

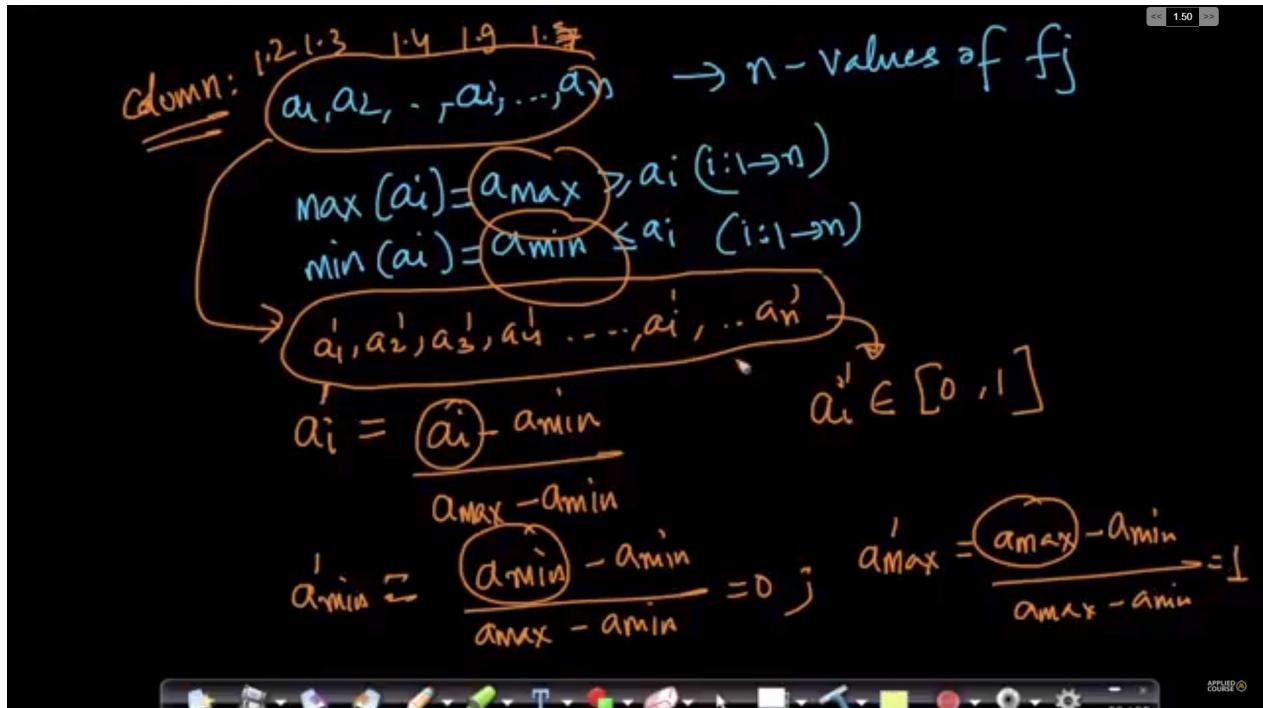
NOTES

Data prepossessing:

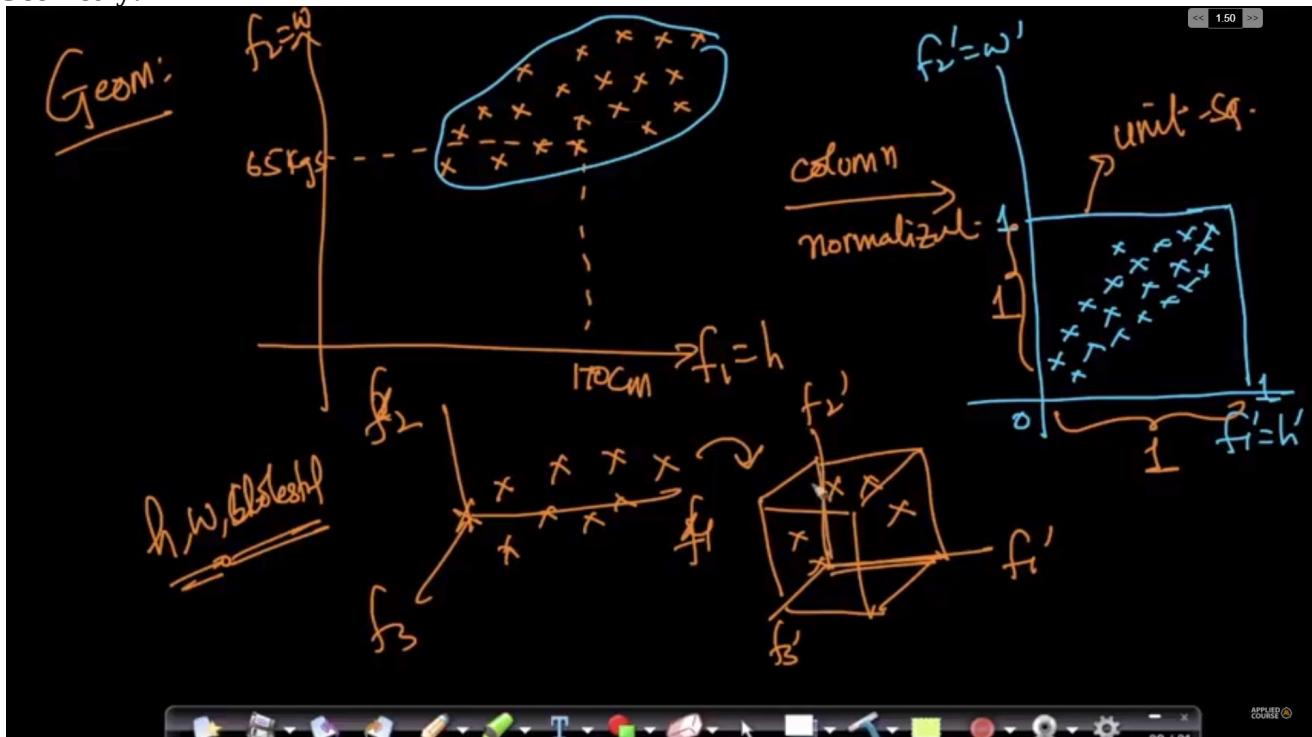
Column Normalization:

getting values between 0 and 1, to get rid of scale of attributes.

Notation:



Geometry:



Column Standardization:

Converts the data into mean=0 and standard deviation = 1.

Mean vector:

Notation:

Mean of a data matrix: Dimensionality reduction and visualization Lecture 6@ Applied AI Course

1.50

$$x_1 = \begin{bmatrix} 2.2 \\ 4.2 \end{bmatrix} \in \mathbb{R}^2$$

$$x_2 = \begin{bmatrix} 1.2 \\ 3.2 \end{bmatrix} \in \mathbb{R}^2$$

$$\underline{\underline{x_1 + x_2 = [3.4, 7.4]}}$$

$$\bar{x} \in \mathbb{R}^d$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{1}{n} (x_1 + x_2 + \dots + x_n)$$

Mean vector

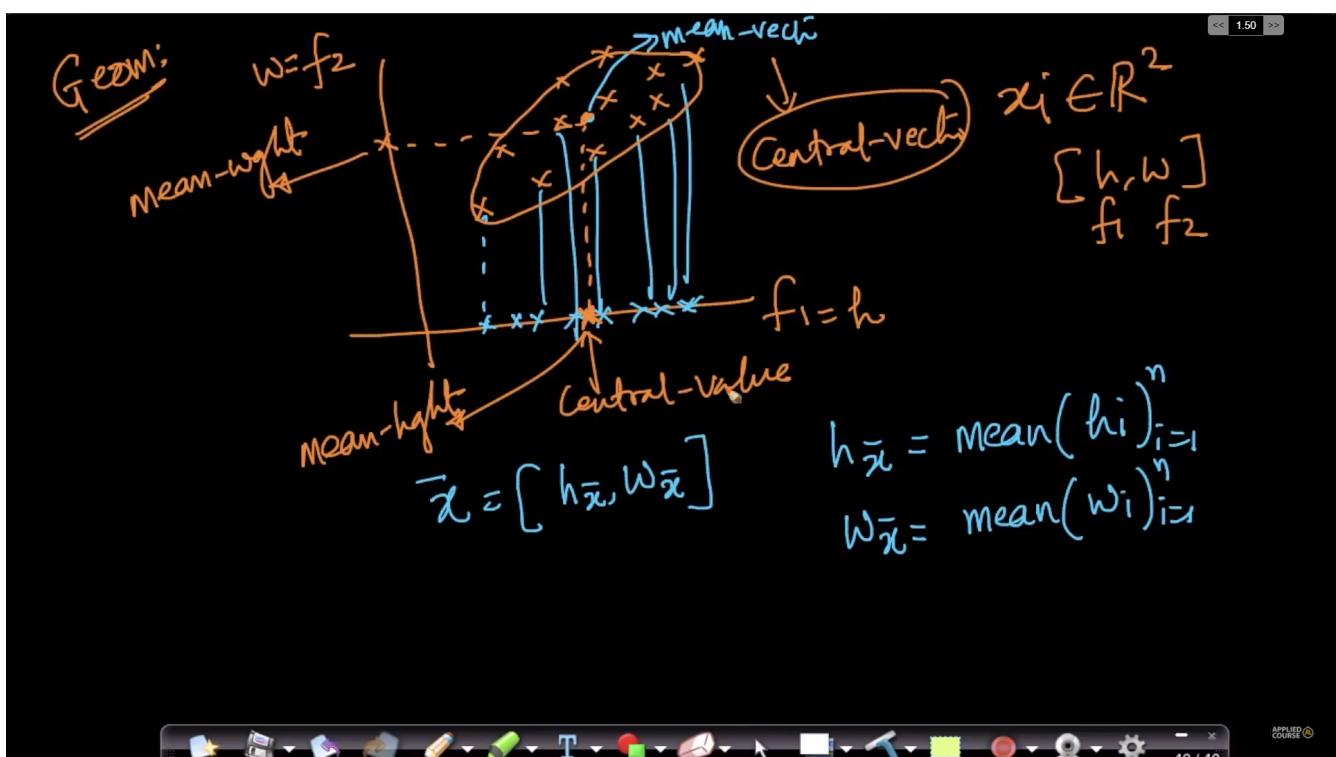
$$x_i \in \mathbb{R}^d$$

2:24 / 5:48

APPLIED COURSE

YouTube

Geometry:



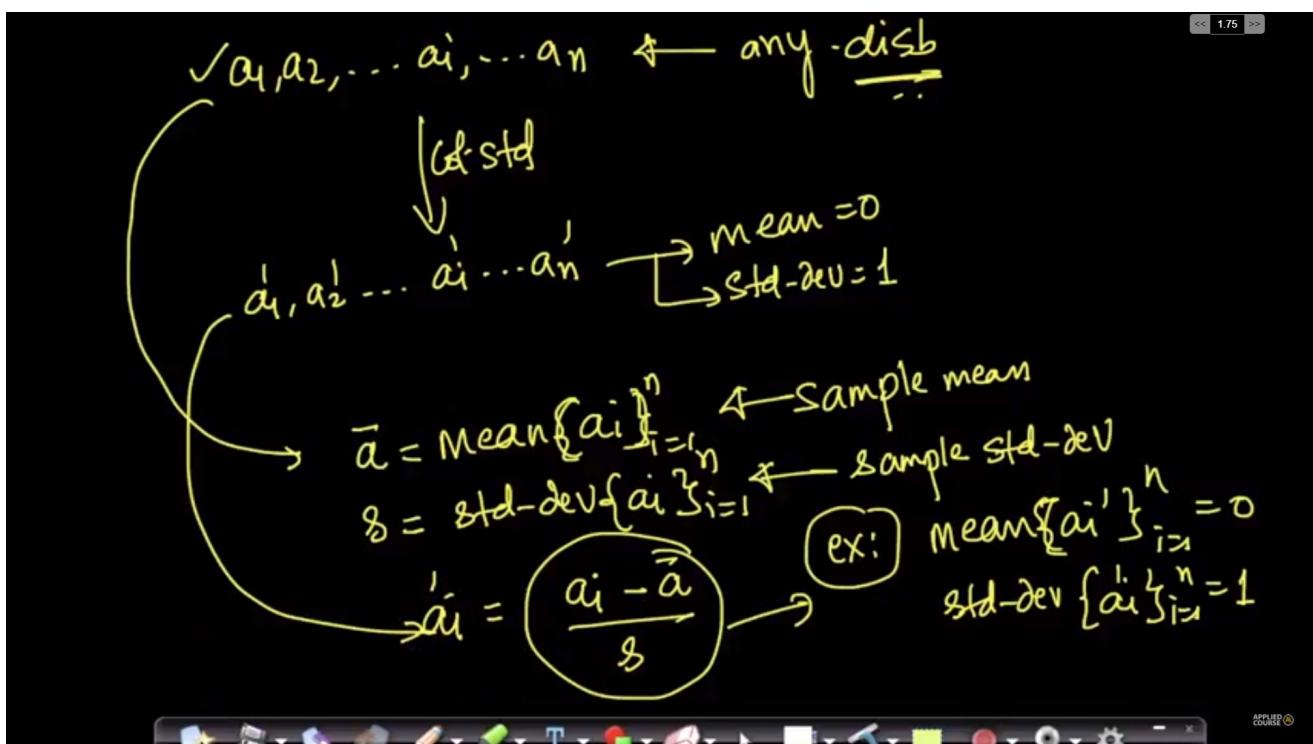
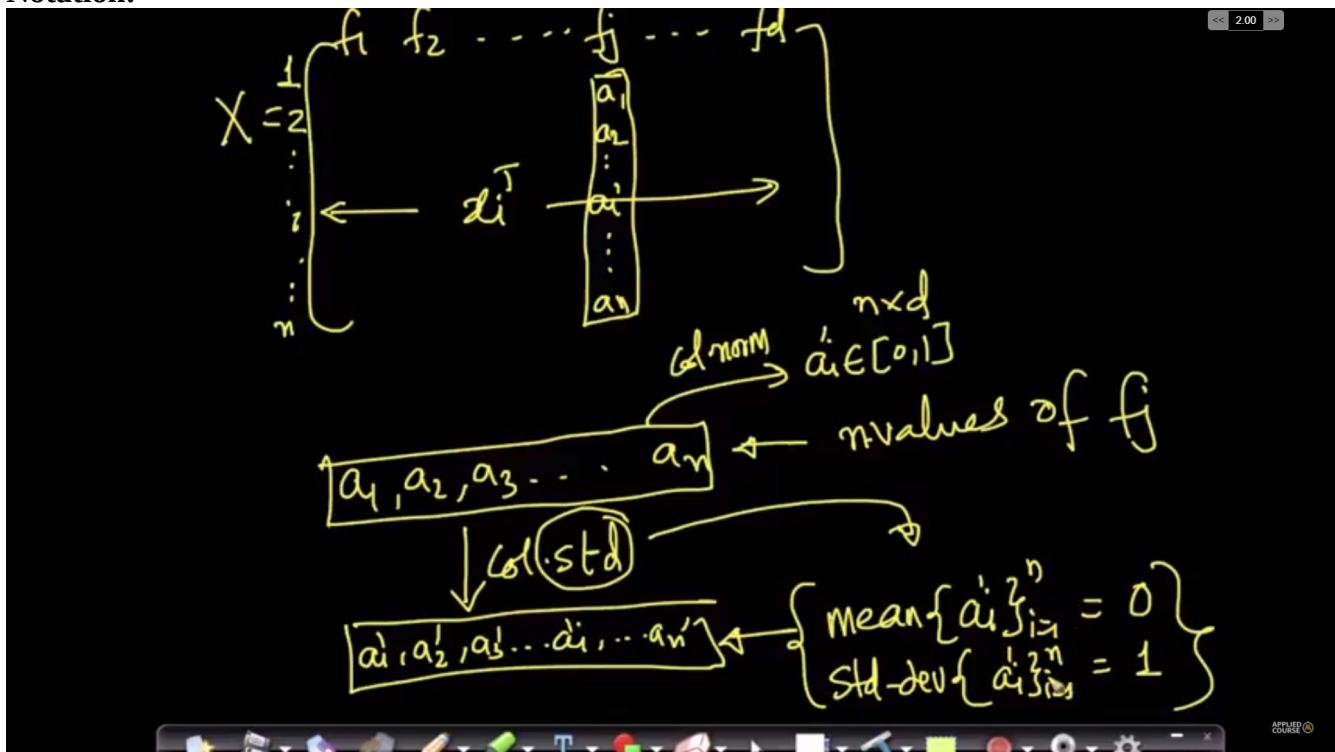
APPLIED COURSE

40 / 40

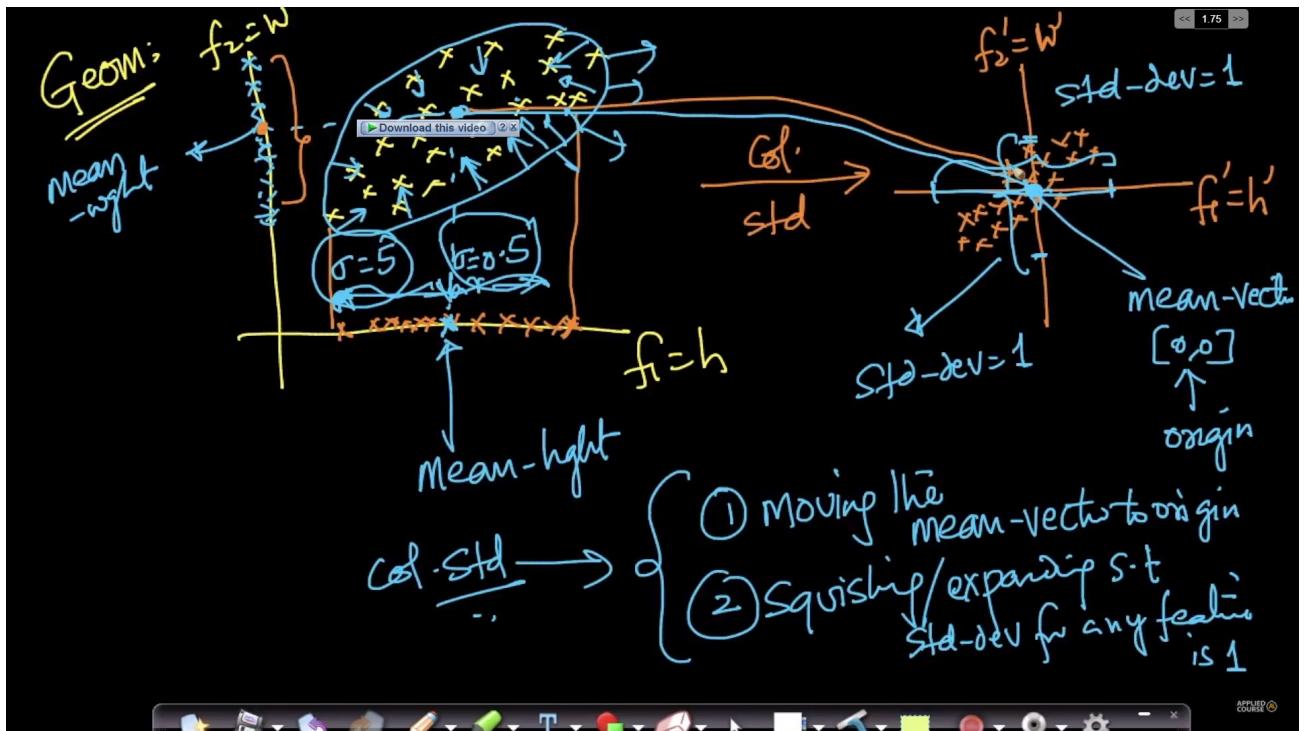
Here mean vector is the central vector of all the data points.

Standardization:

Notation:



Geometry:



Co – Variance of the Data Matrix.

Notation:

$$S_{ij} = \text{cov}(f_i, f_j)$$

$i: 1 \rightarrow d$
 $j: 1 \rightarrow d$

$$\boxed{\text{cov}(x, y) = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_x)(y_i - \mu_y)}$$

$$\text{cov}(f_i, f_j) = \text{Var}(f_i)$$

$$\text{cov}(x, x) = \text{Var}(x)$$

The variance is also called symmetric matrix, along the diagonal we will have variance because

$X = \frac{1}{n} \begin{bmatrix} f_1 & f_2 & \dots & f_d \end{bmatrix}$

Let \textcircled{X} col.-Standardized \Rightarrow mean $\{f_i\} = 0$
 $\text{std-dev}\{f_i\} = 1$

avg

$\text{mean}(f_1)$

$\text{mean}(f_2)$

$\text{cov}(f_1, f_2) = \frac{1}{n} \sum_{i=1}^n (x_{i1} - \mu_1)(x_{i2} - \mu_2)$

μ_1

μ_2

$\text{mean}(f_1)$

it's mu(greek alphabet), μ_1 , μ_2 are 0 ie the means are 0 since data has been column standardised

$S_{d \times d} = \frac{1}{n} (\textcircled{X}^T) (\textcircled{X})_{n \times d} = \textcircled{d \times d} \checkmark$

$\textcircled{(*)} \text{ assuming } X \text{ has been col.-Std}$

LHS $S_{ij} = \text{cov}(f_i, f_j) = \frac{\textcircled{f_i}^T f_j}{n}$

$S_{ij} = \frac{f_i^T f_j}{n}$

The formula $(X\text{-transpose}) * (X)$ holds because the data is column standardized.

PCA:

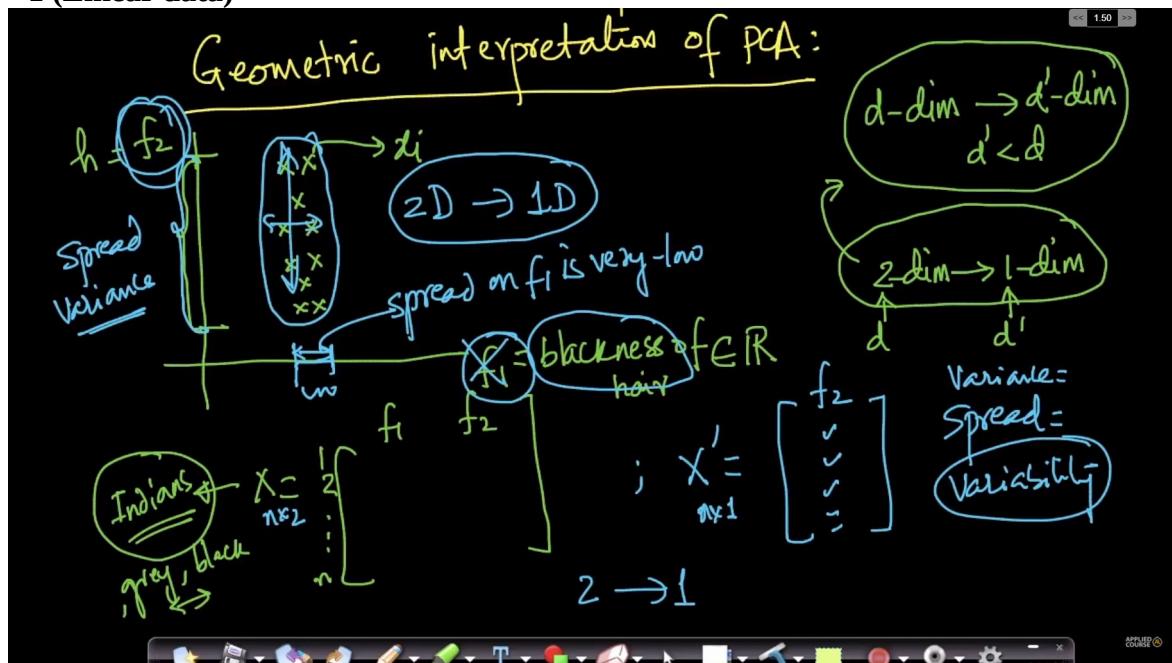
There are two kinds of defining the PCA

1. Variance maximization
2. Distance minimization

Variance Maximization

Geometry:

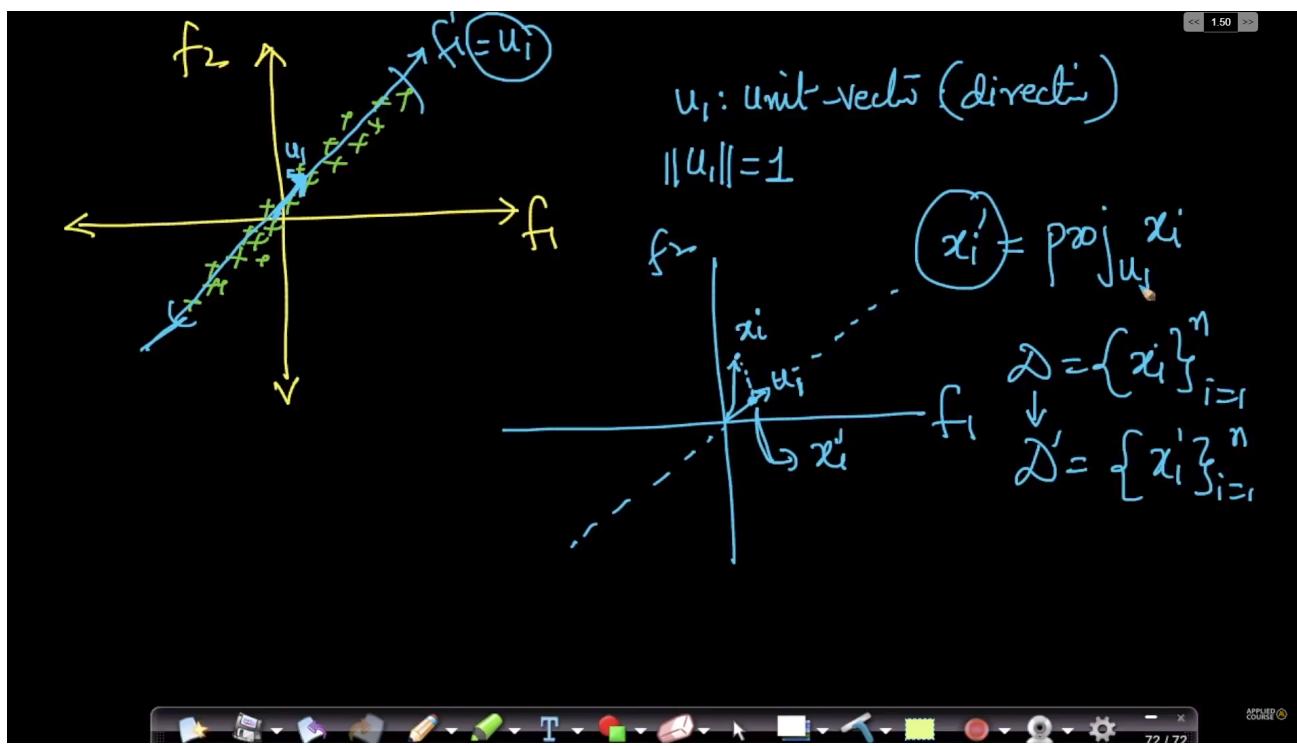
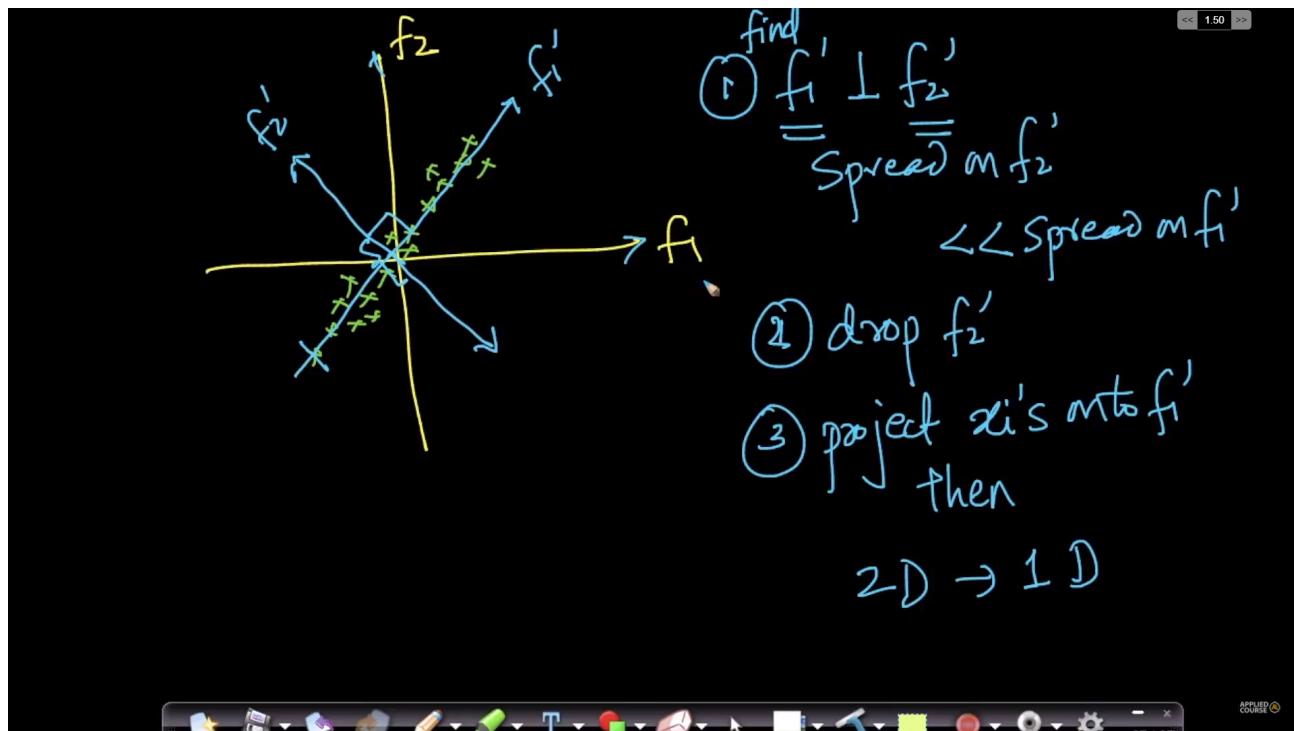
Case – 1 (Linear data)



Here, Skipping the Feature 1 is the good choice as the maximum variance can be captured by feature 2.

Case – 2(inclined data)

Towards the direction of f_1' the maximum variance is captured. Here, We can drop f_2' the perpendicular to f_1' .



Here, the x_i' is the projection on u_i '(unit vector along the maximum variance).

Formulation of u_i :

$$\begin{aligned} \bar{x}_i^{\perp} &= \text{proj}_{u_i} x_i = \frac{u_i \cdot x_i}{\|u_i\|^2 = 1} = \boxed{u_i^T x_i} \\ \bar{x}_i^{\perp} &= u_i^T x_i \\ \check{x}_i^{\perp} &= u_i^T \bar{x} \quad \text{mean } \{x_i\}_{i=1}^n \\ \text{mean } \{x_i\}_{i=1}^n & \end{aligned}$$

The objective is to find the u_i ' vector of maximum projected variance values of x_i on u_i is maximal.

$$\begin{aligned} \textcircled{*} \text{ find } u_i \text{ s.t } \text{Var} \left\{ \text{proj}_{u_i} x_i \right\}_{i=1}^n \text{ is maximal.} \\ \text{Var} \left\{ u_i^T x_i \right\}_{i=1}^n &= \frac{1}{n} \sum_{i=1}^n \left(u_i^T x_i - \underbrace{\underbrace{u_i^T \bar{x}}_{=0}}_{\text{avg } \bar{x}_i} \right)^2 \\ \text{scale} &= \left(\underbrace{u_i^T}_{(1 \times n)} \bar{x} \right)_{(n \times 1)} \\ X &: \text{Col. Standardized} \\ \check{x} &= [0, 0, 0, \dots, 0] \end{aligned}$$

Defining PCA Objective function for variance maximization:

1.50

$$\text{Var} \{x_i'\}_{i=1}^n = \frac{1}{n} \sum_{i=1}^n (u_i^T x_i)^2$$

objective of an optimization problem

$$\max_{u_i} \frac{1}{n} \sum_{i=1}^n (u_i^T x_i)^2$$

Data-matrix

Optimizn problem

s.t. $u_i^T u_i = 1 = \|u\|^2$

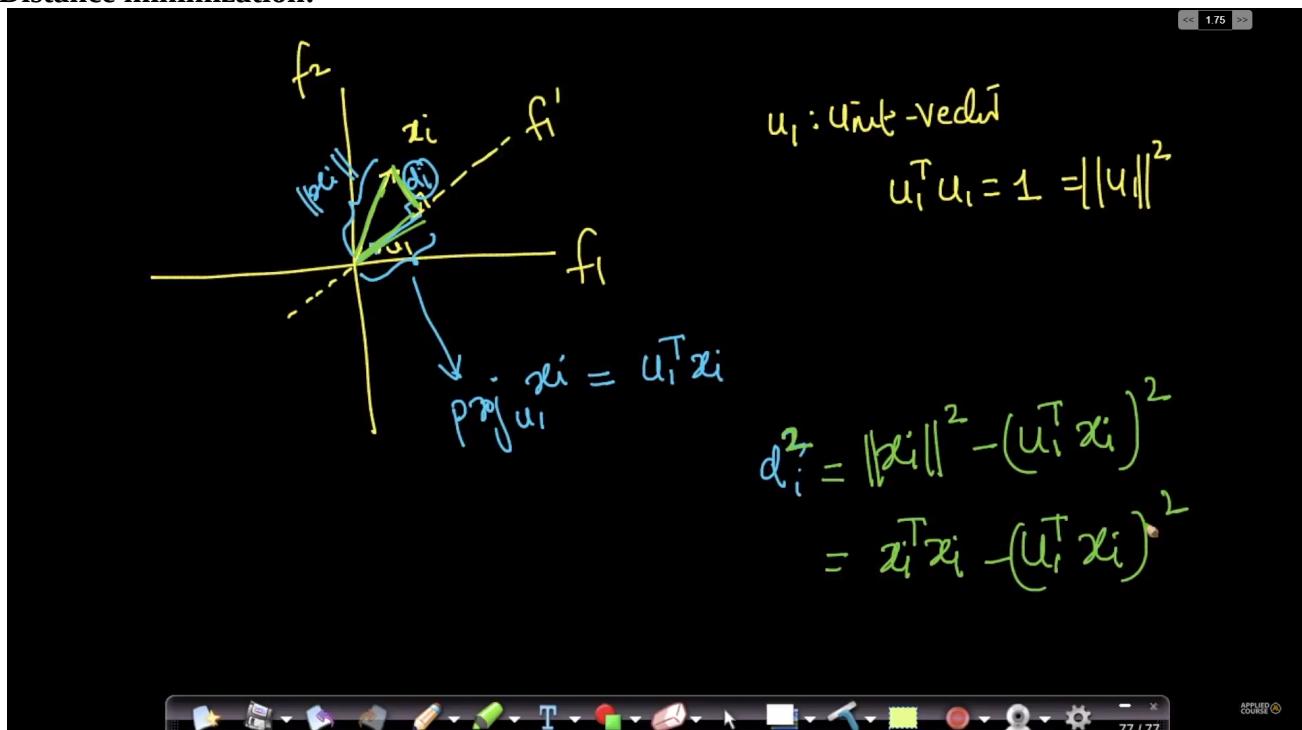
Constraint $\leftarrow u_i \text{ is a unit vec}$

$u_i = [\infty, \infty]$

APPLIED COURSE

The objective function if to maximize the function on **constraint** u_i is a unit vector.

Distance minimization:



Here d_i is the is the distance of the point x_i on the unit vector, the distance d_i must be minimization.

$$\min_{u_i} \sum_{i=1}^n \left(\mathbf{x}_i^T \mathbf{x}_i - (u_i^T \mathbf{x}_i)^2 \right)$$

$\text{s.t. } u_i^T u_i = 1$

$x_i = \begin{bmatrix} \leftarrow x_i^T \rightarrow \end{bmatrix}$

$$\max_{u_i} \frac{1}{n} \sum_{i=1}^n (u_i^T \mathbf{x}_i)^2$$

$\text{s.t. } u_i^T u_i = 1$

} - Variance
Maximize

Here the distance must be minimized with $(u_i^T \mathbf{x}_i)^2$ is equal to 1.

Two kinds of minimization:

Eigen values and Eigen Vectors:

These are used to get the direction of the u_i .

$$S_{d \times d}$$

Maximal eigen-value
 $\lambda_1 > \lambda_2 > \lambda_3 > \lambda_4 \dots \geq \lambda_d$

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 $d \times 1$ Vect \rightarrow

$\lambda_1 v_1 = S v_1$

λ_1 : eigen value of S
 v_1 : eigen vec to S corr. to λ_1

Every pair of Eigen vectors are perpendicular to each other.

Step by step procedure of compunction PCA and obtaining u_i .

Here each eigen vector corresponds to each feature.

Formulation:

$$X = \begin{bmatrix} & & \\ & \checkmark & \\ & & \end{bmatrix}_{n \times d}$$

1. Col. Std of X is done

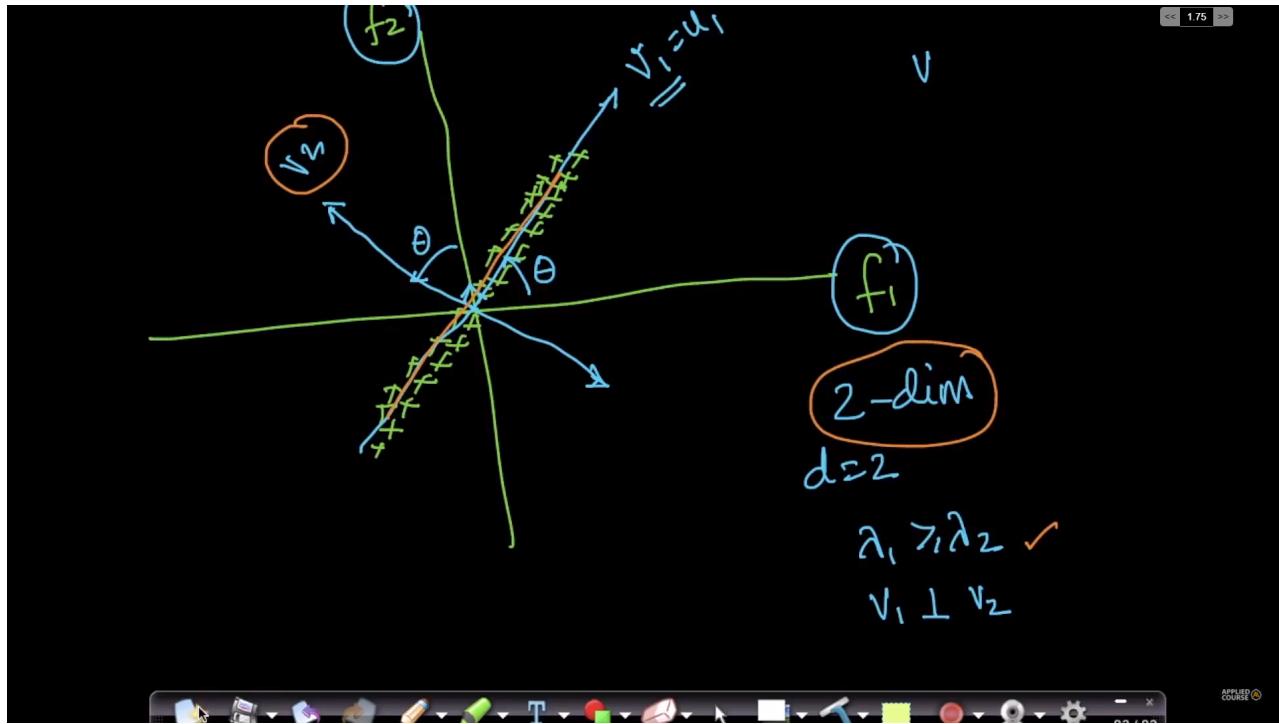
2. $S_{\text{def}} = X^T X$

3. eigen values & vecs of S
 $\lambda_1 > \lambda_2 > \dots > \lambda_d$
 v_1, v_2, \dots, v_d

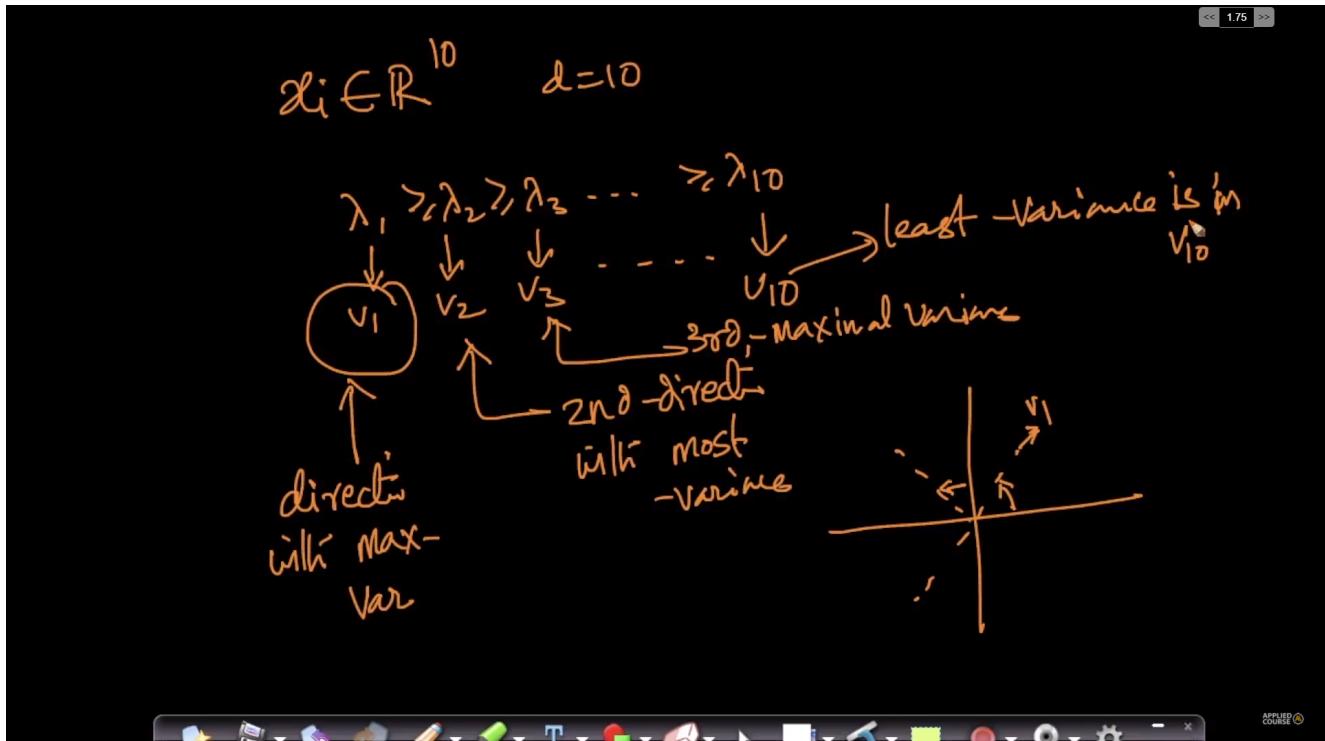
4. $u_1 = v_1$

Geometry:

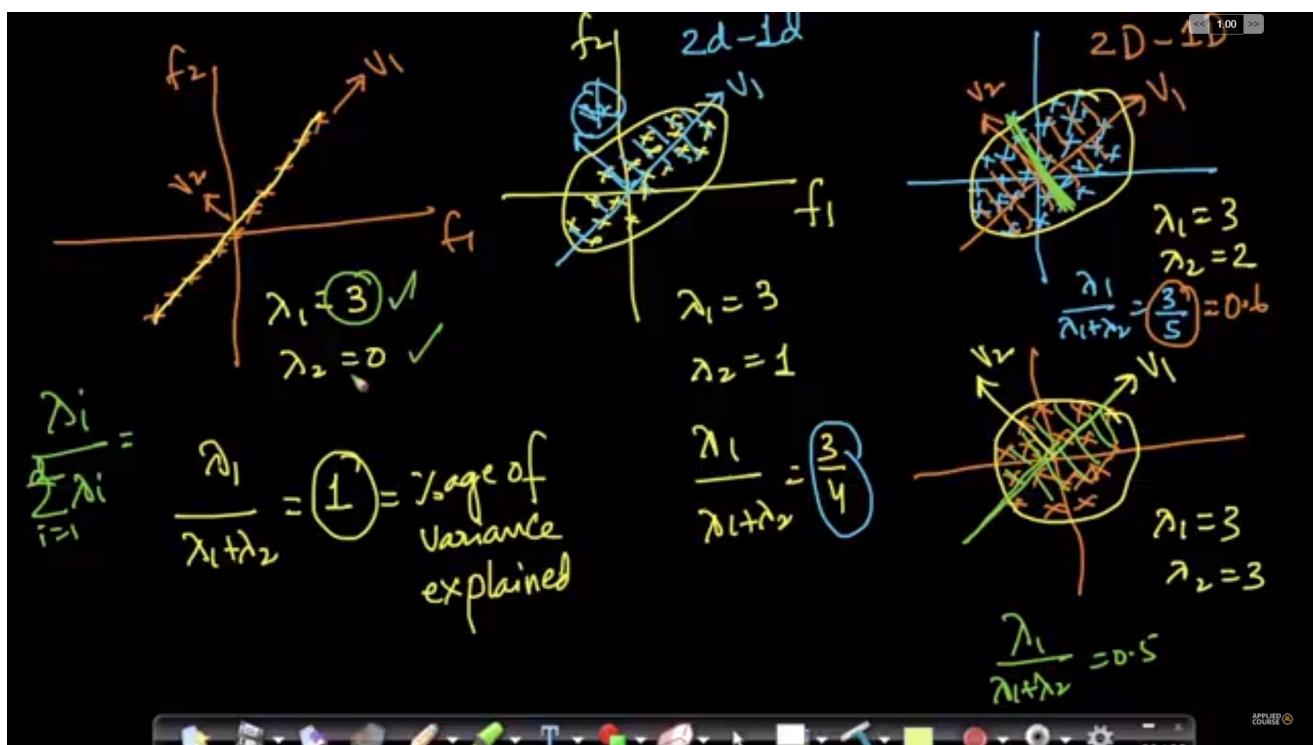
This is for two dimensional data.



This is for 10 dimensional data.



Explanation of lambdas: lambdas explain percentage of variance, the percentage shows



Percentage of variance preserved can be known by lambdas.

Applying PCA:

For visualizing the data the top two eigen vectors are useful

$$X = \begin{bmatrix} f_1 & f_2 & \dots & f_{10} \\ \vdots & \vdots & & \vdots \\ x_1^T & & & x_n^T \end{bmatrix}_{n \times d}$$

$\xrightarrow{\substack{\dim \text{ reduce} \\ (\text{PCA})}}$

$$X' = \begin{bmatrix} V_1 & V_2 \\ \vdots & \vdots \\ x_1^T & x_n^T \end{bmatrix}_{n \times 2}$$

() visualize*

$S = X^T X$

$\text{eigen}(S) = \lambda_1 > \lambda_2 > \lambda_3 \dots > \lambda_{10}$

$x_i' = [x_i^T v_1, x_i^T v_2]$

$v_1 \quad v_2 \quad \dots \quad v_{10}$

For reducing the dimensions of the high dimensional space, we can use the several components.

$$x_i \in \mathbb{R}^{100 \times 1} ; \quad x_i' \in \mathbb{R}^{d'}$$

Press Esc to exit full screen

$d \xrightarrow{\text{PCA}} d'$ $100 \xrightarrow{\text{look}} 100K$ ✓ preserve

Let $\frac{\lambda_1 + \lambda_2 + \dots + \lambda_{51}}{\sum_{i=1}^{100} \lambda_i} = 0.99$ $d' = 51$ $d' < 100$

99% of the variance

Code sample:

In [28]: # PCA for dimensionality reduction (non-visualization)

```

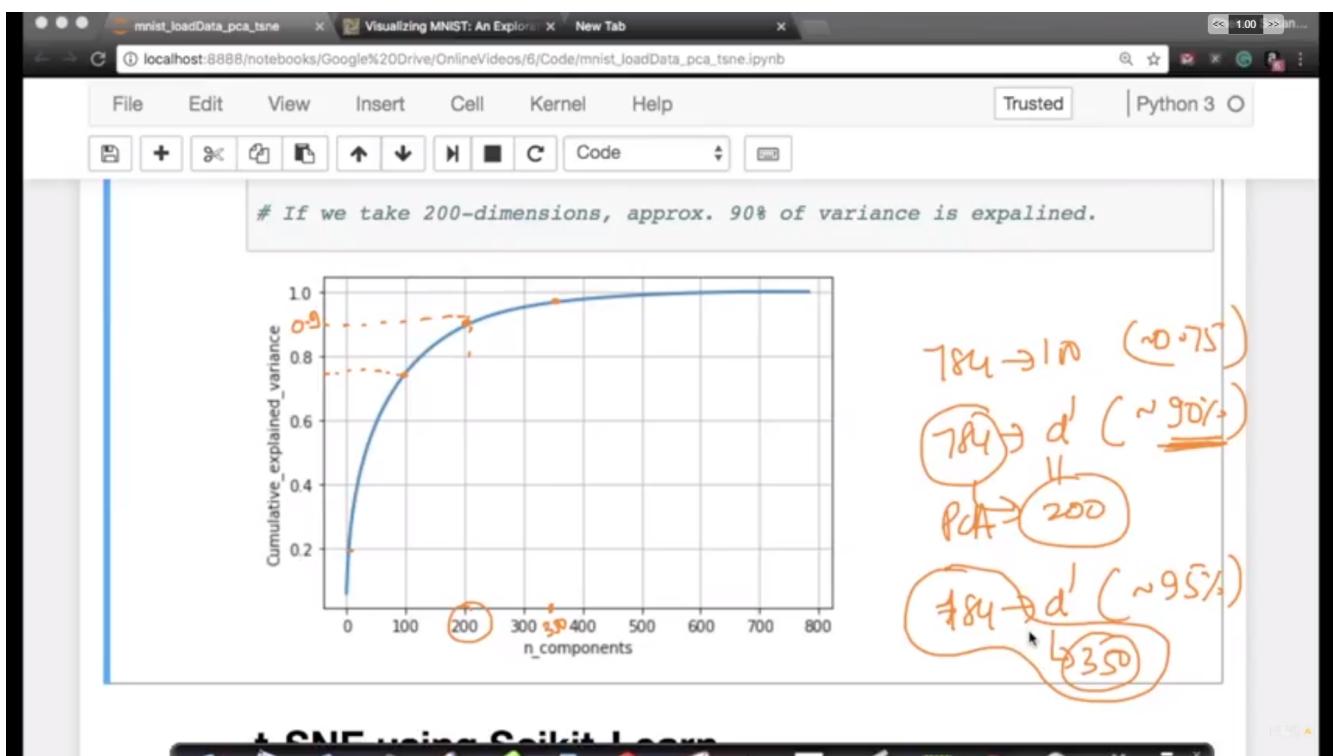
pca.n_components = 784
pca_data = pca.fit_transform(sample_data)

percentage_var_explained = pca.explained_variance_ / np.sum(pca.explained_v
cum_var_explained = np.cumsum(percentage_var_explained)

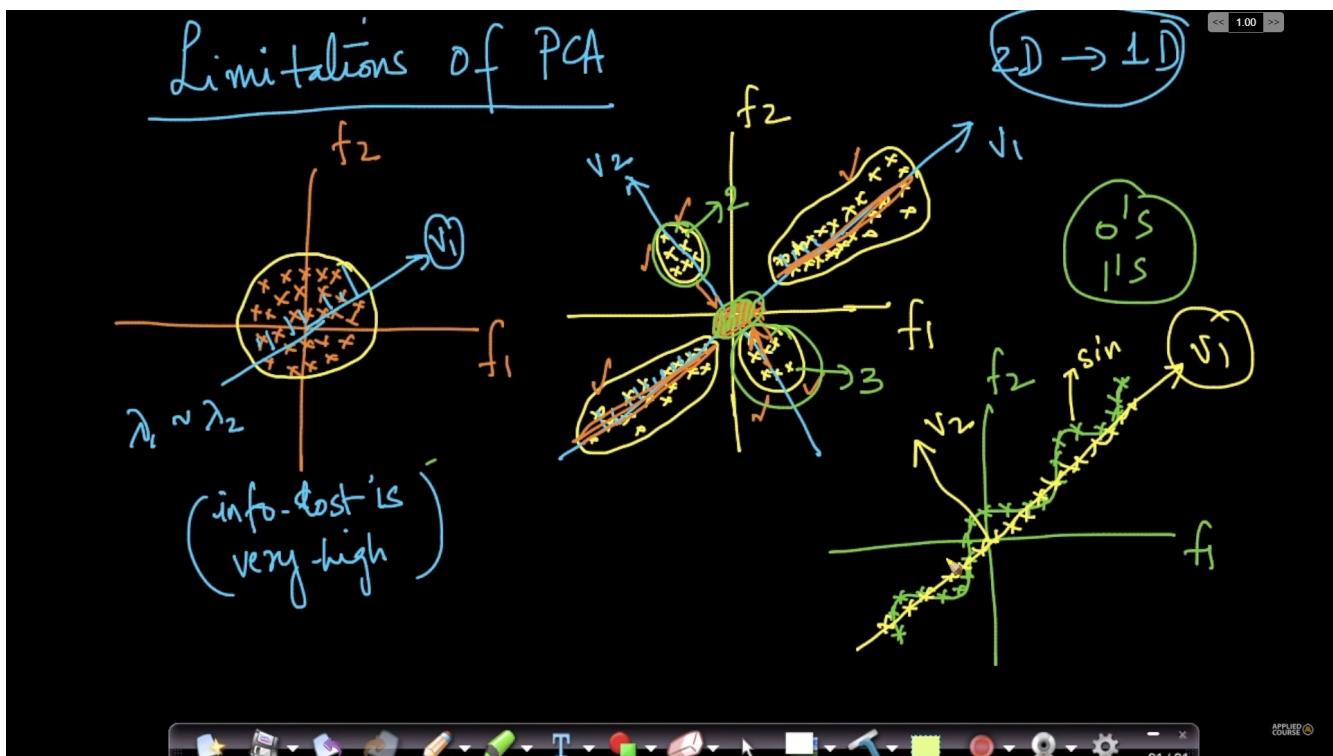
# Plot the PCA spectrum
plt.figure(1, figsize=(6, 4))

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()

```



Limitation of PCA:



Here we PCA cannot work because the projection of the features on 'ui' will overlap on each other. And the relationship of blobs of data that are previously present is lost.

Jupyter notebooks:

<C:\Users\rahul\Downloads\AppliedAIcourse\Notes\13.10#14.9#14.10#15.7.pdf>