Techniques in Dimensionality Reduction

COD

Curse of Dimensionality describes the explosive nature of increasing data dimensions and its resulting exponential increase in computational efforts required for its processing and/or analysis.

Techniques in Dimensionality Reduction

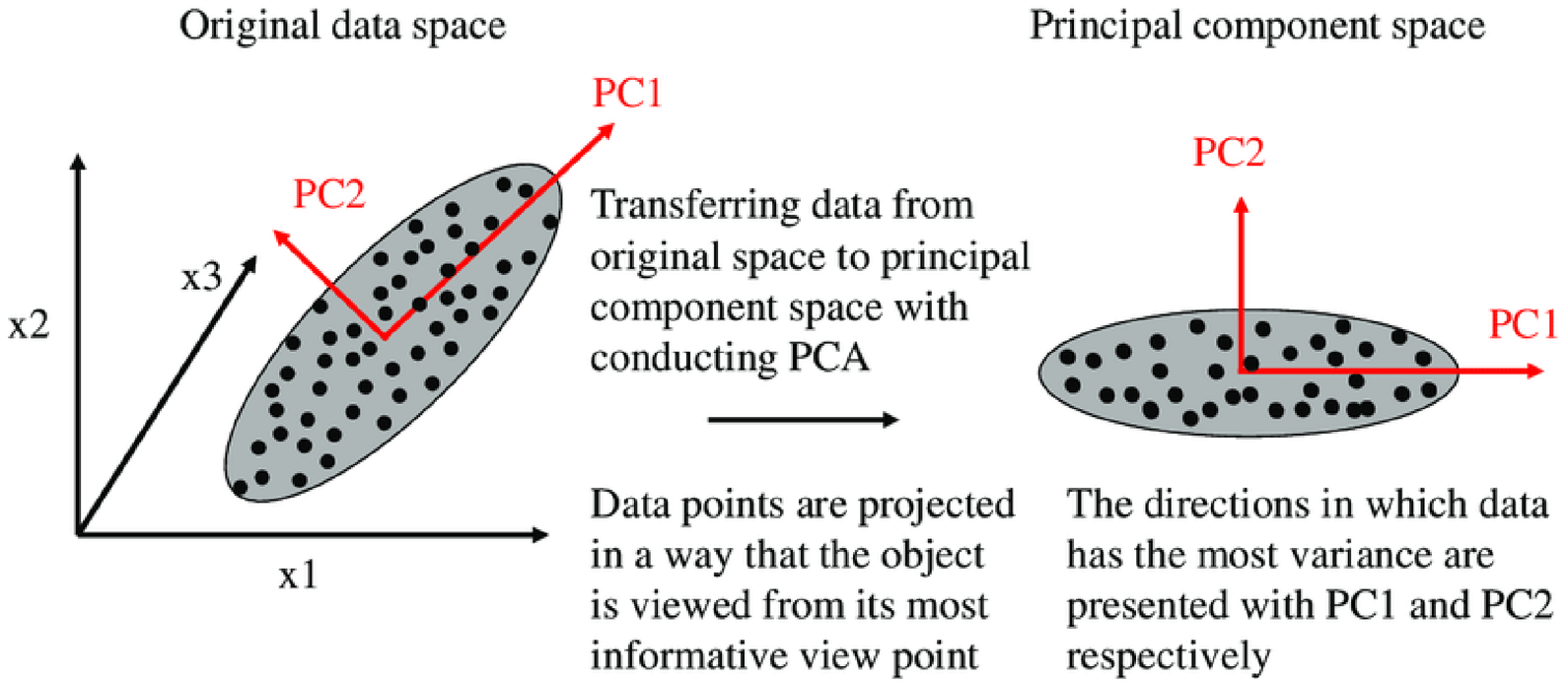
1. Feature Selection Methods.

2. Manifold Learning (e.g. t-sne, MDS, etc..)

3. Matrix Factorization (e.g PCA, Kernal PCA, etc..)

PCA

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique and it comes under an unsupervised machine learning algorithm because we don't need to provide a label for dimension reduction. We can use PCA for dimensionality reduction or we can use PCA for analysis of higher dimension data in a lower dimension. PCA algorithm's, task is to find new axes or basis vectors that preserve a higher variance for data in lower dimensions. In PCA, new axes are known as PC's.



Why & when PCA:-

1- Better Perspective and less Complexity: When we need a more realistic perspective and we have many features on a given data set and specifically when we have this intuitive knowledge that we don’t need this much number of features.
Similarly, in many other practices modelling is easier in 2D than 3D , right?
2 – Better visualization: When we cannot get a good visualization due to high number of dimensions we use PCA to reduce it into a shadow of 2D or 3D features (or even more but convenient enough for better parallel coordinates or Andrew Curve, e.g. when you transfer 100 features into 10 features you cannot still depict it as 2D or 3D but you can get a much better Andrew Curve)
3- Reduce size: When we have too much data and we are going to use process-intensive algorithms (like many supervised algorithms) on the data so we need to get rid of redundancy .
Sometimes change of perspective matters more than Dimension reduction and we want to exploit dimensionality :
4- Different perspective: Maybe you don’t have any of these motivations but you merely need to improve your knowledge on your data. PCA can give you the best linearly independent and different combinations of features so you can use to describe your data differently.

Steps in Principal Component Analysis and Mathematical Proof

Step1: Standardization of the continuous features in the dataset.

Step2: Computing the covariance matrix.

Step3: Computing eigenvalues and eigenvectors for a covariance matrix.

Step 4: Sort the eigenvalues of the covariance matrix in descending order and select the n number of eigenvalues from the top. where n is the number of axes you need.

Step 5: Select the eigenvectors corresponding to the above-selected eigenvalues.

Step 6: To get the data set for the new feature space, just use the formula of Projection of data points onto new axes. U_transpose * Xi_vectors . Where U is the basis vector(axes) and Xi(data points). .

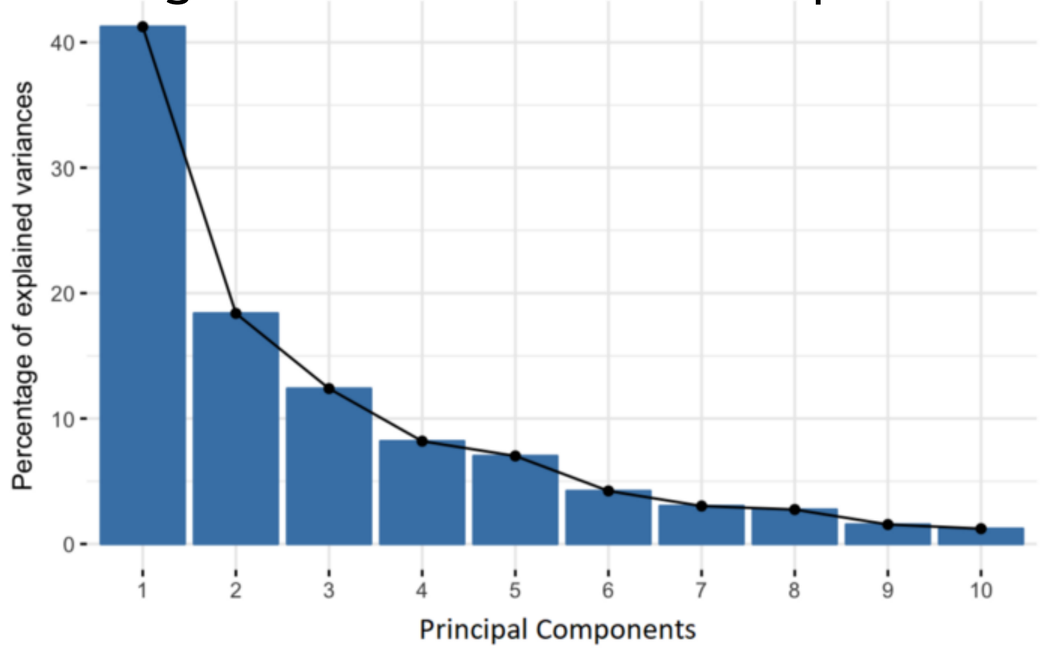
Normalize your Data before performing PCA :-

The reason you should think about Normalizing/ standardizing your data before performing PCA is that PCA is a variance maximizing exercise . PCA calculates a new projection of your data set. It projects your original data onto directions which maximize the variance. So the new axis are based on the standard deviation of your variables. So a variable with a high standard deviation will have a higher weight for the calculation of axis than a variable with a low standard deviation. If you normalize your data, all variables have the same standard deviation, thus all variables have the same weight and your PCA calculates relevant axis.

More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

What are Principal Components

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. So, the idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, until having something like shown in the scree plot below.



Organizing information in principal components this way, will allow you to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.

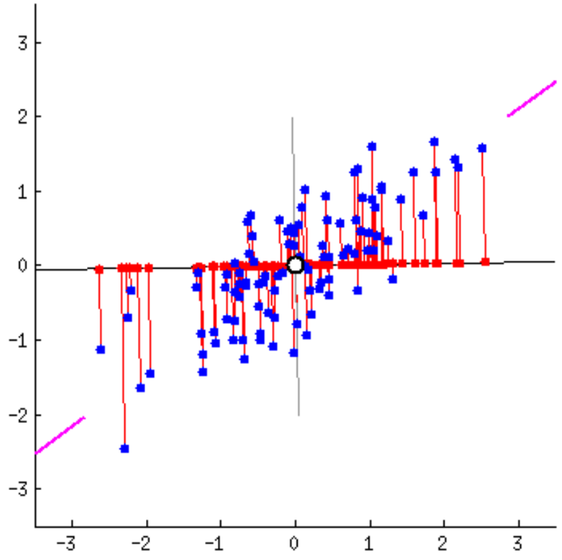
An important thing to realize here is that, the principal components are less interpretable and don't have any real meaning since they are constructed as linear combinations of the initial variables

What are Principal Components

Geometrically speaking, principal components represent the directions of the data that explain a **maximal amount of variance**, that is to say, the lines that capture most information of the data. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more the information it has. To put all this simply, just think of principal components as new axes that provide the best angle to see and evaluate the data, so that the differences between the observations are better visible.

How PCA Constructs the Principal Components

As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set. For example, let's assume that the scatter plot of our data set is as shown below, can we guess the first principal component ? Yes, it's approximately the line that matches the purple marks because it goes through the origin and it's the line in which the projection of the points (red dots) is the most spread out. Or mathematically speaking, it's the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).



The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance.

This continues until a total of p principal components have been calculated, equal to the original number of variables.

STEP 3: COMPUTE THE EIGENVECTORS AND EIGENVALUES OF THE COVARIANCE MATRIX TO IDENTIFY THE PRINCIPAL COMPONENTS

it is eigenvectors and eigenvalues who are behind all the magic explained above, because the eigenvectors of the Covariance matrix are actually the directions of the axes where there is the most variance(most information) and that we call Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which give the amount of variance carried in each Principal Component.

Why do we need a covariance matrix

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

Why do we need a covariance matrix and eigenvalues and eigenvectors of it?

We know PCA will try to find new PCs (axes), but how is it gonna do it. it tries a bunch of vectors(xi) and tries to project all of the data points onto that vectors(xi) and it will calculate the variance of the vectors(xi) and select the vectors(xi) which preserve a higher variance of the datapoint.

Think of this like a linear regression without gradient descent, where our model will try a bunch of lines and calculate the error between the actual and prediction values. For the best fit line, we considered the cost function in linear regression. But the PCA will select the axes based on the Eigenvalues. And the axes are nothing but an Eigenvector corresponding to that eigenvalues.

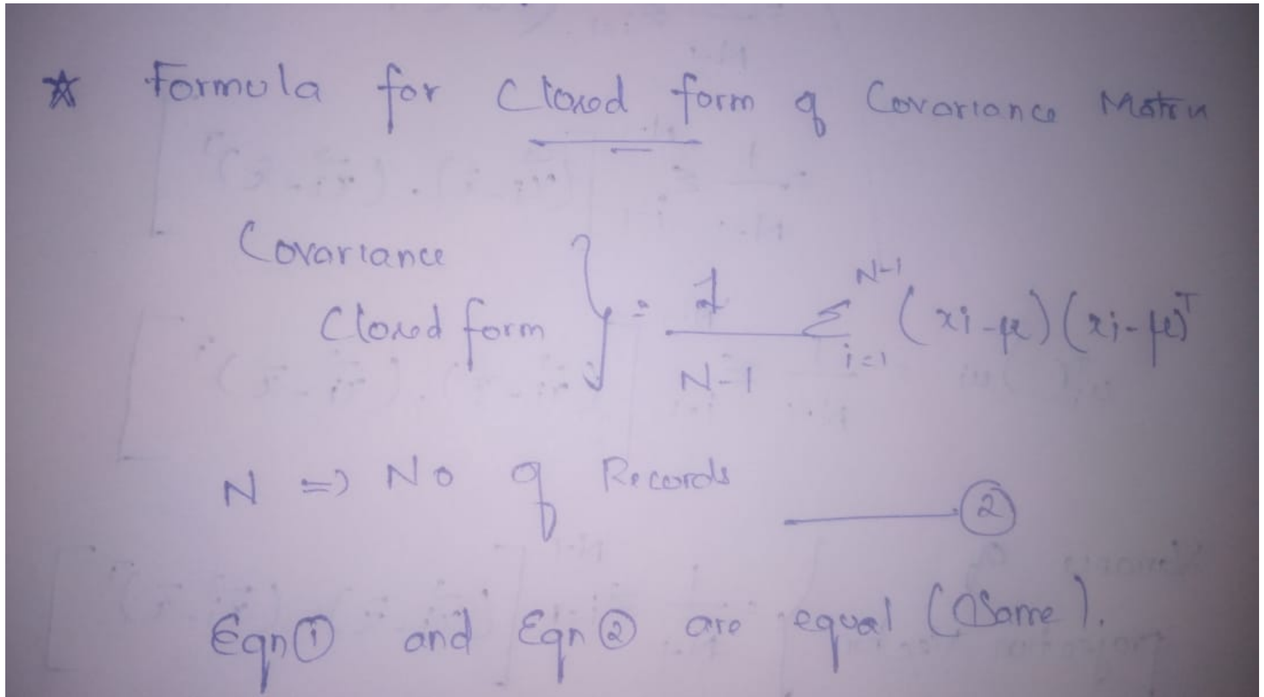
New Axes = Eigenvectors

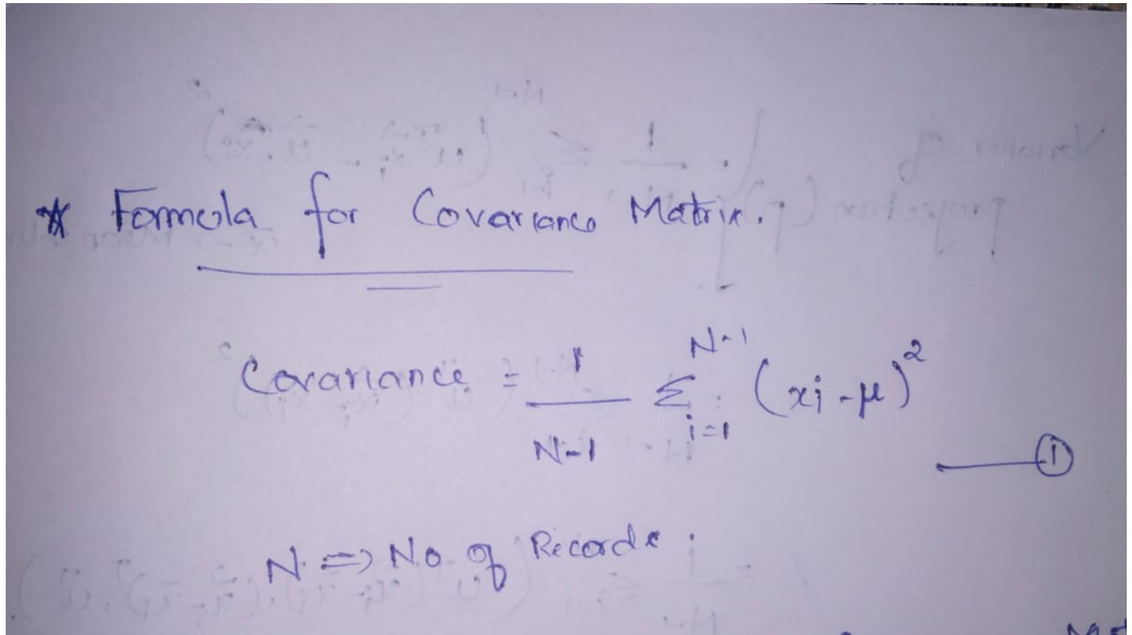
But why eigenvectors and eigenvalues ?

Why do we need a covariance matrix and eigenvalues and eigenvectors of it?

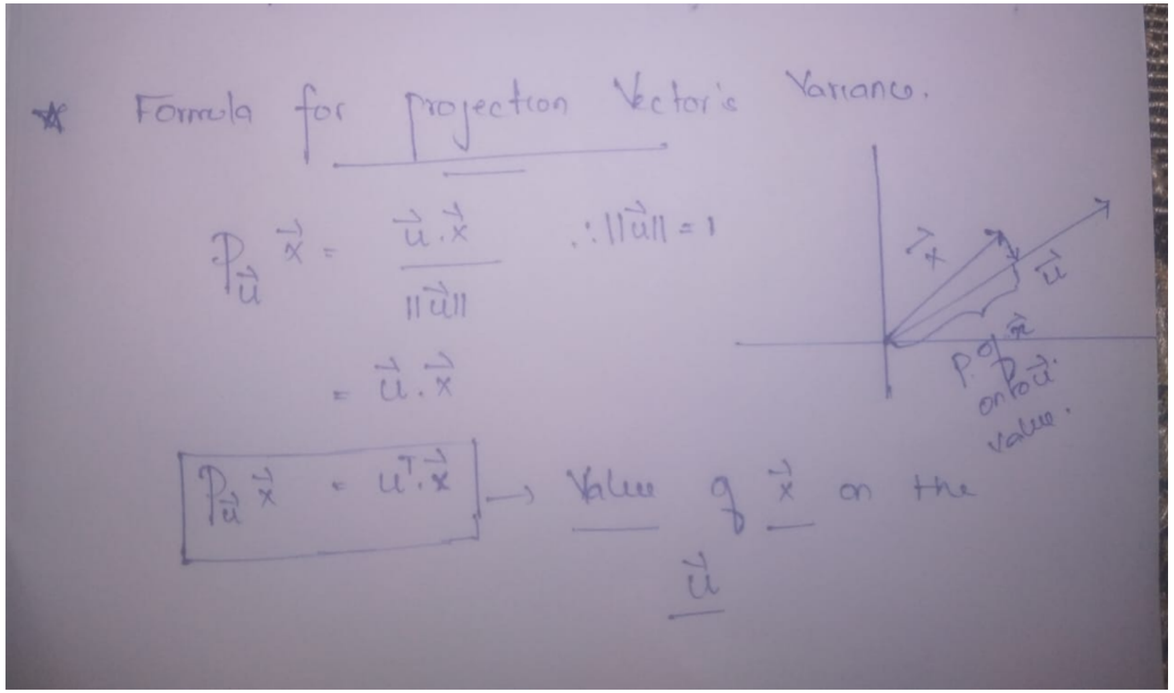
Before the proof, we need to understand some math formulas involved in Principal Component Analysis.

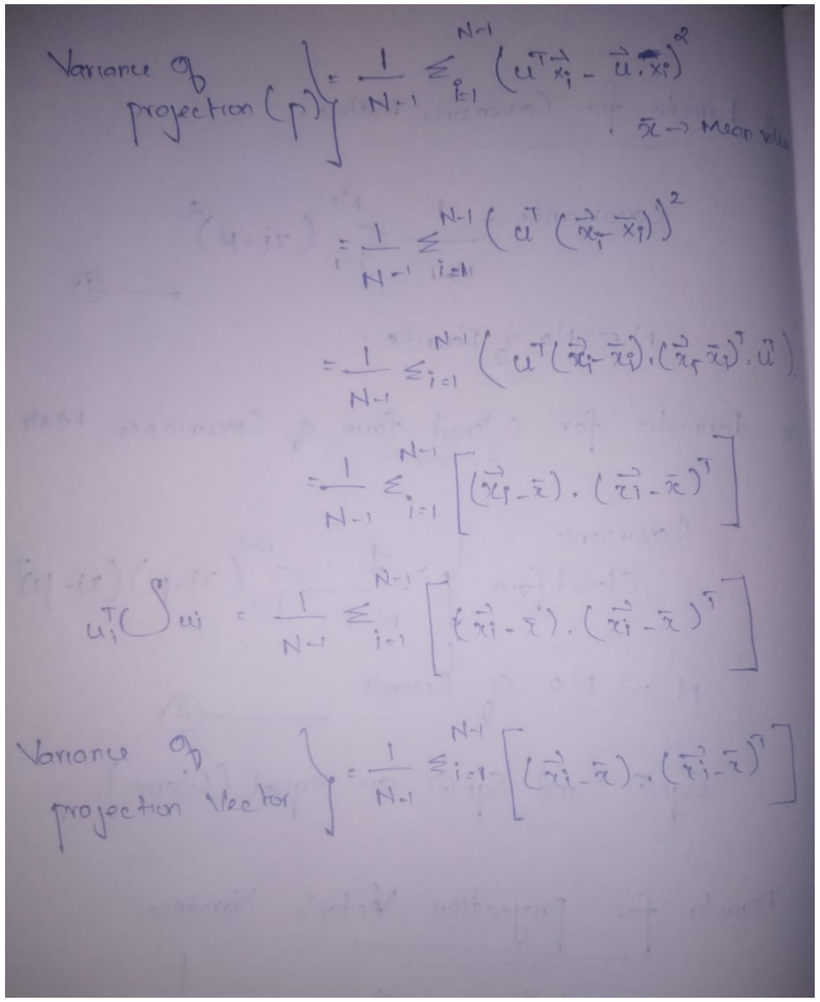
1. We need a formula for the Variance of the projection vector.
2. Need to know about a Closed-Form of the covariance.





Why do we need a covariance matrix and eigenvalues and eigenvectors of it?





From the equation of the Variance of projection vector, we can see some magic that both closed covariance matrix and variance of projection vector are the same. So we already know that the closed covariance matrix and covariance matrix are the same. So, now variance of projection vector and covariance matrix are the same

LAST STEP: RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES

In the previous steps, apart from standardization, you do not make any changes on the data, you just select the principal components and form the feature vector, but the input data set remains always in terms of the original axes (i.e, in terms of the initial variables).

In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Pros and Cons :

Pros :

1) Reduces Overfitting : PCA helps the model not to overfit by reducing unwanted features from the dataset.

2) Helps in Visualization : It is difficult to visualize data which are high in dimension (4D or more), PCA helps to visualize it by reducing the dimensionality.

3) Improves Model Performance : Having too many features will lead the model not to give best and accurate results, but thanks to PCA , It speeds up the machine learning algorithm by getting rid of correlated variables.

Cons :

1) Standardization is a must on the dataset before implementing PCA, else PCA will not be able to optimal principal Components.

2) Chances of information loss if we don't choose the principal components with care.

3) Input variables tend to become less interpretable.

DATE

used in → Feature Selection

Curse of Dimensionality (COD)

→ Feature Extraction

ज्यादा feature तो ज्यादा dimension → that is COD ज्यादा बढ़ जाते हैं मतलब नहीं।

as No of feature ↑ Beyond limit तक accuracy ~~accuracy~~ बढ़ेगा
Complex हो जायेगा acc. May decrease
तो और feature add करने पर कोई फायदा नहीं that is COD

eg in Image

Sparsity हुई हुई as in (higher dim)

eg wallet ढूंढना on road, playground, building में ढूंढना तो ~~...~~ ~~easiest~~ way to find wallet is 1-D (on road)

How to

As dim ↑,
- Performance decrease
- Computation ↑

How to solve ... Data Reduction ...

on road, playground, building में तो easiest way to find wallet is 1-D (on road)

As dim ↑
- Performance decrease
- Computation ↑

How to solve → Dimensionality Reduction ← How to (COD↓)?

Feature Selection
- forward selection
- Backward elimination
- ...

Feature Extraction
- PCA
- LDA
- tsne

Feature Extraction

PCA Principle Component Analysis

↘ unsupervised Mlearning prob.

↘ Complex technique (old)

PCA is a technique which can transform higher dim data to lower dim data while keeping essence of data

(मतलब data का behavior को पकड़े रखते lower dim में reduce करना)

(eg. photos में 1-D और 2-D के)

Why we use PCA in ML & (Benifits)

1 → Higher और lower dim → so size और time 2 तरह so algo run fast

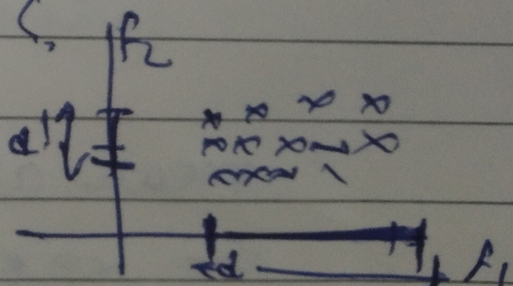
(faster execution of algo.)

2 → Visualization Higher dim और plot नहीं और समझ में नहीं reduce और 2 या 3 तो visualize कर सकते ।

Geometric Intuition: -

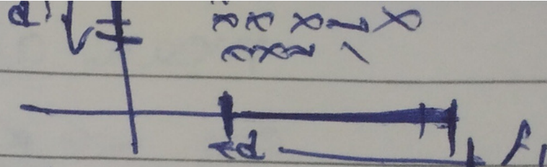
first know feature selection ?

f_1 और f_2 में f_1 (Projection) so we select f_1 (और नहीं depend)



PCA

↳ यहाँ (variance) spread done हो गया



PCA

↳ यदि spread (variance) same है दोनों पर तो decision नहीं कर पायेंगे कि किसको select/Remove करें तो feature selection fail so solved by feature extraction

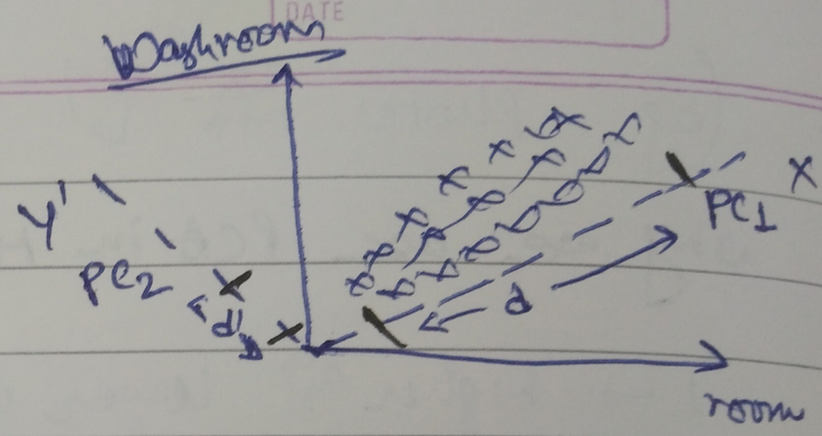
Hao feature extraction solve this problem

PCA eg.

Room	washroom	Price

Price depend करे room / washroom पर let दोनों f_1 & f_2 equally ≈ spread करे तो PCA क्या करेगा कि कोई एक को remove करेंगे new dim बनाता है जो size

Geom. Int. (of PCA)



So PCA rotate the axis

Now $d > d'$

So transform the data acc to. PC1

Imp Why Variance (spread) is important in (stat.)

Variance बताता ~~variation~~

(Spread ∝ Variance) Variance is Not exactly spread

mean absolute deviation (MAD) भी same (spread) को represent करता है But यहाँ variance को क्यों लेते हैं?

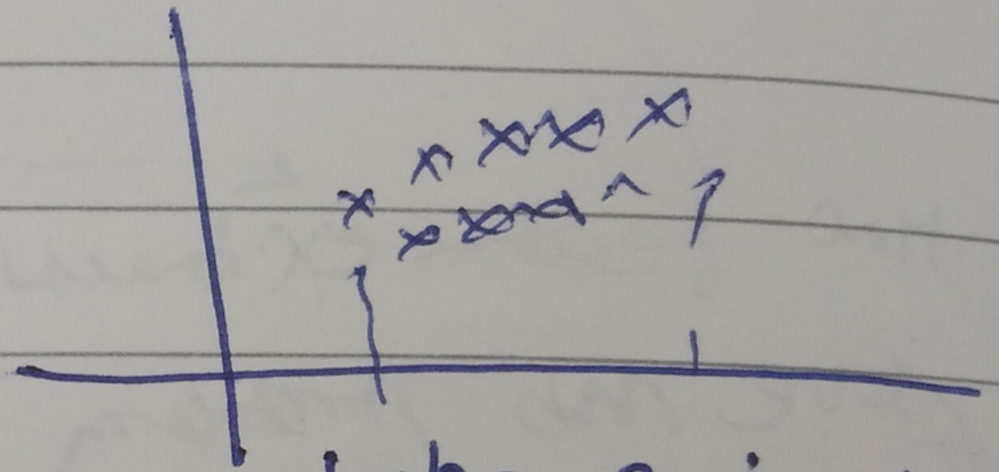
क्यों use करे ?

Because MAD is NOT (diff)

बाद के optimization tech ने diff करना पड़ सकता है|

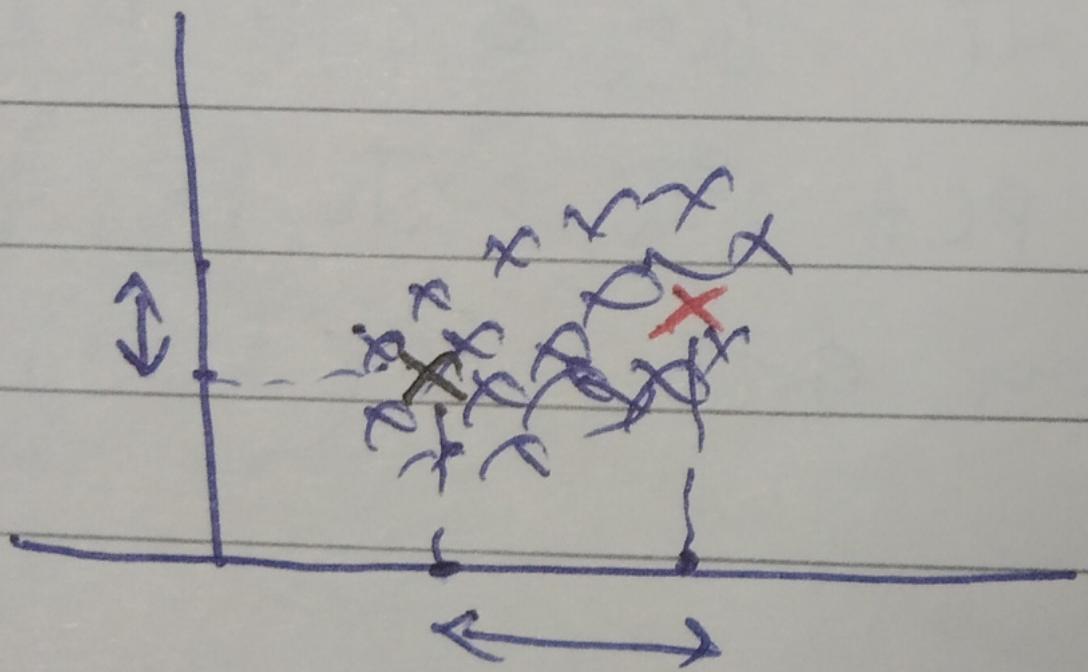
$$MAD = \frac{\sum |x_i - \bar{x}|}{n}$$

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



take projection then find variance.

Variance Imp for PCA ?



let PCA की help से Red/Black Point को check करना है|

Proj on x-axis में acc variance ज्यादा है
as compare to y
तो यही distance based algo uses करेगा तो y-axis
में acc distinguish नहीं कर पायेगा क्योंकि dist/
variance कम है so variance ज्यादा होना चाहिये

eg. photographer taking photo's of Mayor (Mayor)
तो फिर angle से ज्यादा spread (अन्दर से
बाहर spread) वाले angle से लेगा
say x-axis (in above example)

(photo) clear

(PCA dim. reduce करने के लिए)

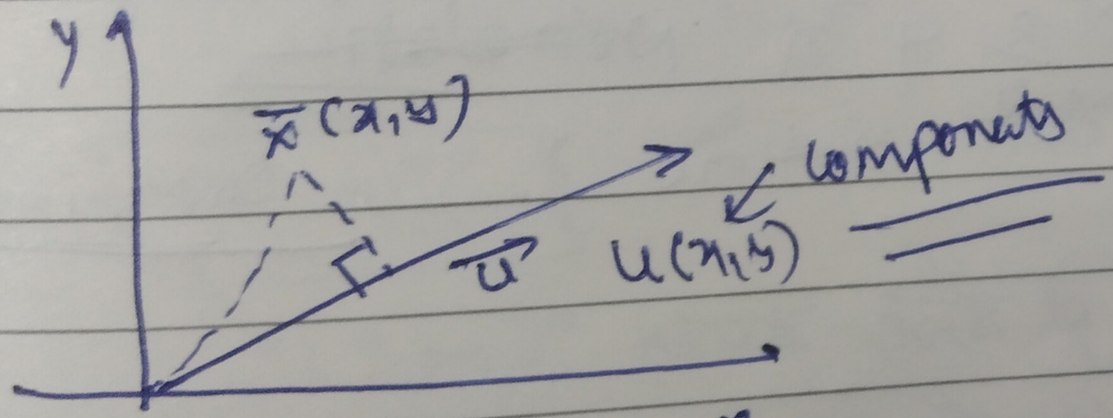
PCA mathematically कैसे Prob solve करते है ?

Problem Formulation

In PCA, we have to find a direction (unit vector) such that the variance of all points (after projection) is Maximum

project (unit vector) $\frac{\vec{u} \cdot \vec{x}}{|u|}$ x on $\vec{u}$ vector

$\vec{u} \cdot \vec{x} = u^T x$



We know Variance $\sigma^2 = \frac{1}{n} \sum (x_i - \bar{x})^2$

$\Rightarrow u^T \cdot \bar{x}$ ⟷ mean of the point we project

objective f^u

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

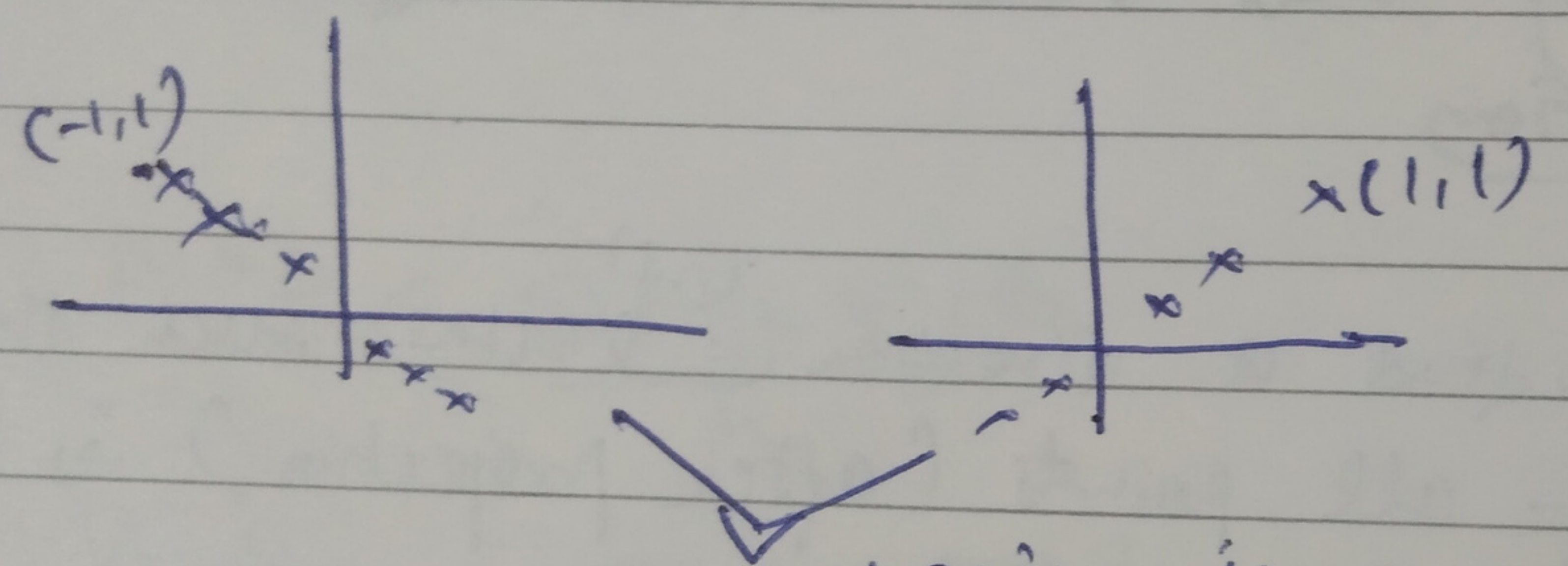
find unit vector such that variance is maximum

Some concepts:

Covariance & Covariance Matrix

variance only एक axis का spread बताता है
But Multiple के compare नहीं कर पाता
(relationship)

eg



variance is same (~~Say on~~
मतलब Dono dataset same है but
कैसा नहीं है

So अगर X & Y के ~~No relation~~
relation कैसे find करें ? → Covariance

Correlation & covariance is nearly same but correlation is (-1 to 1) but no restriction on covariance

Covariance Matrix

let two feature

x_1	cov(x_1, x_1)	cov(x_2, x_1)
x_2	cov(x_1, x_2)	cov(x_2, x_2)
	x_1	x_2

cov(x_1, x_1) = var(x_1)

cov(a, b) = cov(b, a)

$$\begin{bmatrix} var\ x_1 & cov(x_1, x_2) \\ cov(x_1, x_2) & var\ x_2 \end{bmatrix}$$

symm.

Cov. matrix to give full details about data

diag ← give की सारे axis पर data कैसे फैला हुआ है [spread]

other ← cov ← term → बताते हैं कि data का orientation क्या है

etg

Linear Transformation:—

→ vector transform हो गया

Matrices

→ are basically L.T. जो Co-ordination system में change करते है

Eigenvectors:— वो vectors जिनके ऊपर LT लगाते है तो उनका dirⁿ change नहीं होता। But scale change हो सकता है
$\vec{v}$
may ↑, ↓, या -ve dirⁿ हो सकता है

eigenvalue:

λ   let पहले eigenvector का mag 1 था बाद में 3 हो गया तो scaled by 3 times ↑

so 3 is eigenvalue

$$A\vec{v} = \lambda\vec{v}$$

→ LT on $\vec{v}$ मतलब उसके λ का multiple (Mag ↓, ↑ But direction में change नहीं)

After solving last optimization (t^{th} objective b^{th}) we find that,

- The largest eigenvector of the covariance matrix always points into the direction of the largest variance of the data & magnitude of this vector equals the corresponding eigenvalue.

So ~~इसलिये~~ Covariance Matrix का eigenvectors ~~eigenvalue के बराबर होगी~~

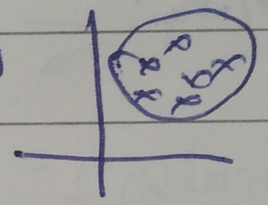
larger $\vec{e_v}$ ⇒ largest spread

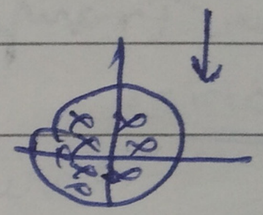
Step by step sol^{n} :—

f_1	f_2	f_3

1) mean center of all colum.

eg





2) Go find the cov. matrix

3) Find the eigen value / vector

4) $v_1, v_2 \ldots$ $\lambda_1, \lambda_2 \ldots$

$\max(\lambda_1, \lambda_2 \ldots)$ is PC1

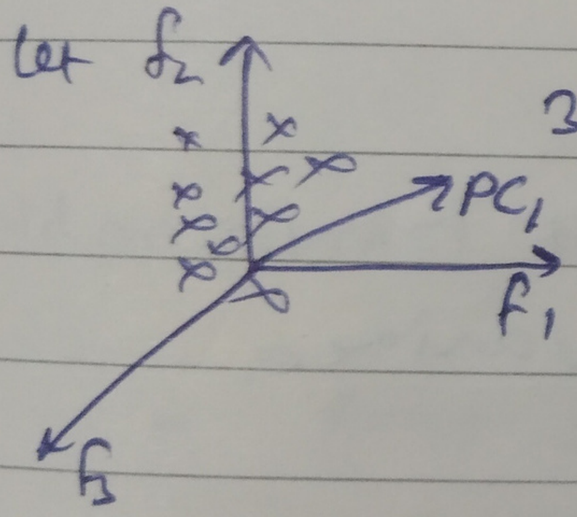
eg 3-D $\lambda_1, \lambda_2, \lambda_3$ let $\lambda_1 > \lambda_2 > \lambda_3$

PC1 PC2 PC3

1-D 2-D

How to transform Points ?

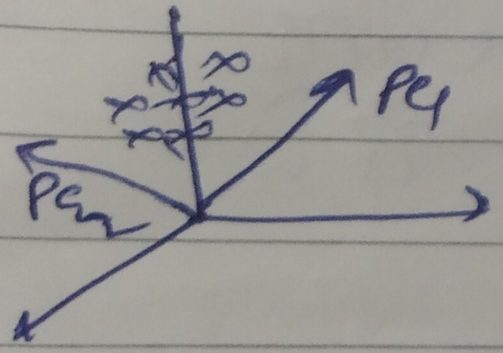
(How we get PC's ... → How to transform ?



3D → 1D

By dot product (projection) concept on PC_1

PC_1	output



2D → on PC_1, PC_2

PC_1	PC_2	output

Finding optimum number of Principle Components :-

$d_1 \; -- \; d_2$

$d_i' = \left(\frac{d_i}{d_1 + d_2 - - d_n} \right) \times 100$ ← Percentage value

→ ये बताता है कि — % variance explain कर रहा है उस data को

we need 90% variance approx.

add -- d_i' -- d_{15}'

let d_{15}' comp give ≈ 90% ✓

so 15 Component लेंगे ।

pca.explained_variance_ratio_ → eigen value

pca.

np.cumsum(↘))

(or) plt.plot(↗)

so we get कौन-कितना explain कर रहा

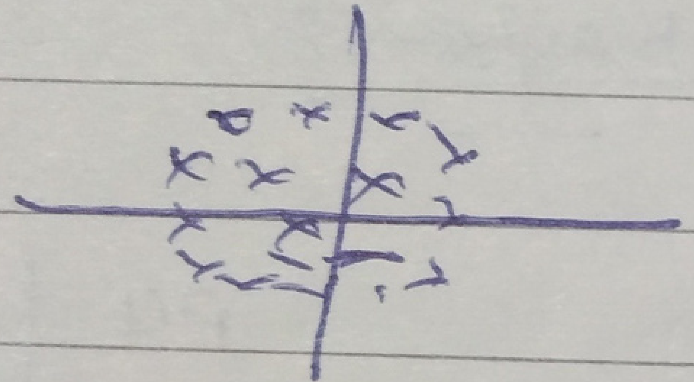
≈ on 90%.

what PCA will produce a set of Principal components which

- Reduce noise, by maximizing feature variance
- Reduce redundancy, by minimizing the covariance b/w pairs of features.

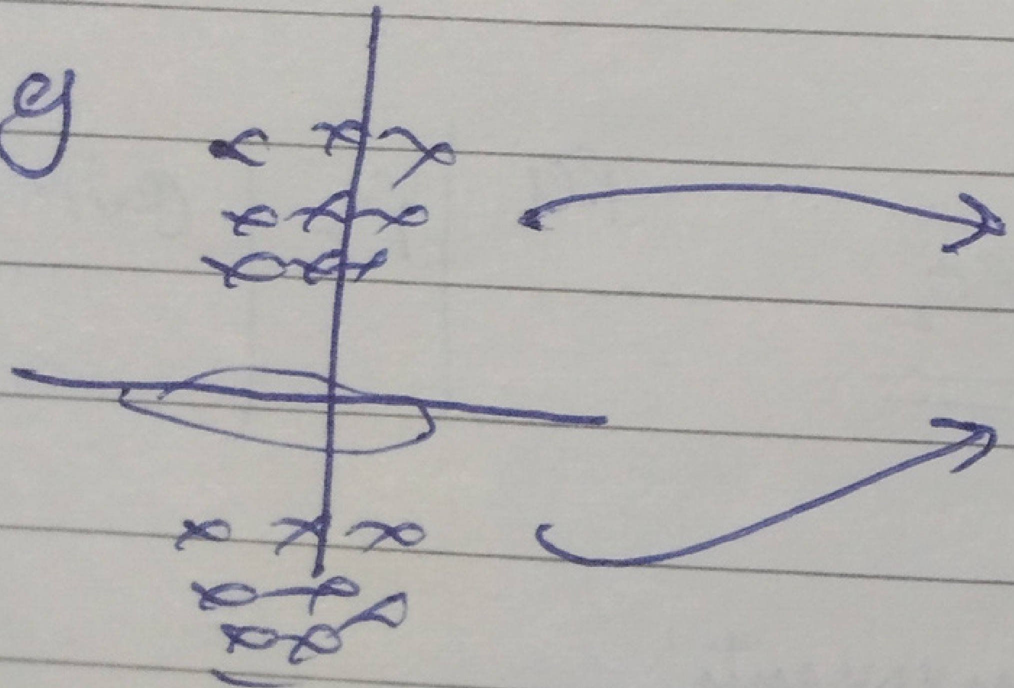
when PCA does not work?

eg

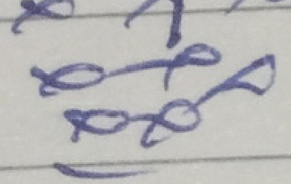


Let data is circle तो किसी भी तरह का transformation करना है रहेगा axes पर same variance

eg

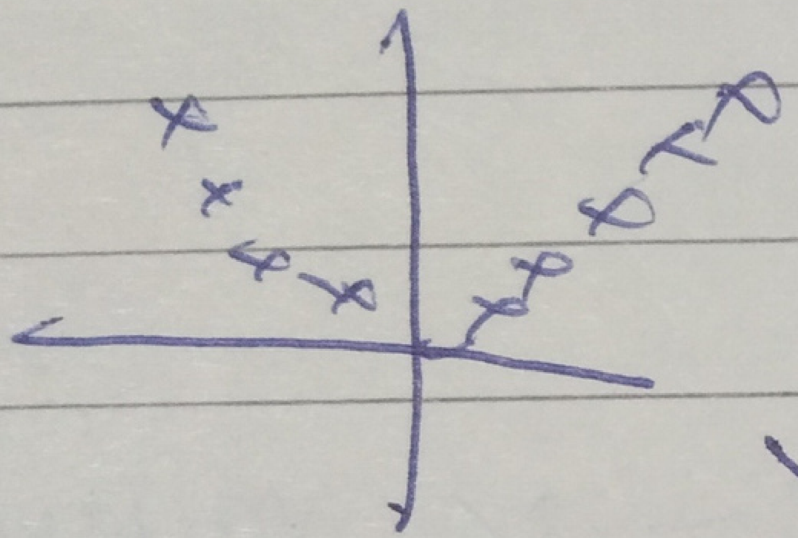


दोनों का project same जगह तो differentiate भी नहीं होगा



data में specify pattern

eg



> यदि ⊥ से लगायेंगे तो special pattern loose हो जायेगा so PCA ज्यादा ~~…~~ अच्छे से काम नहीं करेगा।

To reduce dimensionality, autoencoder is another commonly used method. But, the latent space of the autoencoder is not necessarily uncorrelated. While PCA guarentees that all features are uncorrelated with each other.

PCA. Effect of Normalization → common rep., all feature comparable.

* principal components are → linearly indep, Orthogonal.

Normalization also guarantees that

- Standard SI.

eg. from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

- from sklearn.decomposition import PCA
pca = PCA(n_components = *)

No of comp. Retain करने हैं total

By default None ← All

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

Now fit an any model ←

eg knn

from sklearn.neighbors import KNeighborsClassifier

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy_score(y_test, y_pred)
```

plot →

```
import plotly.express as px
y_train = y_train.astype(str)
fig = px.scatter(x=X_train[:,0], y=X_train[:,1],
                 color=y_train,
color_discrete_sequence = px.colors.qualitative.G10)
fig.show()
```

for 3d

px.scatter_3d ←