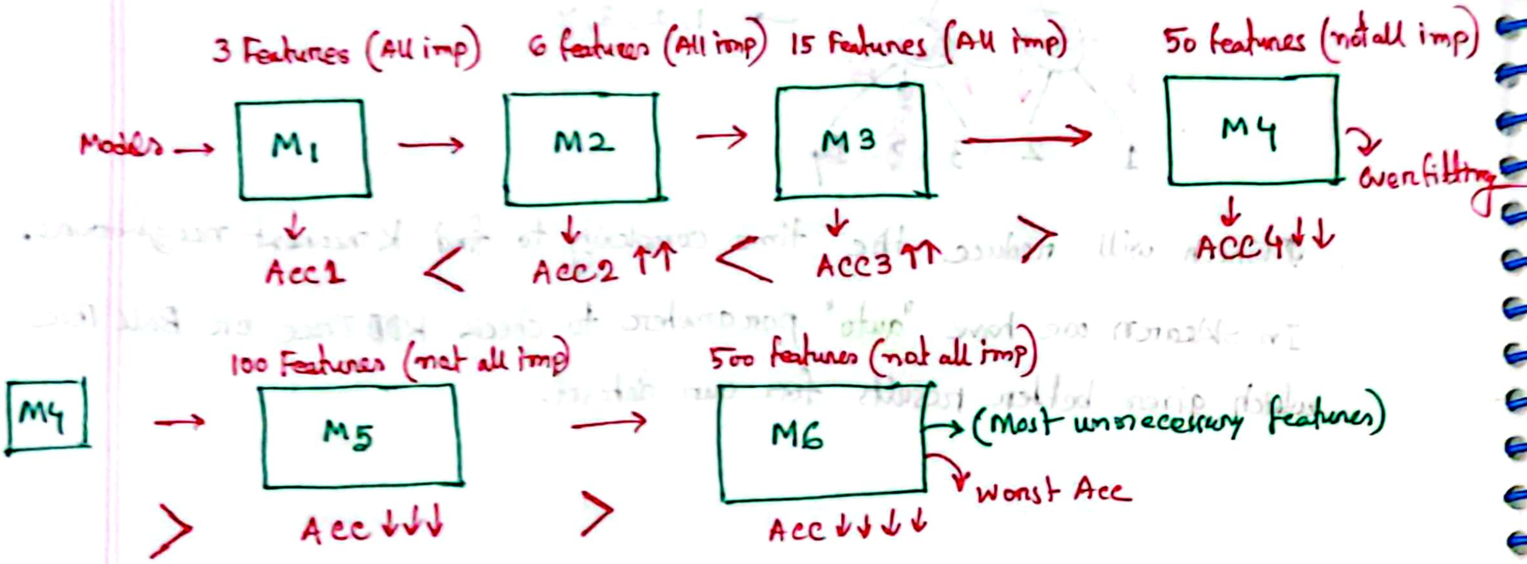
Curse of Dimensionality:

Suppose our dataset has 500 features.

If we train the models with



In short the important and small number of features that actually is necessary in order to train your model, you add them and you can have a good accuracy but if you start adding unnecessary features to your training model, the model starts getting overfitted and the Acc of the model start decreasing. More the dimensionality is increasing after certain features added, the acc keeps decreasing. This is the curse of dimensionality.

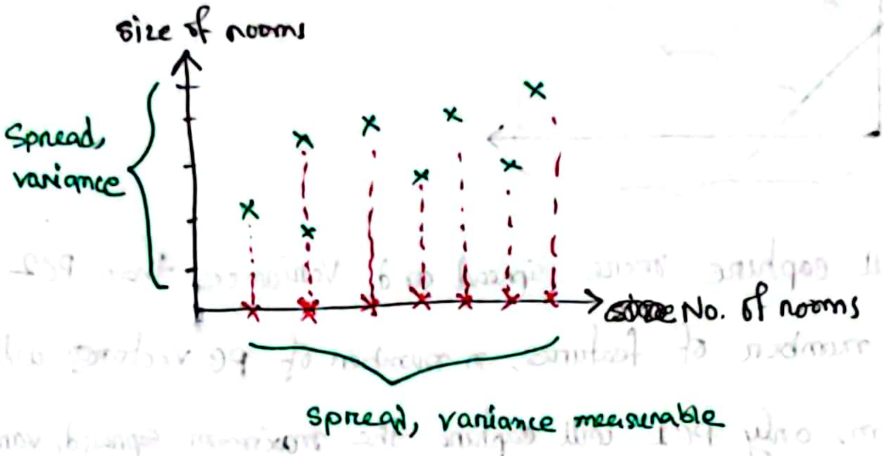
There are two ways to remove curse of dimensionality.

- Feature Selection (Choose the most important features)
- Dimensionality Reduction (PCA)
  - ↳ Feature Extraction convert into
  - ↳ Modifying 500 features to 20 features.

Geometric Intuition of PCA: [For Dimensionality Reduction]

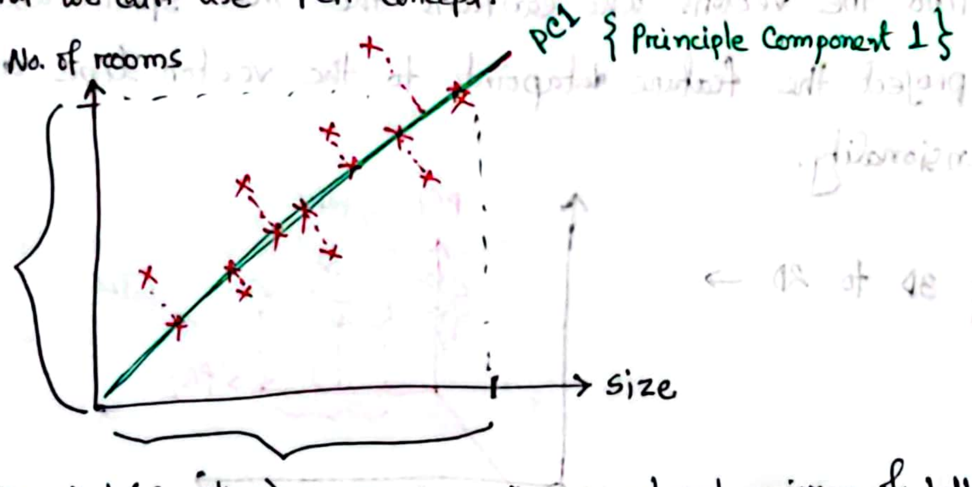
Features → size of rooms, no. of rooms, price

First we can do feature selection. Suppose we only took the no of rooms and not taken the size of the rooms.



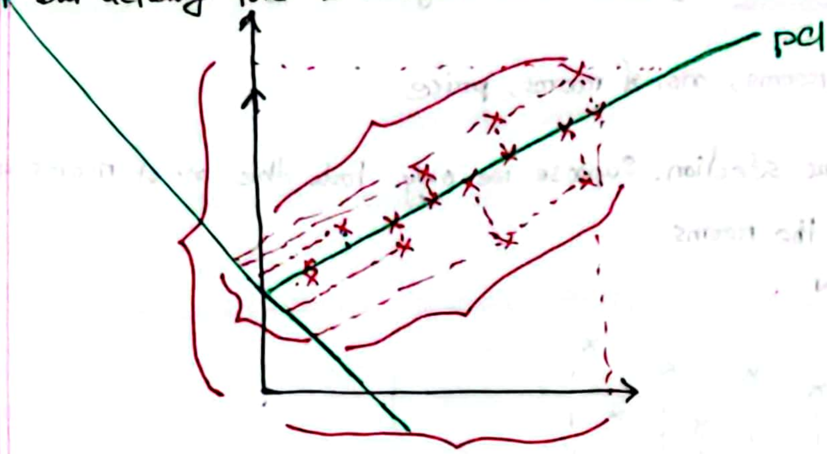
So, we are basically ignoring the spread, variance of size of the rooms and taking only the spread, variance of the No. of rooms. By selecting features like this we can reduce the dimension from 2D to 1D.

But what if we can't ignore the features. We have to keep both of them but at the same time, we need to reduce dimensionality.
For that we can use PCA concept.



By doing that (Projecting) we can keep the spread and variance of both of the axis.

But actually for 2 features 2 PC vectors will be generated (PC1, PC2)



But PC1 will capture more spread and variance than PC2.

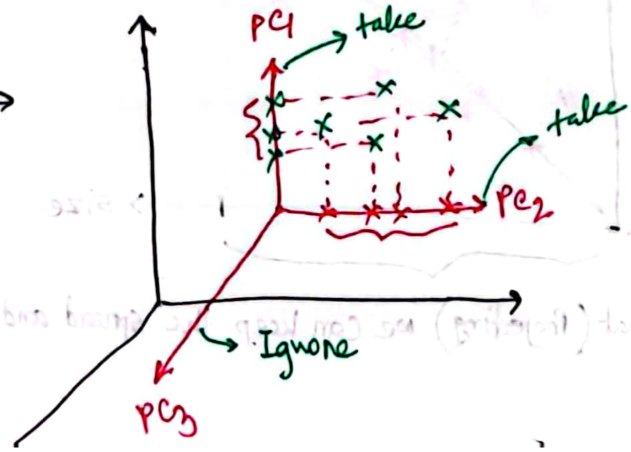
So, for n number of features, n number of PC vectors will be generated but from them, only PC1 will capture the maximum spread, variance.

So, what we are doing is, we are extracting information ~~from~~ (spread, variance) from n features and making a feature which can hold all the information.

We are reducing dimensions by feature extraction (PCA) Technique.

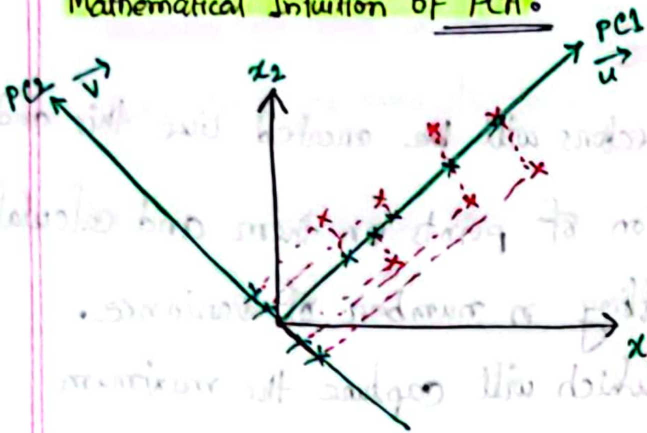
So our aim in PCA is, to apply some transformation on the features so that we can find the vectors who contains the max spread and variance. Then we project the feature datapoints to the vector line and reduce the dimensionality.

Converting 3D to 2D →



You can convert from n dimensions to $\{n-1, n-2, \ldots 1\}$ dimension using PCA.

Mathematical Intuition of PCA:



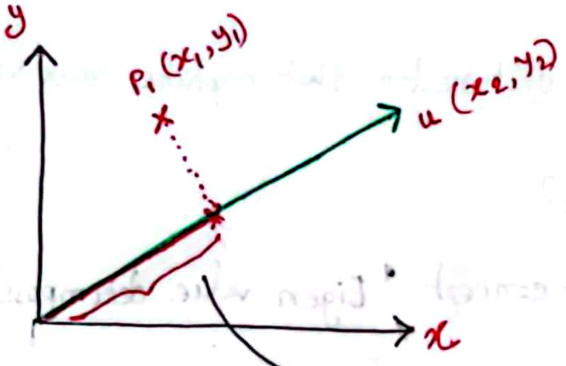
target $2D \rightarrow 1D$

$PC1 > PC2$

In terms of variance and spread.

There are two things we have to understand.

- Projections
- Optimization (Max Variance to find)



$$\text{Proj}_{P_1} u = \frac{P_1 \cdot u}{\|u\|} \rightarrow \|u\| = 1 \quad \text{(Unit vector)}$$

$$= P_1 \cdot u = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \cdot \begin{bmatrix} x_2, y_2 \end{bmatrix}$$

$$= x_1 x_2 + y_1 y_2$$

$=$ scalar value

like this we can calculate distance of all the projections on $\vec{u}$.

Suppose the distance for P_0 is P_0'

for n number of points we will get $[P_0', P_1', P_2', P_3', \ldots \ldots P_n']$ scalar distances.

from which we will capture variance.

$$\boxed{P_1', P_2', P_3', \dots, P_n'}$$

$\Downarrow$

Scalar values

$\Downarrow$

Capture variance

So, for n number features n vectors will be created like this and we can find ~~their~~ the projection of points on them and calculate variance. So, we are also getting n number of variance. Lastly we will select that vector, which will capture the maximum variance.

$$\text{Max Variance} = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n} \Rightarrow \text{cost Function}$$

$$x_i = [P_1', P_2', P_3', \dots P_n']$$

Goal: Find the best unit vector that capture max variance

Question: How to find the vectors?

We find the vectors using the concept "Eigen value decomposition"

Eigen Value decomposition
- $\rightarrow$ Eigen Values
- $\rightarrow$ Eigen Vectors

Eigen values $\rightarrow$ Tells how much variance a vector captures

Eigen values $\uparrow\uparrow$ $\rightarrow$ High variance capturing

Eigen values $\downarrow\downarrow$ $\rightarrow$ Low variance capturing

Steps to do to find Eigen values and Eigen vectors throug Eigen Value Decomposition

Step 1: Covariance Matrix between features.

Suppose we have 2 features (f_1, f_2) and we want to reduce $2D \rightarrow 1D$ then first we have to find $Cov(f_1, f_2)$.

Step 2: Eigen values and Eigen vectors will be found out using this covariance matrix A $Av = \lambda v$

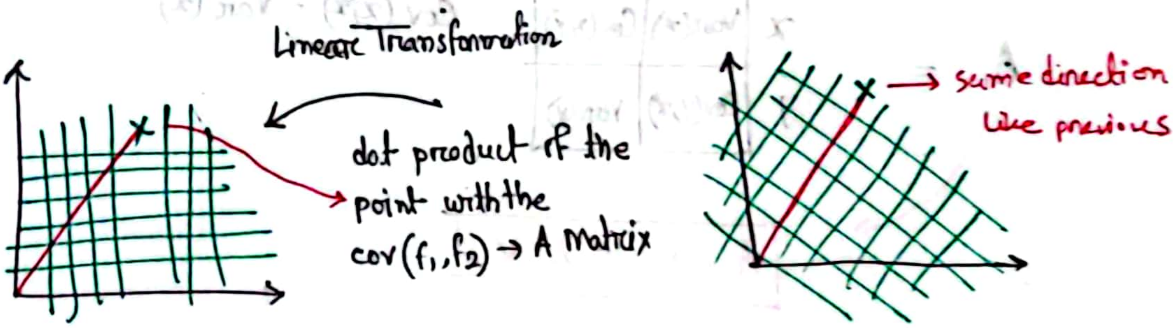
λ = Eigen value

A = Matrix

v = vector.

Step 3: Capturing the eigen vector which contains the maximum eigen value.

We already know how to find the covariance Matrix A.
Now, how will find the eigen vector v and eigen values λ



After applying Linear transformation the grid shape is changing so the magnitude and direction of the point will also change. But there are some points where the direction of the point will be same though the magnitude can be changed. Where the direction will be same, that will be our eigen vector.

Similarly we will find the eigen vectors for all the other points.

The magnitude of the eigen vectors will be the eigen values.

We will find from the

eigen vectors

↳ Max magnitude (Max Eigen value)

↳ Our PC1

Steps to calculate Eigen values and Eigen vectors: [2D-1D]

① Covariance of features

$x_1, y_1 \quad Z \rightarrow O/P$

$$Cov(x,y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

λ

$$A = \begin{array}{c|cc} & x & y \\ \hline x & Var(x) & Cov(x,y) \\ y & Cov(y,x) & Var(y) \end{array}$$

$Cov(x,x) = Var(x)$

$$\boxed{A \cdot v = \lambda \cdot v}$$

(Dot) Multiplying Matrix A with vector v, we can get λ [eigen value] and v [eigen vector]