Principle of Component Analysis (PCA)

* Curse of dimensionality
  ⇓
  features.

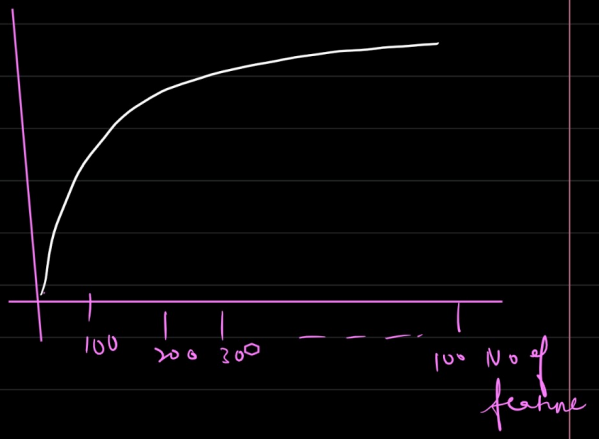
To predict price of house

f_1	f_2	...	f_{100}	y

20f	50f	100f	250f	500f	750f	1000f
M1	M2	M3	M4	M5	M6	M7

Acc Rsquare < Acc Rsquare ↑ < Acc Rsq ↑ < Acc Rsq ↑ ≈ Acc Rsq ≈ Acc Rsq ≈ Acc Rsq.

* With increase in no of features, after one point of time the Acc (rsquare) will not increase Rsq in that proportion.



⇓

Why?

→ Few of features will be multicollinear ($f_1 f_2 f_3 \approx f_4$)

→ few of features might be exactly same

f_1 — 1, 1, 1, 1, ... 1

f_2 — 1, 1, 1, ... 1

→ No variance / information in feature.

→ lots of duplicate entries

f_1
1
1.01
1.02
⋮

* With increase in no of feature performance of model degrades.

Execution time

Evaluation metrics

Analogy

* you went to buy a house.

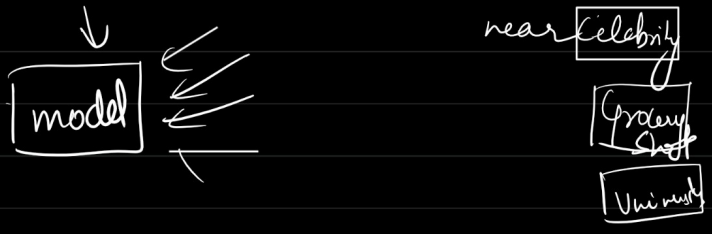
Broker

2 BHK → 60 Lakhs

3 BHK → 80 Lakhs

beach → ↑↑

Airport → ↑↑



* Curse of dimensionality → With increase in no of feature the performance of model degrades.

To remove COD :-

① Feature selection ② Feature Extraction

↓

PCA (Dimensionality reduction technique)

* Why we should remove COD ?

① To improve the performance of Model.
② Visualise the data → for insights
③ Prevents from overfitting.
④ Better interpretation.
⑤ To remove Curse of dimensionality

1000d → 3d

* Feature Selection

Area of house (X_1)	Near to airport (X_2)	Price of house y

① Correlation (Pearson)

$$Corr = \frac{Cov(x,y)}{\sigma_x \sigma_y}$$

$x \uparrow y \uparrow$
$x \downarrow y \downarrow$
$x \uparrow y \downarrow$
$x \downarrow y \uparrow$

② $$Cov(x,y) = \sum_{i=1}^{n} \frac{(x_i - \bar{x}) * (y - \bar{y})}{N-1}$$

$Corr(X_1, y) = 0.82$

$Cor(X_2, y) = 0.3$

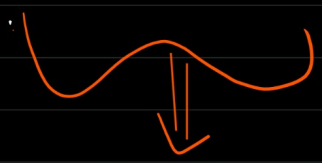
f_1 f_2 f_3 — — — — f_{1000} → y

0.82 -0.83

* Feature Extraction

(In feature Selection, you dropped the features. But let's if you want all the features and also curse of dimensionality, then comes the technique feature extraction)

Room Area	No. of rooms	Price of house



data transformation to extract New feature which represents both the the features.

House Area (using Domain expertise)	Price

but say → #of bathroom | distance from airport | distance from University.

Principal Component Analysis

→ Dimensionality reduction technique.

Principal Component Analysis (PCA — A dimensionality reduction technique)

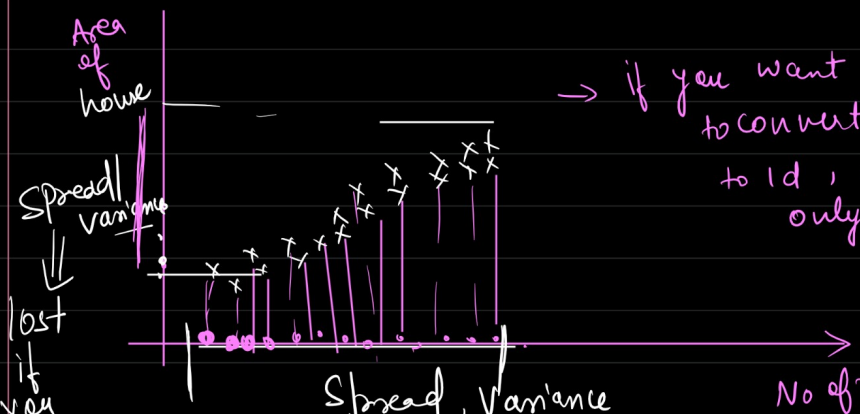
(Geometrical Intuition)

Area of land	No of Rooms	Price of house

* Since PCA is a dimensionality reduction technique → we will try to reduce the dimension

2d ⟶ 1d. → Using PCA.

* ML is all about learning patterns of data.



→ if you want to convert to 1d, take only No of rooms

⇓

x-axis

⇓

We will loose area of house (Actually this is happening in feature selection)

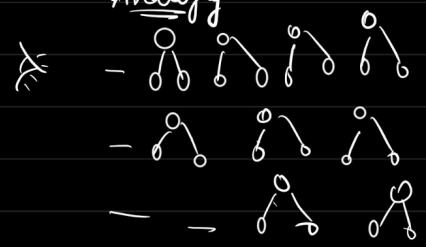
⇓

You want both the features.

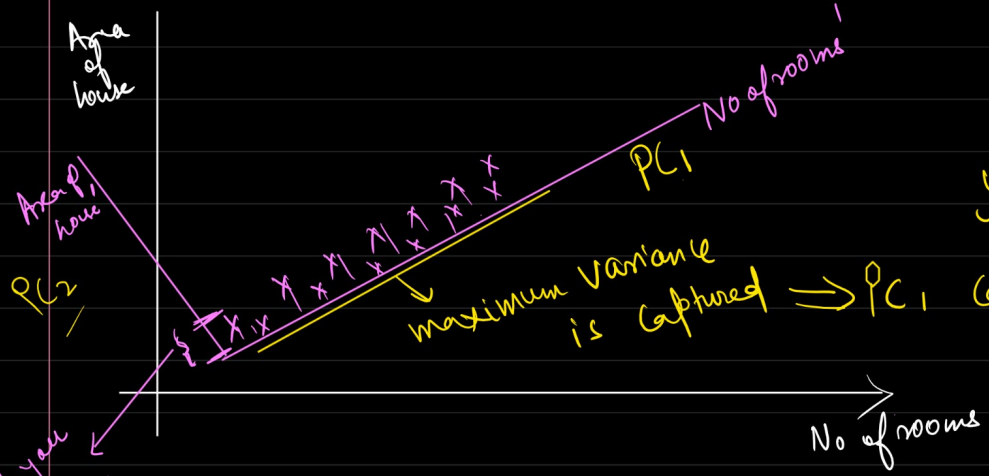
Lost if you take only one feature

→ if Spread ↑, Variance ↑

Analogy



Photographer ↑



maximum variance is captured ⇒ PC_1 can be used for model training.

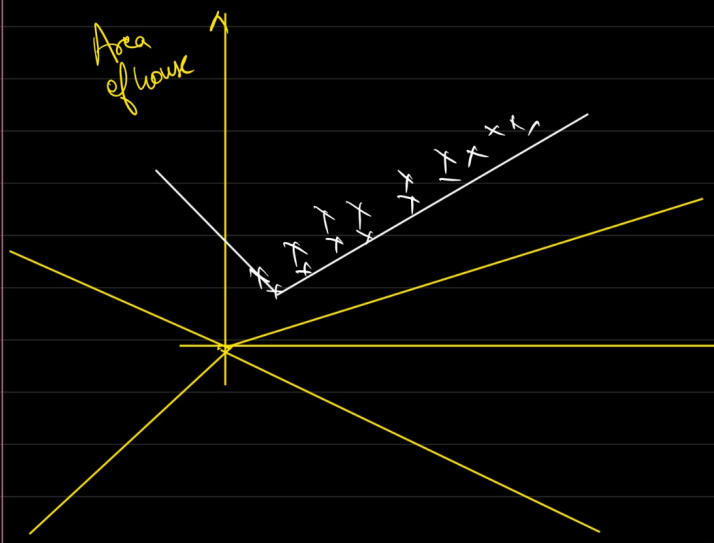
if you use only new feature no of rooms! then very less amount of variance will be lost..

2d — PC_1 PC_2

⇓

PC_2 is used because it captures maximum Spread of data

2d — 1d (PC_2)



$PC_1 \rightarrow$ maximum Variance (Spread)

* No of Principal components will be equal to no of features.

* $PC_1 > PC_2 > PC_3 > PC_4 \dots > PC_n$
Var Var Var Var Var

f_1	f_2	f_3	---	f_{1000}	y	PC_1	PC_2	PC_3		PC_{1000}

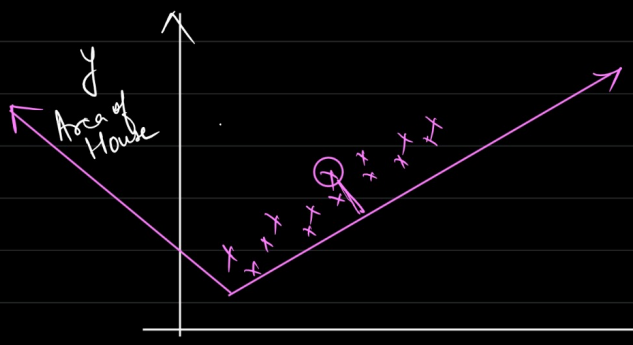
$f_1 \; f_2 \; f_3$

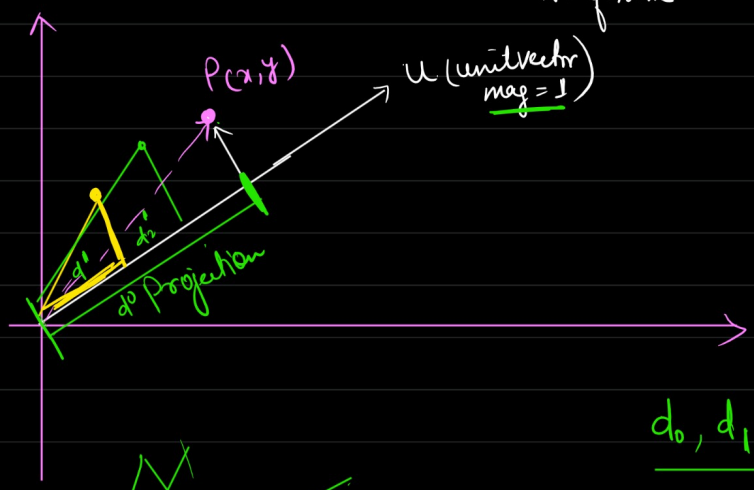
$PC_1 \; PC_2 \; PC_3$

$Var(PC_1) > Var(PC_2) > Var(PC_3)$

axis transformation $\Rightarrow$ Eigen decomposition of Covariance Matrix

* Mathematical Explanation of PCA





$$\text{Proj of } P_1 \text{ on } u = \frac{P \cdot u}{\|u\|}$$

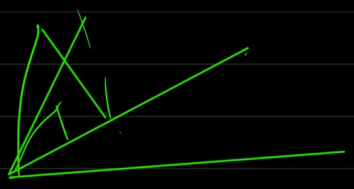
$$\text{Proj of } P_1 \text{ on } u = P \cdot u$$

⇓

Scalar value

⇓

Projection.



$d_0, d_1, d_2, d_3 \dots d_n$

⇓

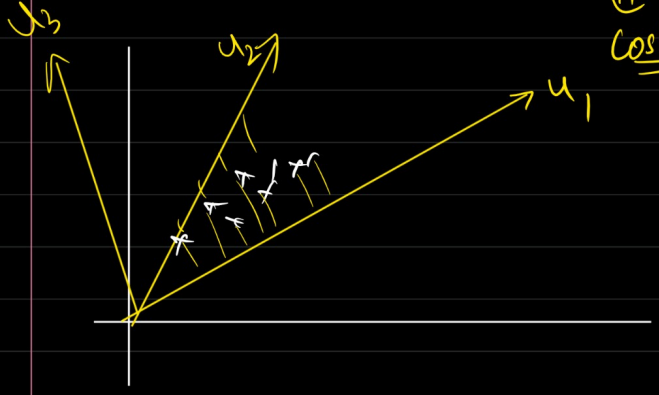
You want that Unit vector where variance / spread is maximum.

$$\text{Var} = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}$$

⇓

Cost fn

⇒ Aim is to find that unit vector which captures the maximum variance after projection.



spread

$$\text{var} = \frac{\sum (x_i - \bar{x})^2}{n}$$

⇓

$$\text{Co-variance} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y - \bar{y})}{n}$$

* Covariance matrix.

	x_1	x_2
x_1	$Cov(x_1, x_1)$	$Cov(x_1, x_2)$
x_2	$Cov(x_2, x_1)$	$Cov(x_2, x_2)$

$Cov(x_1, x_1) = Var(x_1)$

$$\begin{bmatrix} Var(x_1) & Cov(x_1, x_2) \\ Cov(x_2, x_1) & Var(x_2) \end{bmatrix} \rightarrow \text{covariance matrix}$$

	x_1	x_2	x_3
$\rightarrow x_1$	$Var(x_1)$	$Cov(x_1, x_2)$	$Cov(x_1, x_3)$
x_2	$Cov(x_1, x_2)$	$Var(x_2)$	$Cov(x_2, x_3)$
x_3	$Cov(x_3, x_1)$	$Cov(x_3, x_2)$	$Var\, x_3$

* if you decompose a covariance matrix of features, then you will get eigen value and Eigen vector and Eigen vector with highest magnitude eigen value captures the maximum variance/spread.

(2nd highest magnitude — 2nd highest variance
3rd " " — 3rd " "
and so on)

Linear transformation of a matrix

$$A \cdot \vec{V} = \lambda \cdot \vec{V}$$

$\vec{V}$ ↓ Eigenvector, λ ↓ Eigen value

A — a matrix

{ When we use covariance matrix as A → Eigen decomposition of covariance matrix) !! Eigenvector & Eigenvalue.

Linear transformation

↳ A matrix transformation that brings changes in the coordinate system

$$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$ → Matrix → Linear transformation

* → Many vectors were changing both magnitude and direction
* few vectors changed only magnitude and not the direction.

⇓

→ these vectors are called Eigen vectors.

Eigen vectors

(1,0) → (3,0)

3 time changes → Eigen value

→ Eigen values are change in magnitude for Eigen vectors.

* The vectors which only changes magnitude and not direction will be equals to dimension of matrix / no of features.

2×2 ⟶ 2 vectors
3 ⟶ 3 vectors

No of features = No of Principal component

* Higest eigen value → PC1

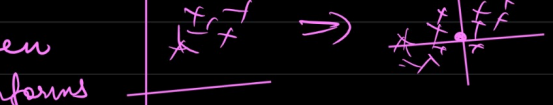
$$A \cdot \vec{V} = \lambda \cdot \vec{V}$$ (direction same = V, stretch/shrink = λ)

* Steps to Calculate Eigen Value & Eigen Vectors (to find PC's)

① Standardise the data (make the data mean centred.)

⇓

because it has been Observed PCA performs



better on mean-centred data.

② Covariance matrix

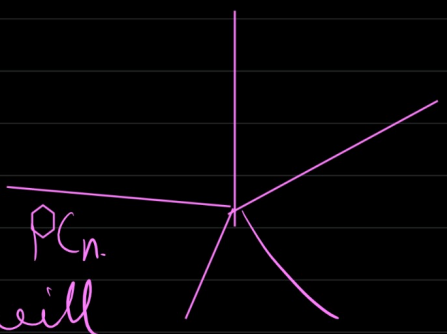
③ Eigen decomposition of Cov matrix.

$$Av = \lambda v$$

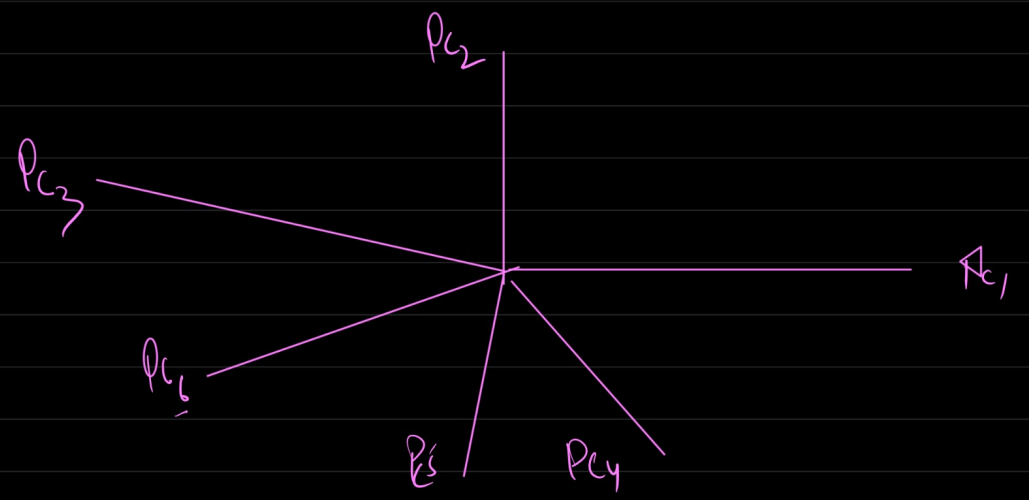
$\lambda \rightarrow$ eigen value

$v \rightarrow$ Eigen vector.

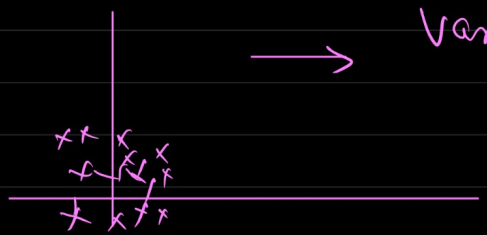
1000 f $\longrightarrow$ 1000 PC's



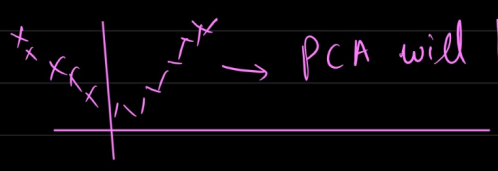
* $PC_1 > PC_2 > PC_3 \ldots PC_n$.
* Max of variance/spread will be captured by first few Principal components
* PC's will be $\perp^r$ to each other.



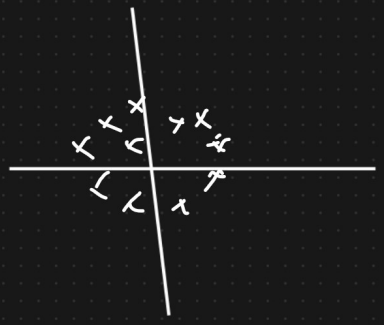
* PCA fails?



$\longrightarrow$ Variance spread across all axis is same.



$\longrightarrow$ PCA will loose this pattern.

①  ⇒ Standardize the dataset ⇒ 

② Cov (x, y)

③ Find out the Eigen value & Eigen vector

$$Av = \lambda \cdot v$$

2 3 . . . n

Variance ↑↑ ⇒ PC1, PC2, PC3 . . .

The "curse of dimensionality" refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings. As the number of dimensions (features) increases, several issues and challenges emerge, making data analysis, modeling, and computation more complex and often less effective. Here are some key aspects of the curse of dimensionality:

1. Sparsity of Data
In high-dimensional spaces, data points tend to be sparse. For example, if you have a dataset with a fixed number of samples, increasing the number of dimensions means that the data points are spread out more thinly across the space. This sparsity makes it difficult to find meaningful patterns or clusters in the data.

2. Distance Metrics
Distance metrics (e.g., Euclidean distance) become less informative as the number of dimensions increases. In high-dimensional spaces, the distance between any two points tends to converge, making it hard to distinguish between nearby and faraway points. This can affect algorithms that rely on distance measurements, such as k-nearest neighbors (k-NN) and clustering algorithms.

3. Volume and Sampling
The volume of a high-dimensional space grows exponentially with the number of dimensions. This means that to cover or sample a high-dimensional space adequately, an exponentially increasing number of samples is needed. For example, in a 2-dimensional space, a grid with 10 intervals per dimension requires 10^2 = 100 points, while in a 10-dimensional space, it requires 10^10 = 10,000,000,000 points.

4. Overfitting
With high-dimensional data, models can easily become too complex and overfit the training data. Overfitting occurs when a model learns the noise in the training data rather than the underlying pattern. This is especially problematic in machine learning, where high-dimensional feature spaces can lead to models that perform well on training data but poorly on new, unseen data.

5. Computational Complexity
The computational cost of many algorithms increases significantly with the number of dimensions. For example, matrix operations, search algorithms, and optimization processes often become more computationally intensive as dimensionality increases.

Strategies to Mitigate the Curse of Dimensionality
1. Feature Selection
Select a subset of the most relevant features based on statistical tests, domain knowledge, or feature importance scores from models like random forests.

2. Feature Extraction
Transform the original high-dimensional features into a lower-dimensional space using techniques like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or t-Distributed Stochastic Neighbor Embedding (t-SNE).

3. Regularization
Apply regularization techniques in machine learning models to prevent overfitting. Regularization methods such as Lasso (L1 regularization) and Ridge (L2 regularization) can help control the complexity of the model.

4. Dimensionality Reduction
Use dimensionality reduction techniques to project high-dimensional data into a lower-dimensional space while preserving as much

Principal Component Analysis (PCA) is a statistical technique used for dimensionality reduction. It transforms high-dimensional data into a lower-dimensional form while preserving as much variance (information) as possible. PCA achieves this by identifying the directions (principal components) along which the data varies the most. Here's an in-depth look at how PCA works and its applications:

Key Concepts of PCA

1. Variance and Covariance

Variance measures how much the data points deviate from the mean.

Covariance measures how much two dimensions vary together. A positive covariance indicates that the dimensions increase together, while a negative covariance indicates that as one increases, the other decreases.

2. Principal Components

Principal Components are the directions in the feature space along which the data varies the most. These are orthogonal (perpendicular) to each other.

The first principal component captures the maximum variance in the data. Each subsequent component captures the remaining variance under the constraint that it is orthogonal to the previous components.

3. Eigenvalues and Eigenvectors

PCA involves computing the eigenvalues and eigenvectors of the covariance matrix of the data. The eigenvectors represent the directions of the principal components, and the eigenvalues indicate the amount of variance captured by each principal component.

Steps to Perform PCA

Standardize the Data: If the data has different scales, standardize it so that each feature has a mean of 0 and a standard deviation of 1.

Compute the Covariance Matrix: Calculate the covariance matrix to understand how the variables of the input data are related to each other.

Calculate Eigenvalues and Eigenvectors: Determine the eigenvalues and eigenvectors of the covariance matrix. The eigenvectors correspond to the principal components, and the eigenvalues indicate their magnitude of variance.

Sort Eigenvalues and Eigenvectors: Sort the eigenvalues in descending order and arrange the corresponding eigenvectors accordingly. The top
k eigenvectors form the new feature space.

Transform the Data: Project the original data onto the new feature space (formed by the top
k eigenvectors) to obtain the reduced-dimensionality data.