vectors & Eigen values)!

L) Eigen vectors & Eigen values:

Steps:

1) cov matrix between features:
2) finding Eigen vectors & Eigen values from this cov matrix
   Av = λv; (where, A→ a matrix, v → vector v, λ → eigen value) → **Linear Transformation of Matrix**
3) **Eigen vector → Eigen value → max magnitude of the max Eigen vector → PCA → capture the maximum variance**
   Whichever, Eigen vector will be largest one, which means the one having high Eigen value, talking about the max magnitude of the max Eigen vector, use as a plane of PCA & this will capture the maximum variance!

M) Example: 2D to 1D:



(4) 2D → 1D

① Standardize the data

② Covariance Matrix of X & Y

$$A = \begin{array}{c|c|c} & X & Y \\ \hline X & Var(X) & Cov(X,Y) \\ \hline Y & Cov(Y,X) & Var(Y) \end{array} \quad 2\times 2$$

③ Find out Eigen vectors And value

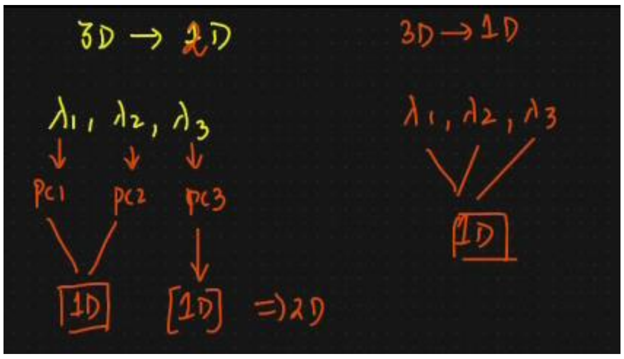
$$A v = \lambda v$$

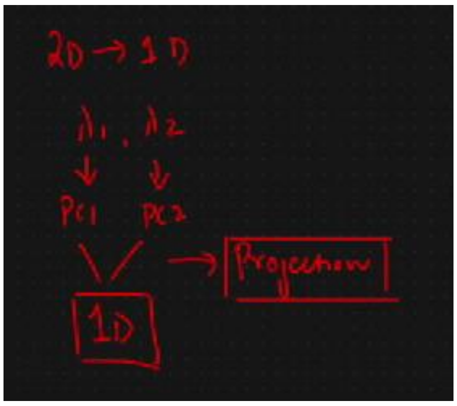
$$\boxed{\lambda_1, \lambda_2} \Rightarrow \text{Eigen values.}$$

$\lambda_1 \downarrow$ PC1, $\lambda_2 \downarrow$ PC2

$\boxed{\lambda_1} \Rightarrow$ magnitude of the Eigen vector

Another example:


3D → 2D
λ1, λ2, λ3
PC1 PC2 PC3
1D
1D ⇒ 2D
3D → 1D
λ1, λ2, λ3
1D


2D → 1D
λ1, λ2
PC1 PC2
1D
Projection