Dimensionality Reduction

PCA and SVD

Dimensionality Reduction

- For the analysis of data structures and algorithms and their limits we have:
  - **Computational complexity theory**
  - and analysis of **time** and **space** complexity
  - e.g. Dijkstra's algorithm or Bellman-Ford?
- For the analysis of ML algorithms, there are other questions we need to answer:
  - **Computational learning theory**
  - **Statistical learning theory**

So, what are the key quantities CLT tries to estimate?

- ***Sample complexity***: How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?
- ***Computational complexity***: How much computational effort is needed for a learner to converge (with high probability) to a successful hypothesis?
- ***Mistake bound***: How many training examples will the learner misclassify

Why Reduce Dimensionality?

1. Reduces time complexity: Less computation
2. Reduces space complexity: Less parameters
3. Saves the cost of aquiring the feature
4. Simpler models are more robust
5. Easier to interpret; simpler explanation
6. Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

Feature Selection/Extraction/Construction

- Feature selection: Choosing $k<d$ important features, ignoring the remaining $d - k$

  Subset selection algorithms

- Feature extraction: Project the

  original x_i , i =1,...,d dimensions to

  new $k<d$ dimensions, z_j , j =1,...,k

  Principal components analysis (PCA), linear discriminant analysis (LDA), factor analysis (FA)

- Feature construction: create new features based on old features: f=(…) with f usually being a non-linear function→support vector machines,…

Key Ideas Dimensionality Reduction

- Given a dataset X
- Find a low-dimensional linear projection
- Two possible formulations
  - The variance in low-d is maximized
  - The average projection cost is minimized
- Both are equivalent

Data Preparation Process

- The more disciplined you are in your handling of data, the more consistent and better results you are like likely to achieve. The process for getting data ready for a machine learning algorithm can be summarized in three steps:
- **Step 1**: Select Data
- **Step 2**: Preprocess Data
- **Step 3**: Transform Data

- **Step 1: Select Data**
- This step is concerned with selecting the subset of all available data that you will be working with.
- There is always a strong desire for including all data that is available, that the maxim “more is better” will hold.
- This may or may not be true.

- Below are some questions to help you think through this process:
- What is the extent of the data you have available? For example through time, database tables, connected systems. Ensure you have a clear picture of everything that you can use.
- What data is not available that you wish you had available? For example data that is not recorded or cannot be recorded. You may be able to derive or simulate this data.
- What data don’t you need to address the problem? Excluding data is almost always easier than including data. Note down which data you excluded and why.

- **Preprocess Data**
- After you have selected the data, you need to consider how you are going to use the data. This preprocessing step is about getting the selected data into a form that you can work.

- **Step 3: Transform Data**
- The final step is to transform the process data. The specific algorithm you are working with and the knowledge of the problem domain will influence this step and you will very likely have to revisit different transformations of your preprocessed data as you work on your problem.

- An important machine learning method for dimensionality reduction is called Principal Component Analysis.

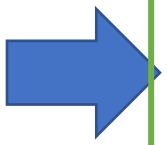
Introduction

- Clustering
  - One way to summarize a complex real-valued data point with a single categorical variable
- Dimensionality reduction
  - Another way to simplify complex high-dimensional data
  - Summarize data with a lower dimentional real valued vector

- Given data points in *d* dimensions
- Convert them to data points in $r<d$ dimensions
- With minimal loss of information

Covariance

- Variance and Covariance:
  - Measure of the “spread” of a set of points around their center of mass(mean)
- Variance:
  - Measure of the deviation from the mean for points in one dimension
- Covariance:
  - Measure of how much each of the dimensions vary from the mean with **respect to each other**



- **Covariance is measured between two dimensions**
- **Covariance sees if there is a relation between two dimensions**
- **Covariance between one dimension is the variance**

Dimension Reduction - 101

PCA

SVD

Principal Component Analysis

Singular Value Decomposition

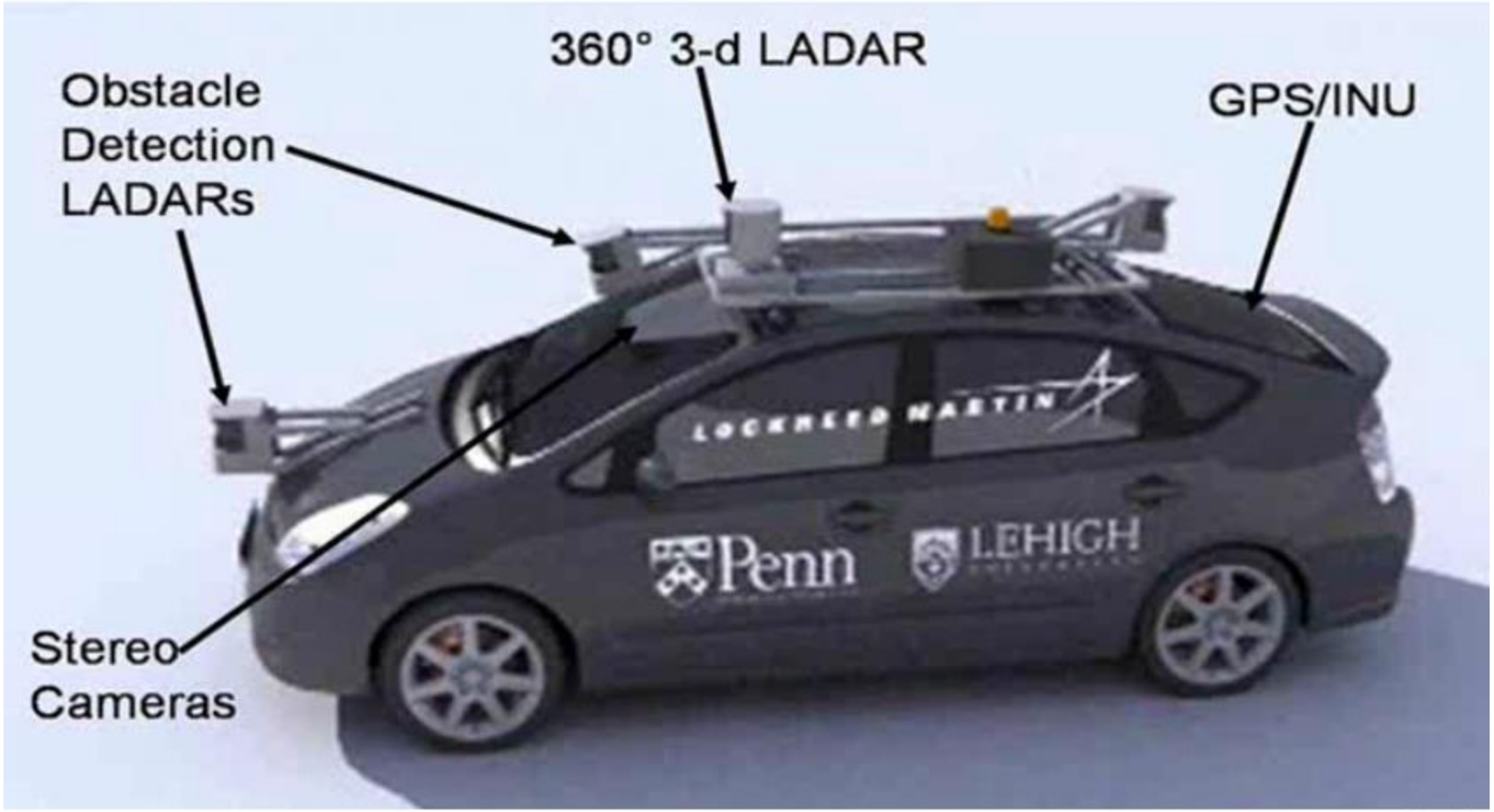
Dimension Reduction Applications

Computational
performance enhanced

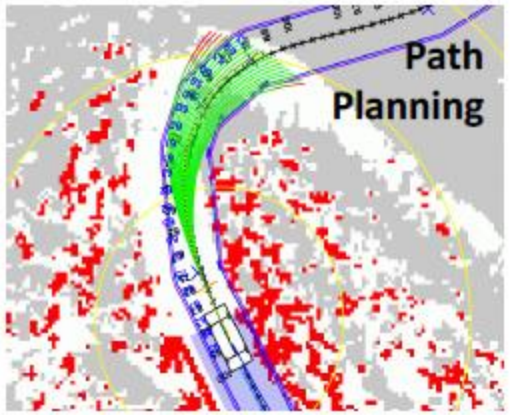
Face recognition

Image compression

Autonomous Car Sensors

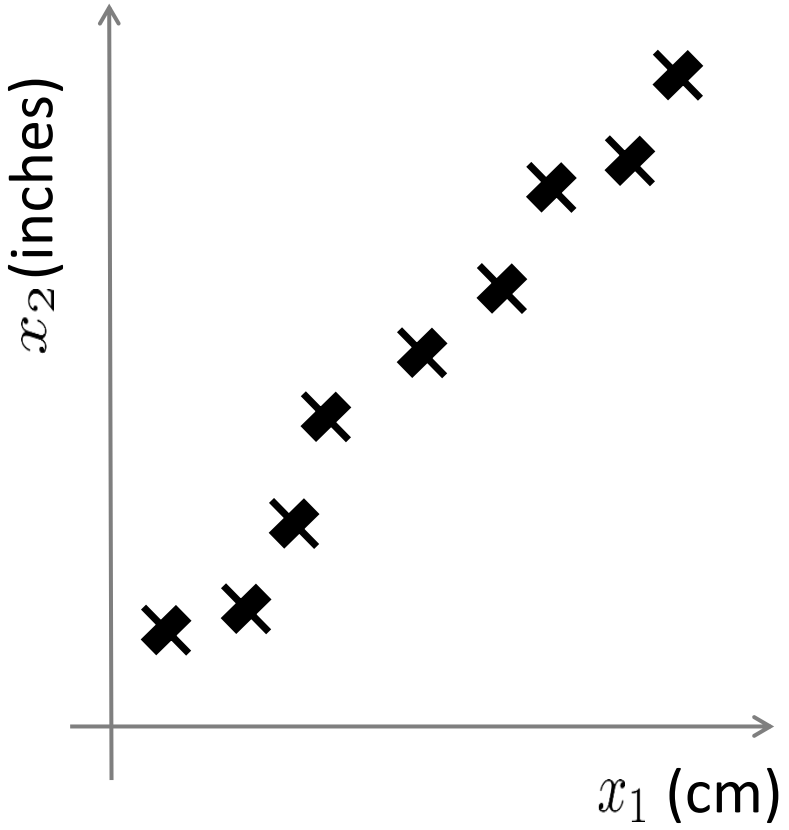


Autonomous Car Technology



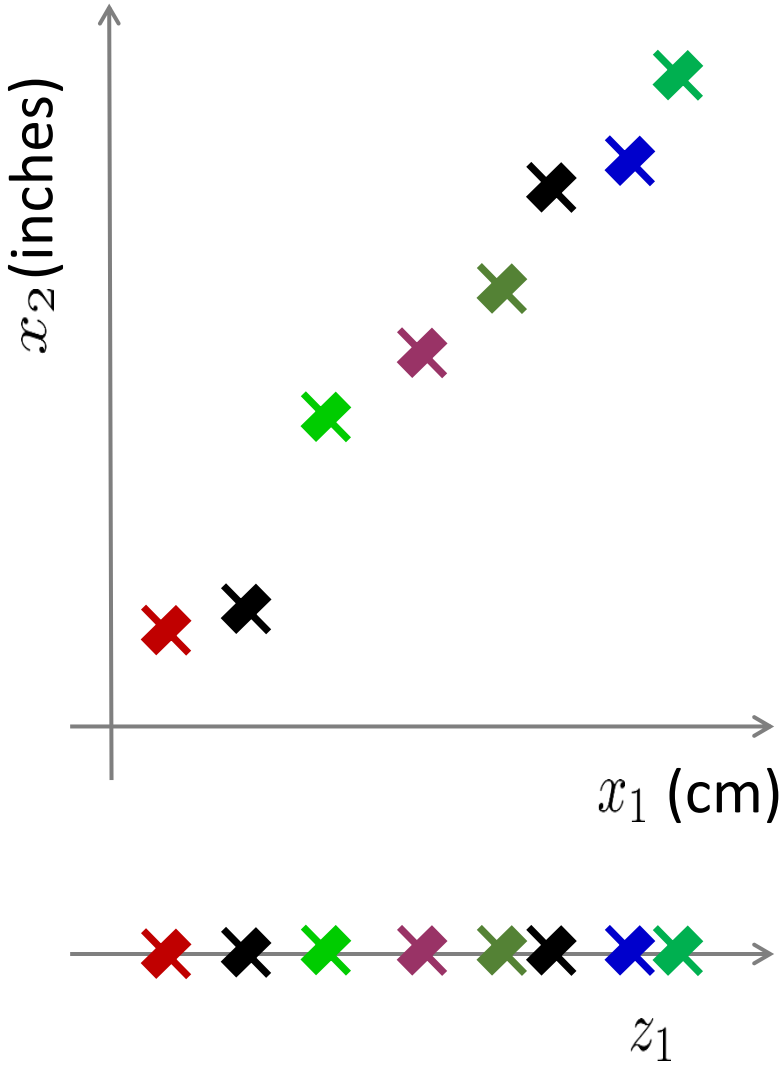
Images and movies taken from Sebastian Thrun's multimedia website.

Data Compression



Reduce data from 2D to 1D

Data Compression



Reduce data from 2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$

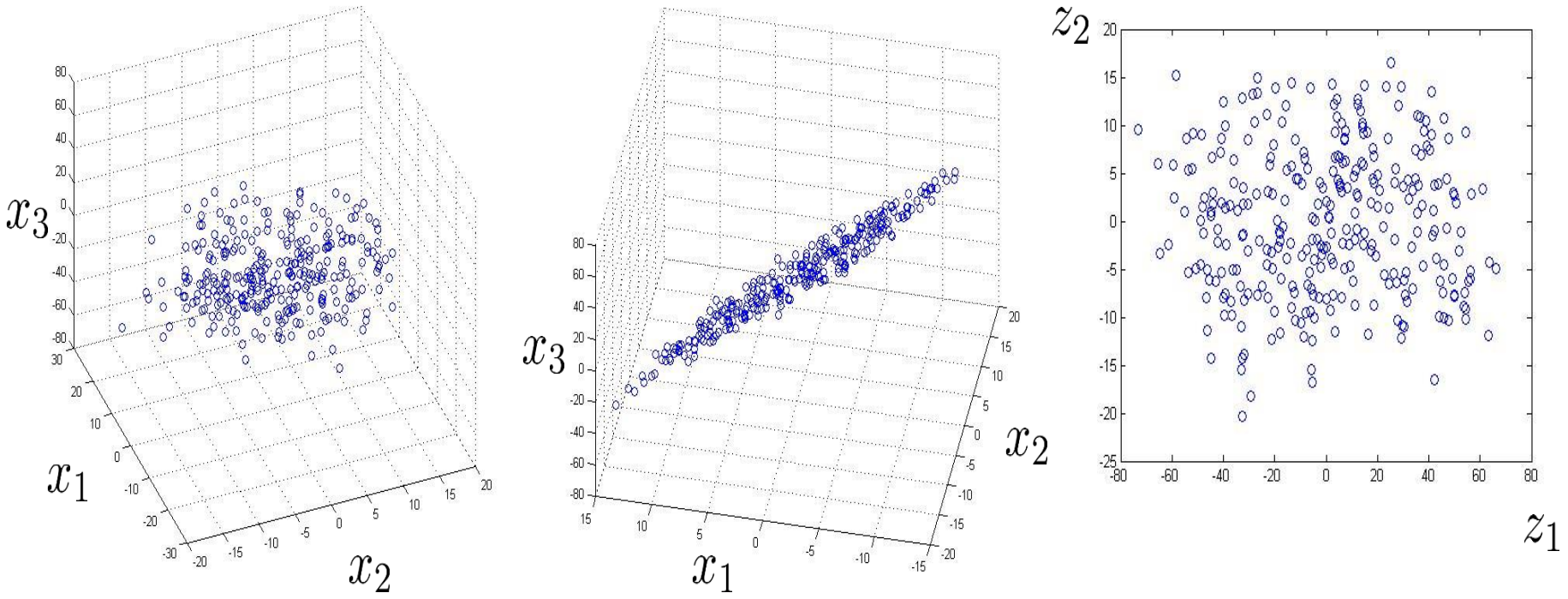
$$x^{(2)} \rightarrow z^{(2)}$$

$$\vdots$$

$$x^{(m)} \rightarrow z^{(m)}$$

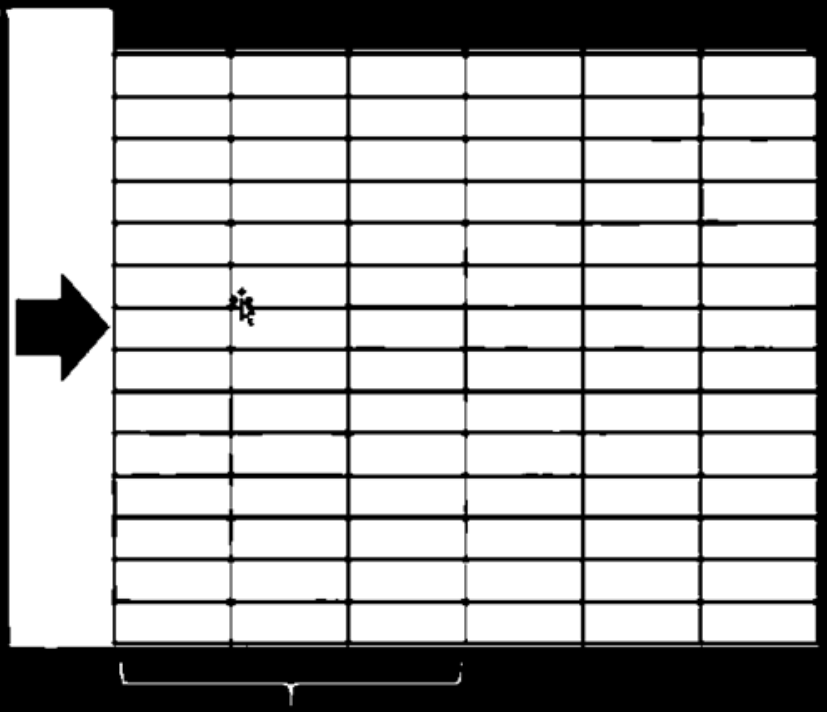
Data Compression

Reduce data from 3D to 2D

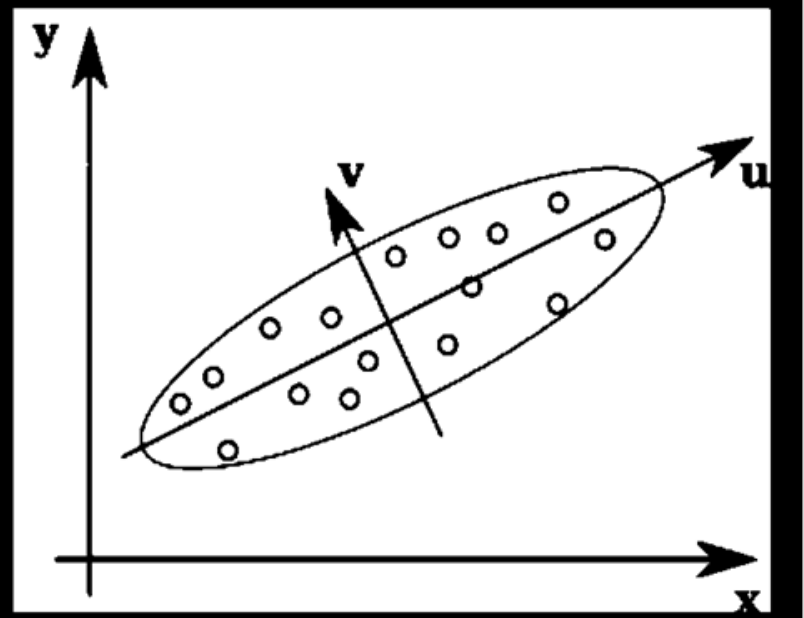


PCA Key Benefit

Brown	1310	89	22	13	22704	94
CalTech	1415	100	25	6	63575	81
CMU	1260	62	59	9	25026	72
Columbia	1310	76	24	12	31510	88
Cornell	1280	83	33	13	21864	90
Dartmouth	1340	89	23	10	32162	95
Duke	1315	90	30	12	31585	95
Georgetown	1255	74	24	12	20126	92
Harvard	1400	91	14	11	39525	97
JohnsHopkins	1305	75	44	7	58691	87
MIT	1380	94	30	10	34870	91
Northwestern	1260	85	39	11	28052	89
NotreDame	1255	81	42	13	15122	94
PennState	1081	38	54	18	10185	80



Less number of columns will capture maximum information



PCA Preliminaries

Input (Before PCA)

- *'n' original columns*

PCA

Output (After PCA)

- *'n' principal components ('n' weighted averages of original measurements)*

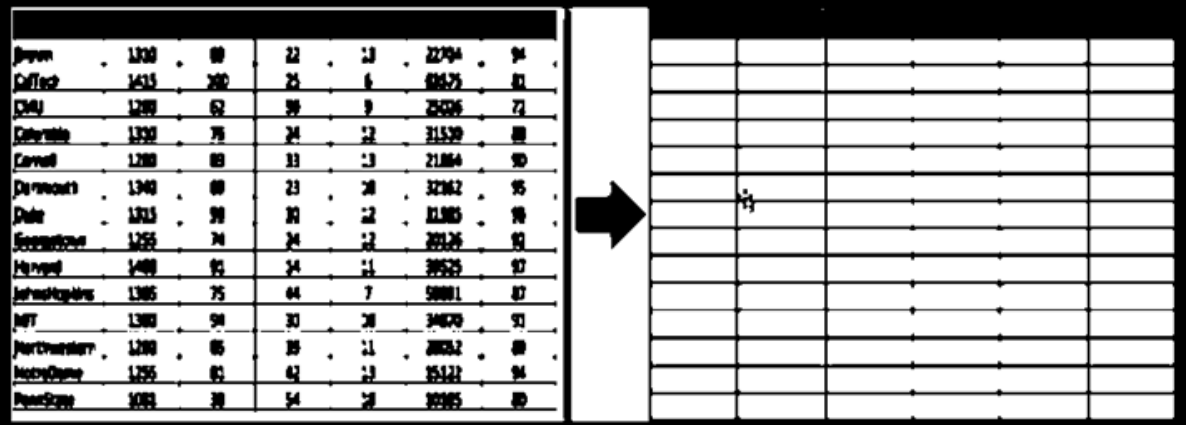
Input (Before PCA)

- *Correlated*

Output (After PCA)

- *Uncorrelated*
- *Ordered by variance*
- *Keep top principal components; drop rest*

PCA Weights Calculation



The i^{th} principal component is a weighted average of original measurements / columns:

$$PC_i = a_{i1}SAT + a_{i2}Top10 + a_{i3}Accept + \ldots + a_{i6}GradRate$$

where $a_i^T = [a_{i1}, a_{i2}, \ldots, a_{i6}]^T$ is a unit vector

Weights (a_{ij}) are chosen such that:

1. PCs are ordered by their variance (PC_1 has largest variance, followed by PC_2, PC_3, and so on)
2. Pairs of PCs have correlation = 0
3. For each PC, sum of squared weights = 1 (Unit Vector) $\sum_{j=1}^{6} a_{ij}^2 = 1$

PCA Weights Calculation

$PC_i = a_{i1}SAT + a_{i2}Top10 + a_{i3}Accept + ... + a_{i6}GradRate$

High Variance = Lots of Information

$$Var(PC_i) = a^2_{i1}\,Var(X_1) + a^2_{i2}\,Var(X_2) + a^2_{i3}\,Var(X_3) + \dots + a^2_{in}\,Var(X_n) + 2a_{i1}a_{i2}\,Cov(X_1, X_2) + \dots + 2a_{i,n-1}a_{in}\,Cov(X_{n-1}, X_n) = \boldsymbol{a}_i^T \sum \boldsymbol{a}_i$$

Also, $Covar(PC_i, PC_j) = 0$*, when* $i \neq j$

- Find weights a_{ij} that maximize variance of PC_i, while keeping PC_i uncorrelated to other PCs
- The covariance matrix of the Xs is needed
- Apply Eigenvector & Eigenvalue decomposition to obtain the weights
- Standardize the data before applying PCA

Univ	Z-SAT	Z-Top10	Z-Accept	Z-SFRatio	Z-Expenses	ZGradRate
Brown	0.4019342	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
CalTech	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
CMU	[illegible]	0.7450836	[illegible]	[illegible]	[illegible]	[illegible]
Columbia	0.4019342	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Cornell	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Dartmouth	[illegible]	0.64423451	[illegible]	[illegible]	[illegible]	[illegible]
Duke	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Georgetown	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.58232445
Harvard	[illegible]	[illegible]	[illegible]	0.4228738	[illegible]	[illegible]
JohnsHopkins	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
MIT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Northwestern	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
NotreDame	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
PennState	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Princeton	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Purdue	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Stanford	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
TexasA&M	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
UCBerkeley	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
UChicago	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
UMichigan	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
UPenn	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
UVA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.58232445
UWisconsin	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Yale	[illegible]	[illegible]	[illegible]	0.4228738	[illegible]	[illegible]

PCA Weights Calculation

Univ	Z-SAT	Z-Top10	Z-Accept	Z-SFRatio	Z-Expenses	ZGradRate
Brown	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
CalTech	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
CMU	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Columbia	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Cornell	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Dartmouth	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Duke	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Georgetown	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Harvard	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Johns Hopkins	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
MIT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Northwestern	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Notre Dame	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Pennsylvania	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Princeton	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Purdue	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Stanford	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Tufts	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
UC Berkeley	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
UC Irvine	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
UCLA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Wash U	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Yale	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Principal Components

Pattern\Component	1	2	3	4	5	6
SAT	-0.457749	0.0896804	-0.167029	0.1512403	0.7206458	-0.458055
Top10	0.4270144	0.199315	0.4970825	0.3748557	0.4620162	0.3560749
Accept	0.4244316	0.320883	0.156279	0.3612572	0.3010596	-0.216934
SFRatio	0.3905483	0.4532564	0.5606231	0.507331	0.2768157	-0.172049
Expenses	-0.362535	0.6544864	-0.208741	-0.515421	0.1725477	0.1737551
GradRate	-0.379404	0.5515354	0.5534726	-0.438633	0.3381256	0.2085175

$$(-0.457)(0.40) + (-0.427)(0.64) + (0.424)(-0.87) + (0.39)(0.06) + (-0.362)(-0.32) + (-0.379)(0.80) = -0.989$$

Scores

Pattern\Component	1	2	3	4	5	6
1	0.989471	1.042806	-0.079428	0.0558	-0.126153	0.0339496
2	-2.765217	2.2134026	-0.819921	0.1409384	-0.123417	0.1770522
3	1.0899894	1.5982518	0.2613967	1.0533541	-0.187939	-0.338695
4	0.726755	0.041335	0.059278	0.154032	0.565941	0.106958
5	-0.30561	-0.622409	-0.010034	0.1679014	0.012469	0.0165794
6	-1.662411	-0.337406	0.2482876	0.0123105	-0.051597	0.0266173
7	-1.22163	-0.481064	0.0314454	-0.201489	0.2875753	0.0764358
8	-0.331906	-0.769305	0.4836124	0.039051	-0.533968	0.1522325
9	-2.326183	-0.378729	-0.113753	-0.444206	-0.225459	-0.26159
10	1.374925	2.0766921	0.4338802	0.619763	0.2254055	0.2308547

PCA Goals: Data Reduction & Weights

Principal Components

Feature\Component	1	2	3	4	5	6
SAT	-0.457749	0.0396804	-0.187039	0.1312403	0.0206458	-0.858055
Top10	0.427144	0.199932	0.497809	0.3748957	0.4820162	0.3960749
Accept	0.424308	0.320893	0.156279	0.0612872	0.8010936	0.216934
SFRatio	0.3906483	-0.432564	-0.606081	-0.507391	0.0768237	-0.172048
Expenses	-0.362523	0.6344864	-0.204741	-0.623401	0.0725477	0.1737631
GradRate	-0.379404	-0.515554	0.5324726	-0.438633	0.3381096	0.0035375

Variances

	1	2	3	4	5	6
Variance	4.6120851	0.7868161	0.2865619	0.1637801	0.1243062	0.0264506
Variance Percentage	76.868084	13.113602	4.776031	2.7296685	2.07177	0.4408438
Cumulative Variance %	76.868084	89.98168?	94.757718	97.487386	99.559156	100

- PCs are ordered by variance
- PC1 captures 76.86% of the information
- First 2 PCs capture 89.98% of the information
- Choose first few PCs & drop the rest

- Learn relationship with PCA by interpreting the weights
- $a_{i1}, a_{i2}, a_{i3}, \ldots\ldots, a_{ip}$ are the coefficients for PC_i
- Coefficients describe the role of original variable in computing PC_i
- They are useful in providing context-specific interpretation of each PC

✓ PC_1 weights measure the degree of:

- High Accept, SFRatio
- Low Expenses, GradRate, SAT, Top10

Principal component analysis (PCA) involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called *principal components*.

The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

Principal Component Analysis

- It is a method that uses simple matrix operations from linear algebra and statistics to calculate a projection of the original data into the same number or fewer dimensions.
- It can be thought of as a projection method where data with m-columns (features) is projected into a subspace with m or fewer columns, whilst retaining the essence of the original data.
- The PCA method can be described and implemented using the tools of linear algebra.

PRINCIPAL COMPONENT ANALYSIS: PCA

- From a set of N correlated descriptors, we can derive a set of N uncorrelated descriptors (the principal components).
- Each principal component (PC) is a suitable linear combination of all the original descriptors.
- PCA reduces the information dimensionality that is often needed from the vast arrays of data in a way so that there is minimal loss of information

I

Functionality 1 = $\mathcal{F}$ (x_1 , x_2 , x_3 , x_4 , x_5 , x_6 , x_7 , x_8 .....)

Functionality 2 = $\mathcal{F}$ (x_1 , x_2 , x_3 , x_4 , x_5 , x_6 , x_7 , x_8 .....)

II

X1 = f (x2)

X2 = g(x3)

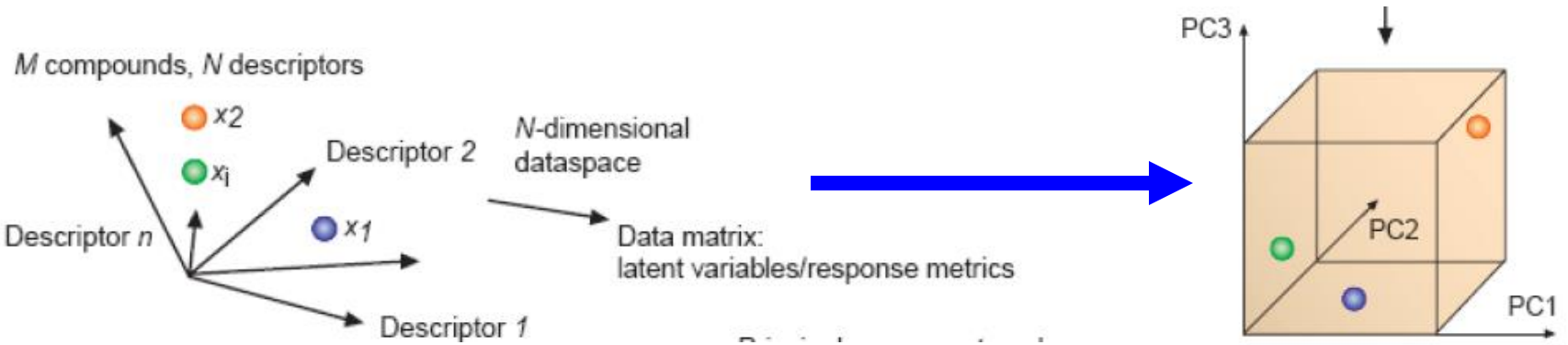
X3= h(x4)

III

PC 1= $A_1 X_1 + A_2 X_2 + A_3 X_3 + A_4 X_4$ .....

PC 2 = $B_1 X_1 + B_2 X_2 + B_3 X_3 + B_4 X_4$ .....

PC 3 = $C_1 X_1 + C_2 X_2 + C_3 X_3 + C_4 X_4$ .....



Mathematically, PCA relies on the fact that most of the descriptors are interrelated and these correlations in some instances are high. It results in a rotation of the coordinate system in such a way that the axes show a maximum of variation (covariance) along their directions.

This description can be mathematically condensed to a so-called eigenvalue problem.

- The data manipulation involves decomposition of the data matrix X into two matrices P and T(Eigen Vector matrix). The two matrices P and T(Eigen value matrx) are orthogonal. The matrix P is usually called the loadings matrix, and the matrix T is called the scores matrix.

- The eigenvectors of the covariance matrix constitute the principal components. The corresponding eigenvalues give a hint to how much "information" is contained in the individual components.

$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$

Determinant of A

$$|A| = ?$$

$$A = \begin{bmatrix} 8 & & \\ & -3 & -2 \\ & -4 & 1 \end{bmatrix}$$

Determinant of A

$$|A| = ?$$

$$|A| = 8\begin{bmatrix} -3 & -2 \\ -4 & 1 \end{bmatrix} - (-8)\begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix} + (-2)\begin{bmatrix} 4 & -3 \\ 3 & -4 \end{bmatrix}$$

$$A = \begin{bmatrix} \blacksquare & -8 & \blacksquare \\ 4 & \blacksquare & -2 \\ 3 & \blacksquare & 1 \end{bmatrix}$$

Determinant of A

$$|A| = ?$$

$$|A| = 8\begin{bmatrix} -3 & -2 \\ -4 & 1 \end{bmatrix} - (-8)\begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix} + (-2)\begin{bmatrix} 4 & -3 \\ 3 & -4 \end{bmatrix}$$

$$A = \begin{bmatrix} & & -2 \\ 4 & -3 & \\ 3 & -4 & \end{bmatrix}$$

Determinant of A

$$|A| = ?$$

$$|A| = 8\begin{bmatrix} -3 & -2 \\ -4 & 1 \end{bmatrix} - (-8)\begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix} + (-2)\begin{bmatrix} 4 & -3 \\ 3 & -4 \end{bmatrix}$$

$$A = \begin{bmatrix} 4 & -3 & -2 \\ 3 & -4 & \end{bmatrix}$$

Determinant of A

$$|A| = ?$$

$$|A| = 8\begin{bmatrix} -3 & -2 \\ -4 & 1 \end{bmatrix} - (-8)\begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix} + (-2)\begin{bmatrix} 4 & -3 \\ 3 & -4 \end{bmatrix}$$

$$|A| = 8(-3-8) + 8(4+6) - 2(-16+9)$$

$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$

Determinant of A

$$|A| = 6$$

$$|A| = 8\begin{bmatrix} -3 & -2 \\ -4 & 1 \end{bmatrix} - (-8)\begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix} + (-2)\begin{bmatrix} 4 & -3 \\ 3 & -4 \end{bmatrix}$$

$$|A| = 8(-3-8) + 8(4+6) - 2(-16+9)$$

$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$

Eigen Value

How to calculate sum of diagonal minors

$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$

Eigen Value

$$\lambda^3 - [Sum\ of\ Diagonal\ Elements]\lambda^2 + [Sum\ of\ Diagonal\ Minors]\lambda - |A| = 0$$

$$A = \begin{bmatrix} -3 & -2 \\ -4 & 1 \end{bmatrix}$$

Eigen Value

-11

$$\lambda^3 - [Sum\ of\ Diagonal\ Elements]\lambda^2 + [Sum\ of\ Diagonal\ Minors]\lambda - |A| = 0$$

$$A = \begin{bmatrix} 8 & -2 \\ 3 & 1 \end{bmatrix}$$

Eigen Value

-11+14

$$\lambda^3 - [Sum\ of\ Diagonal\ Elements]\lambda^2 + [Sum\ of\ Diagonal\ Minors]\lambda - |A| = 0$$

$$A = \begin{bmatrix} 8 & -8 \\ 4 & -3 \end{bmatrix}$$

Eigen Value

-11+14+8=11

$$\lambda^3 - [Sum\ of\ Diagonal\ Elements]\lambda^2 + [Sum\ of\ Diagonal\ Minors]\lambda - |A| = 0$$

$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$

Eigen Value

$$\lambda^3 - [Sum\ of\ Diagonal\ Elements]\lambda^2 + [Sum\ of\ Diagonal\ Minors]\lambda - |A| = 0$$

$$\lambda^3 - 6\lambda^2 + 11\lambda - 6 = 0$$

$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$

Eigen Value

$$\boldsymbol{\lambda = 1, 2, 3}$$

$$\boldsymbol{\lambda^3 - [Sum\ of\ Diagonal\ Elements]\lambda^2 + [Sum\ of\ Diagonal\ Minors]\lambda - |A| = 0}$$

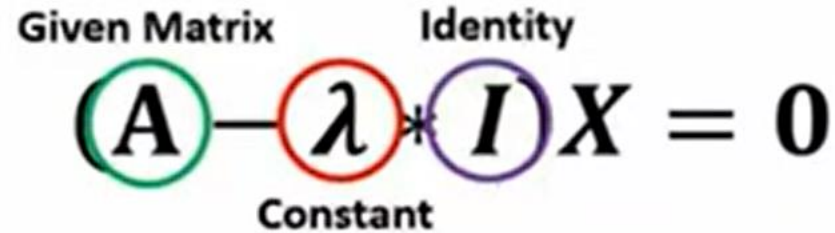
$$\boldsymbol{\lambda^3 - 6\lambda^2 + 11\lambda - 6 = 0}$$

Eigen vectors

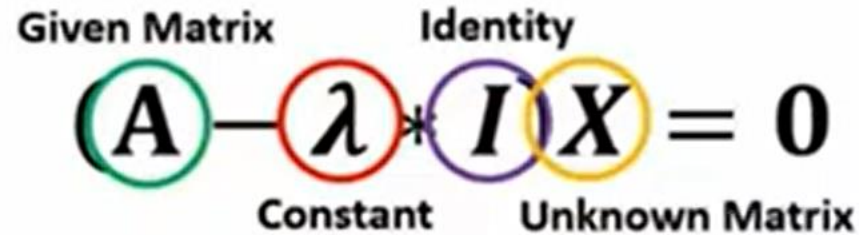
$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$

$$(\mathbf{A} - \boldsymbol{\lambda} * \boldsymbol{I})\boldsymbol{X} = \mathbf{0}$$

$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$



$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$



$$A = \begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix}$$

$$\left(\begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\left(\begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\left(\begin{bmatrix} 8 & -8 & -2 \\ 4 & -3 & -2 \\ 3 & -4 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 8-\lambda & -8 & -2 \\ 4 & -3-\lambda & -2 \\ 3 & -4 & 1-\lambda \end{bmatrix} = 0$$

$$\lambda = 1$$

$$\begin{bmatrix} 8-\lambda & -8 & -2 \\ 4 & -3-\lambda & -2 \\ 3 & -4 & 1-\lambda \end{bmatrix} = 0$$

$$\begin{bmatrix} 7 & -8 & -2 \\ 4 & -4 & -2 \\ 3 & -4 & 0 \end{bmatrix} * \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$\lambda = 1$

$$\begin{bmatrix} 8-\lambda & -8 & -2 \\ 4 & -3-\lambda & -2 \\ 3 & -4 & 1-\lambda \end{bmatrix} = 0$$

$$\begin{bmatrix} 7 & -8 & -2 \\ 4 & -4 & -2 \\ 3 & -4 & 0 \end{bmatrix} * \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Crammer's Rule

$7(x1) - 8(x2) - 2(x3) = 0$

$4(x1) - 4(x2) - 2(x3) = 0$

$\lambda = 1$

$$7(x1) - 8(x2) - 2(x3) = 0$$
$$4(x1) - 4(x2) - 2(x3) = 0$$

$$\frac{x1}{\begin{vmatrix} -8 & -2 \\ -4 & -2 \end{vmatrix}} = \frac{x2}{\begin{vmatrix} 7 & -2 \\ -4 & -2 \end{vmatrix}} = \frac{x3}{\begin{vmatrix} 7 & -8 \\ 4 & -4 \end{vmatrix}}$$

$\lambda = 1$

$$7(x1) - 8(x2) - 2(x3) = 0$$
$$4(x1) - 4(x2) - 2(x3) = 0$$

$$\frac{x1}{8} = \frac{x2}{6} = \frac{x3}{4}$$

$$\frac{x1}{\begin{vmatrix} -8 & -2 \\ -4 & -2 \end{vmatrix}} = \frac{x2}{\begin{vmatrix} 7 & -2 \\ -4 & -2 \end{vmatrix}} = \frac{x3}{\begin{vmatrix} 7 & -8 \\ 4 & -4 \end{vmatrix}}$$

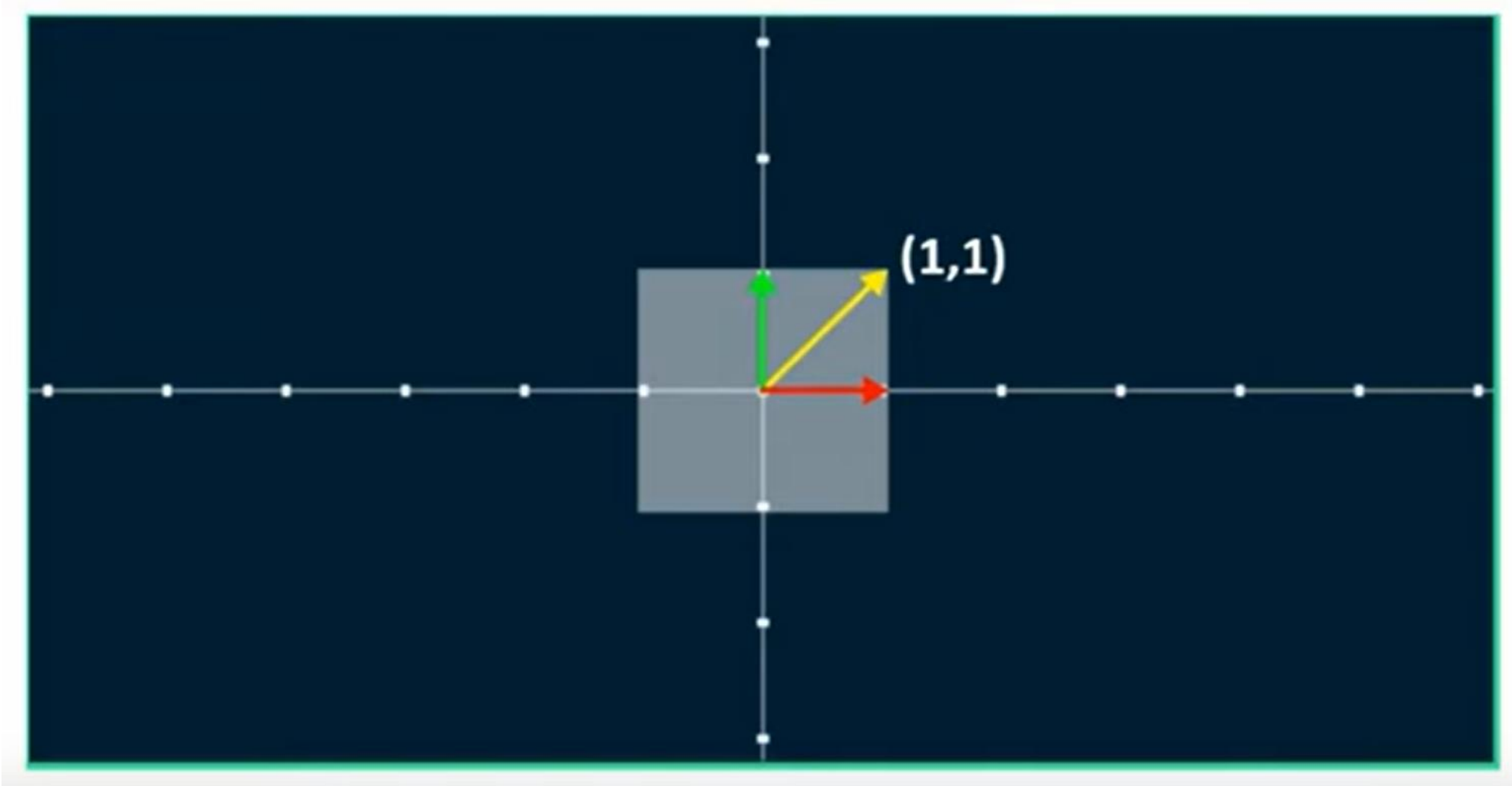
$$\lambda = 1$$

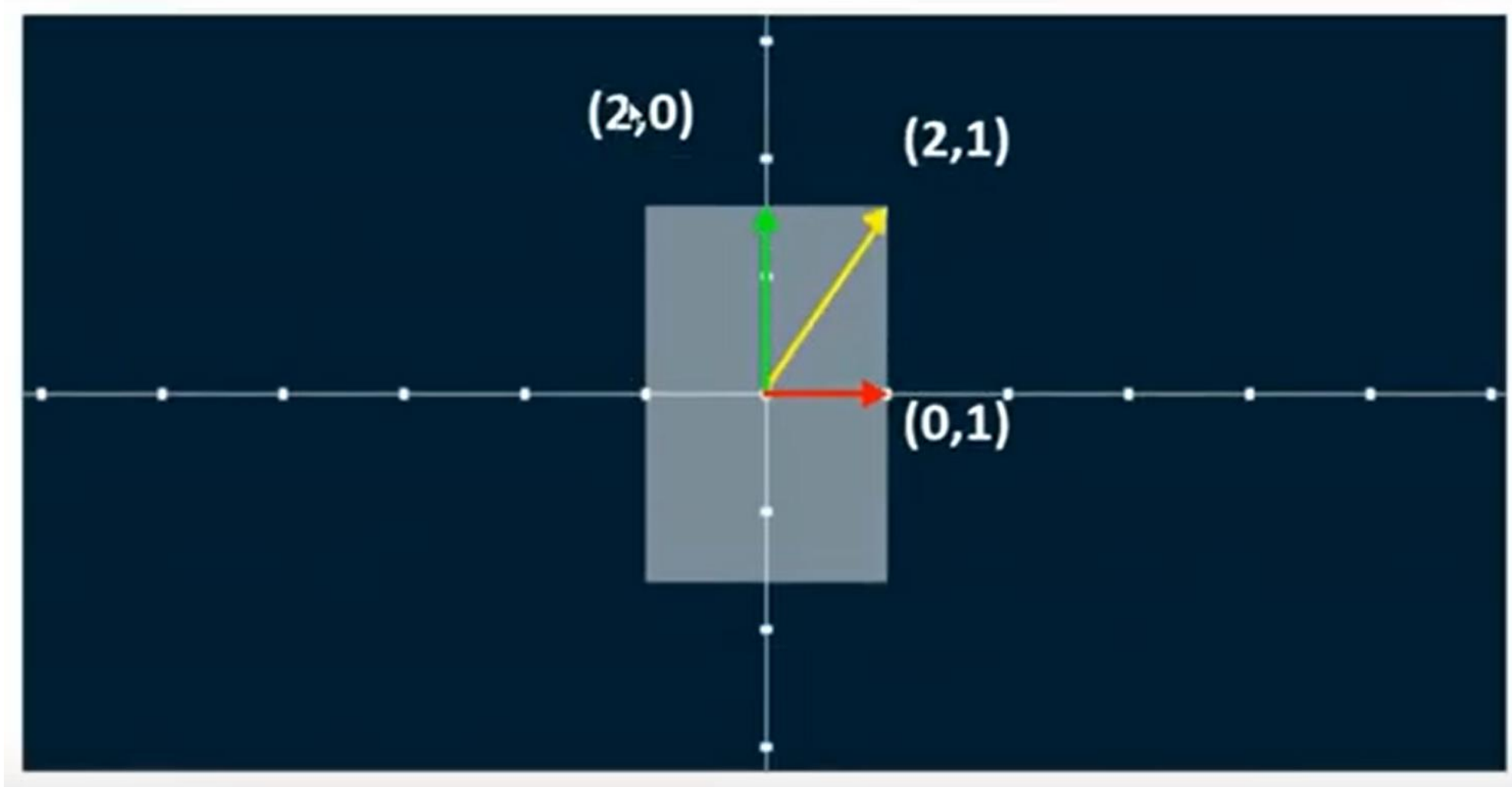
$$7(x1) - 8(x2) - 2(x3) = 0$$
$$4(x1) - 4(x2) - 2(x3) = 0$$

$$\frac{x1}{8} = \frac{x2}{6} = \frac{x3}{4}$$

$$\frac{x1}{\begin{vmatrix} -8 & -2 \\ -4 & -2 \end{vmatrix}} = \frac{x2}{\begin{vmatrix} 7 & -2 \\ -4 & -2 \end{vmatrix}} = \frac{x3}{\begin{vmatrix} 7 & -8 \\ 4 & -4 \end{vmatrix}}$$

$$Eigen\ Vector = \begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix}$$


(1,1)


(2,0)
(2,1)
(0,1)

PCA: algorithmic summary

Generation of data matrix, A

Data matrix, ***A***, has 1700 rows(different molten salts) and 7 columns(properties).
The properties in this example includes
1) equivalent weight 2) melting point 3) temperature of the measurements
4) equivalent conductance 5) specific conductance 6) density 7) viscosity

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Scaling (normalization) of the data matrix, X

X is a scaled matrix of ***A.***
Matrix ***X*** in the left is an example of "Unit Variance" scaling.
Each s_{ij} represents standard deviation.

$$X = \begin{bmatrix} (a_{11} / s_{k1}) & \cdots & (a_{1n} / s_{kn}) \\ \vdots & \ddots & \vdots \\ (a_{m1} / s_{k1)} & \cdots & (a_{mn} / s_{kn}) \end{bmatrix}$$

Covariance matrix of the scaled data matrix, S

S is a covariance matrix of ***X.***

$$S = \mathrm{cov}(X) = \frac{X^T X}{m-1}$$

Eigenvalue decomposition of covariance matrix

P is called as loading (or eigenvector) matrix.
Λ is a eigenvalue matrix (eigenvalues on the diagonal of this diagonal matrix).

$$S = P \Lambda P^{-1} \quad \text{or} \quad \mathrm{cov}(X)\, p_i = \lambda_i p_i$$

Calculation of scores from the loadings

t_i: scores (orthogonal), p_i: loadings (orthonormal)

$$X = t_1 p_1^T + t_1 p_1^T + \cdots + t_k p_k^T + E$$
$$\text{where} \quad k \le \min\{m, n\}$$

Step by step PCA(ref. PCA step by step.ipynb in Jupyter notebook)

- Step-1 Define data

jupyter PCA Step by Step Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [42]:

```
#importing packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy.linalg import eig
```

In [43]:

```
# defining a simple data
Marks = np.array([[3,4], [2,8], [6,9]])
print(Marks)
```

```
[[3 4]
 [2 8]
 [6 9]]
```

jupyter PCA Step by Step Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

```
[[3 4]
 [2 8]
 [6 9]]
```

In [44]:

```
1 Marks_df =pd.DataFrame(Marks,columns=["Physics","Maths"])
2 Marks_df
```

Out[44]:

	Physics	Maths
0	3	4
1	2	8
2	6	9

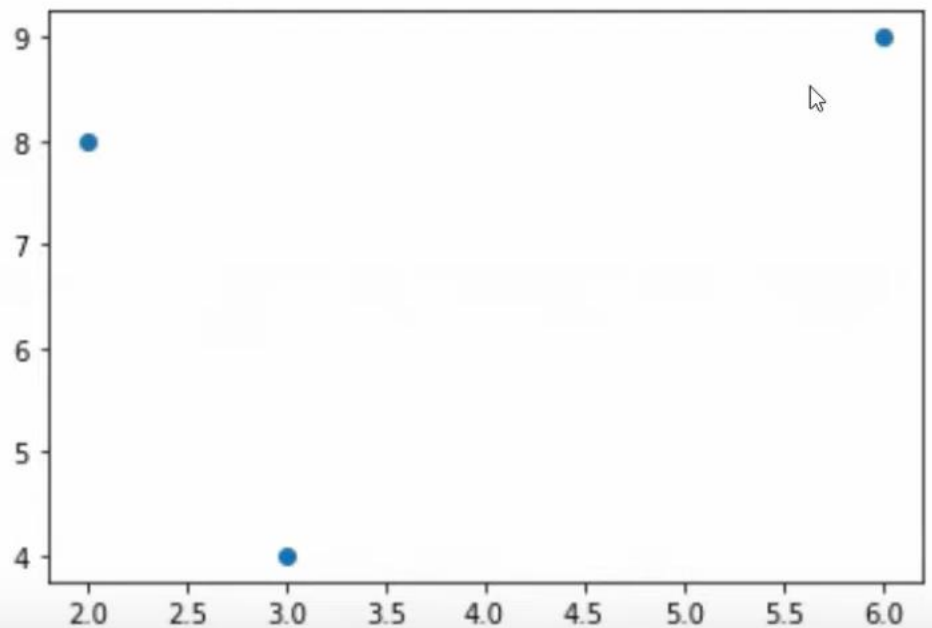
jupyter PCA Step by Step Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trust

Run Code

In [56]:
```
1 plt.scatter(Marks_df["Physics"],Marks_df["Maths"])
```

Out[56]: <matplotlib.collections.PathCollection at 0x21de997a250>



- Step2-Make data mean centered(Scaling)

In [46]:

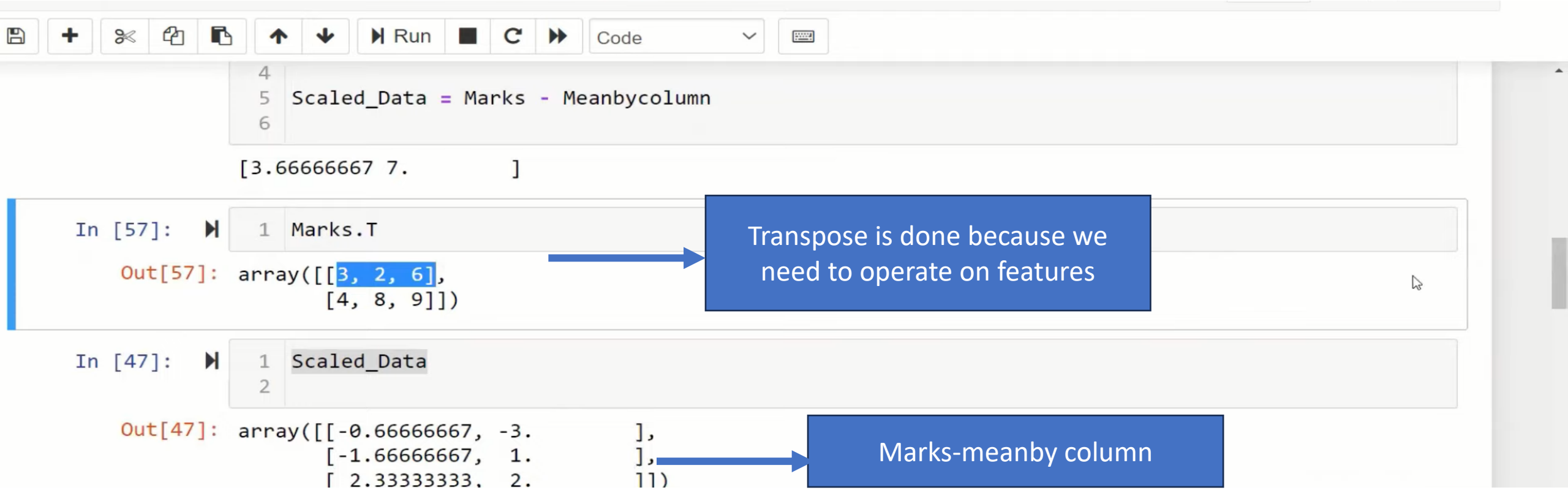
```
#makind data mean centric
Meanbycolumn = np.mean(Marks.T, axis=1)
print(Meanbycolumn)

Scaled_Data = Marks - Meanbycolumn

```

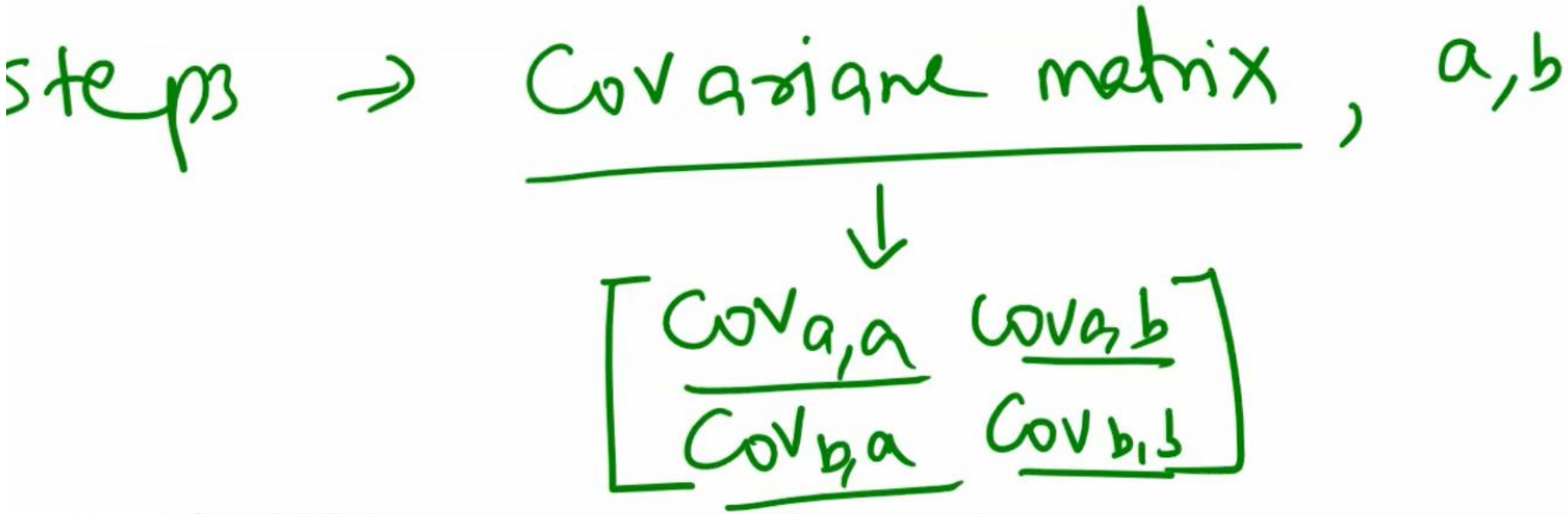
```
[3.66666667 7.        ]
```

Why scaling is required?:to make each feature uniform



Step-3

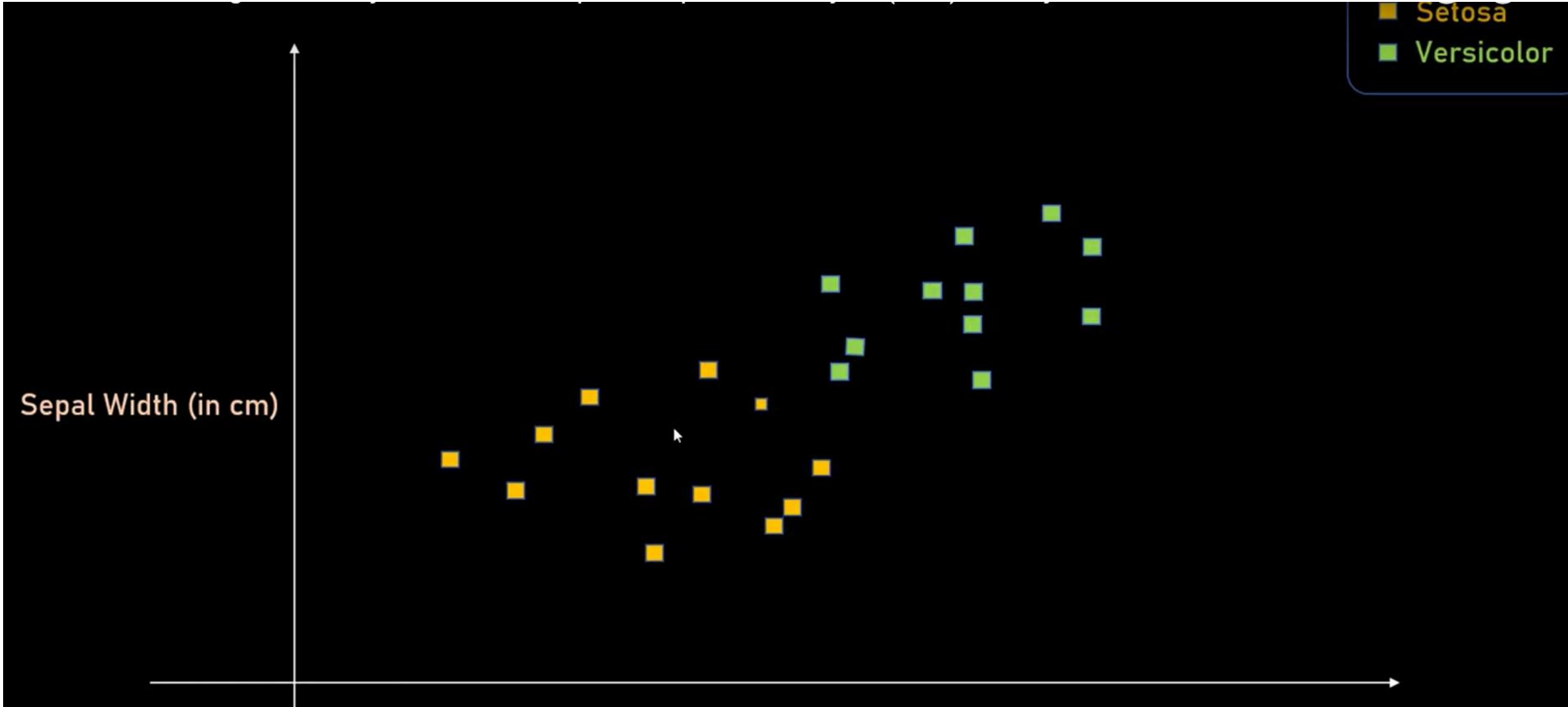
- Find the covariance matrix

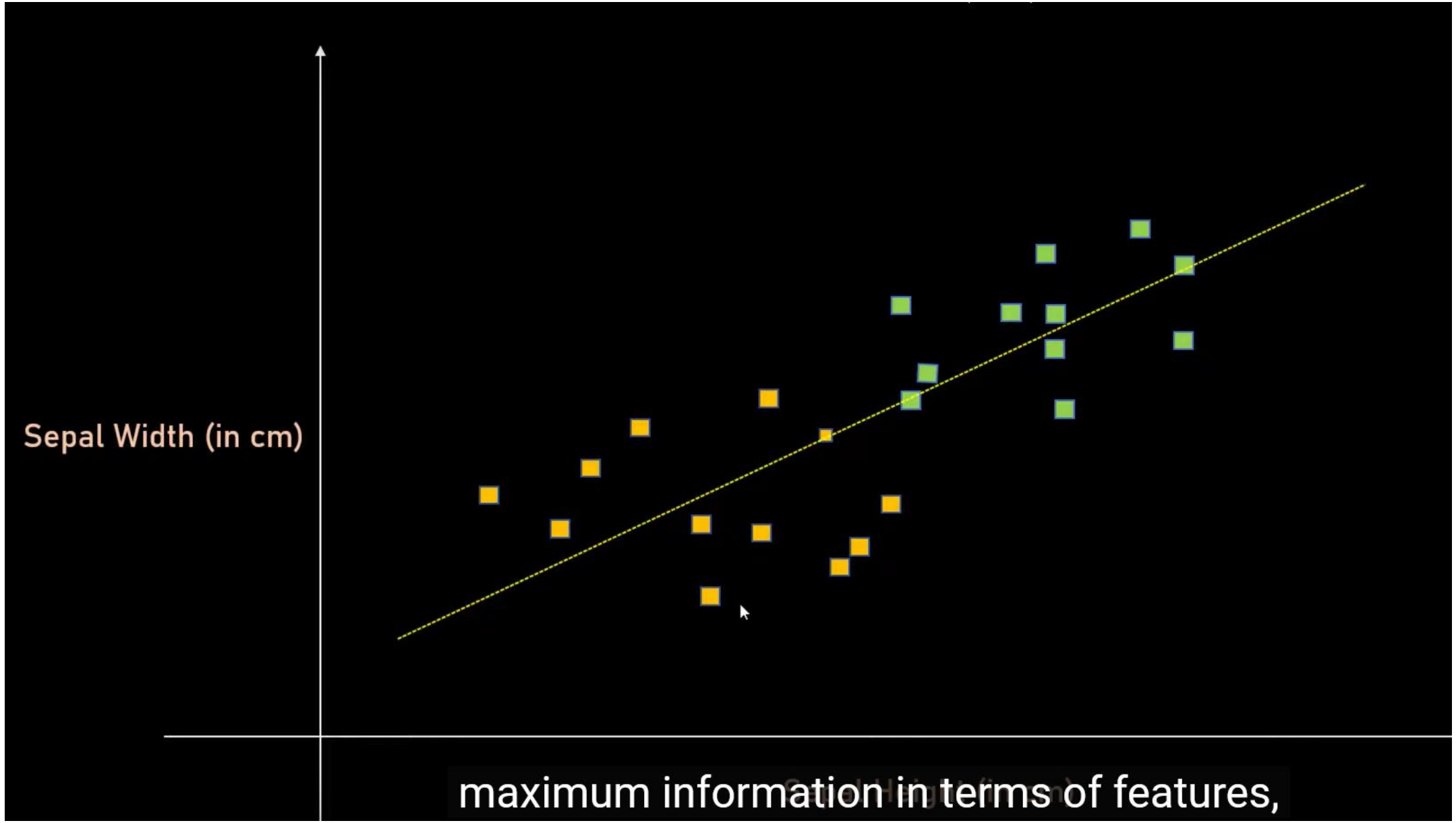


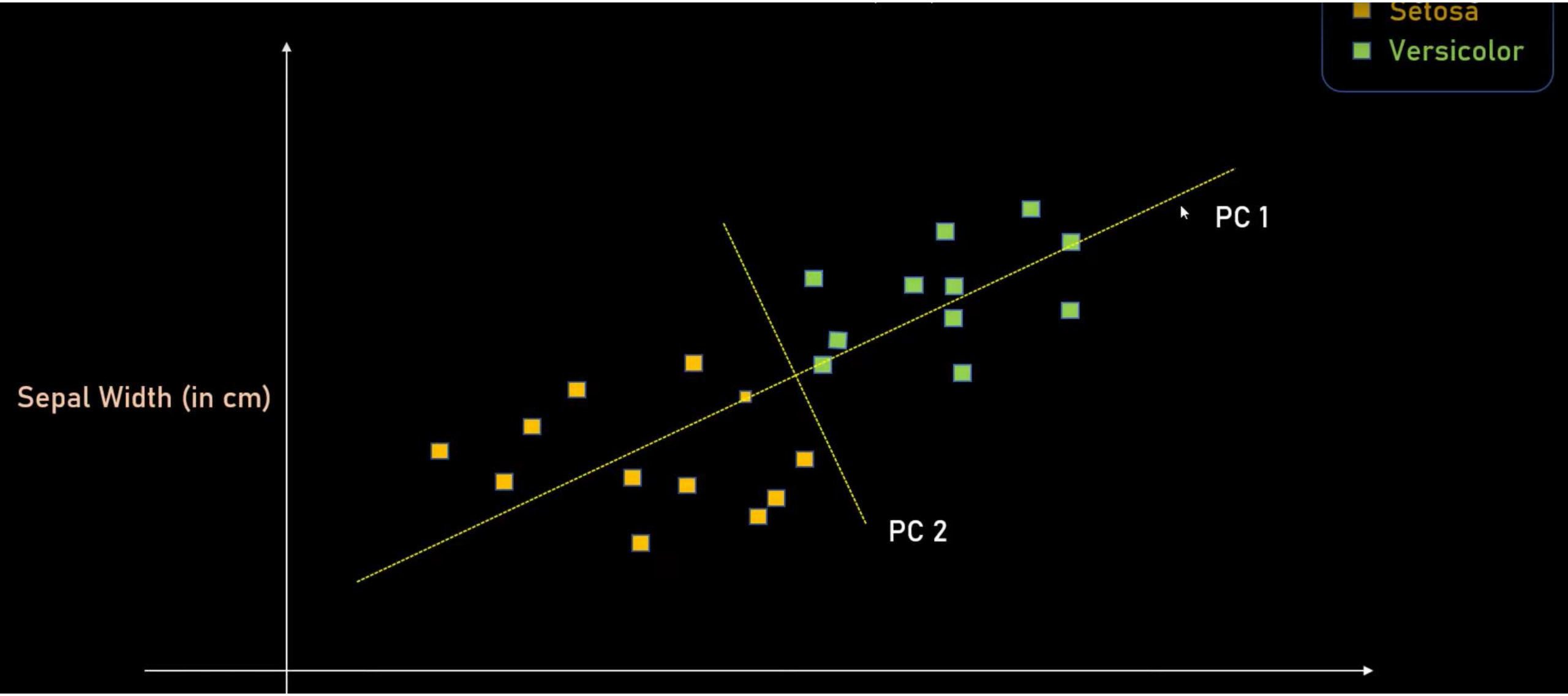
In [60]:
```
#Find covariance matrix of above scaled data
Cov_mat = np.cov(Scaled_Data.T)
Cov_mat
```

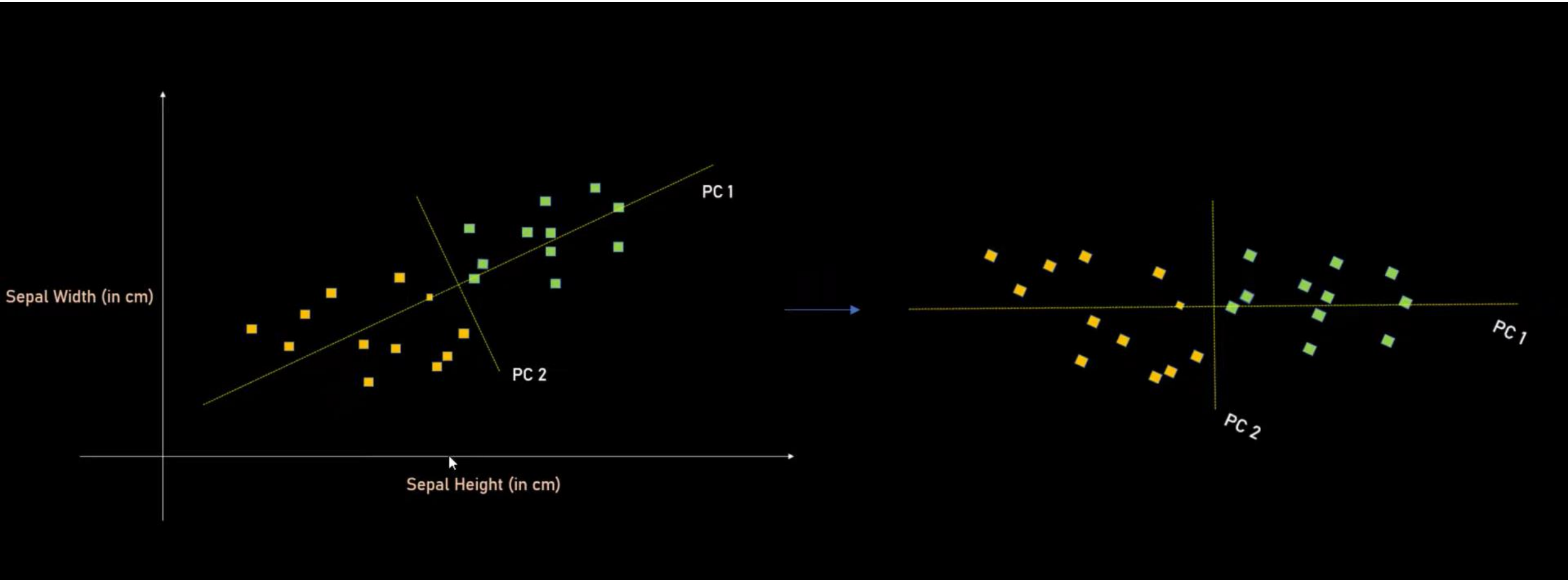
Out[60]:
```
array([[4.33333333, 2.5       ],
       [2.5       , 7.        ]])
```

Why do you need for axis change




Sepal Width (in cm)
maximum information in terms of features,


Setosa
Versicolor
PC 1
PC 2
Sepal Width (in cm)


PC 1
PC 2
Sepal Width (in cm)
Sepal Height (in cm)
PC 1
PC 2

Application: Image compression

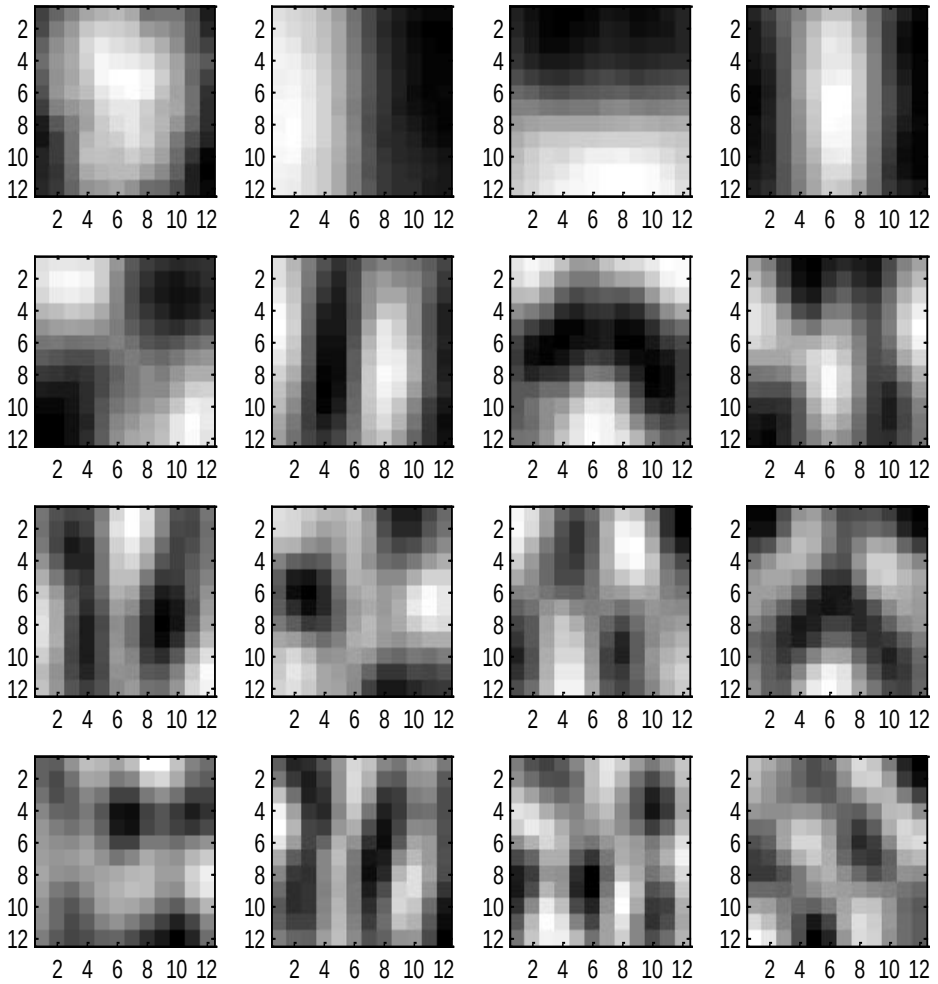
Original Image

- Divide the original 372x492 image into patches:
  - Each patch is an instance that contains 12x12 pixels on a grid
- View each as a 144-D vector

PCA compression: 144D → 60D

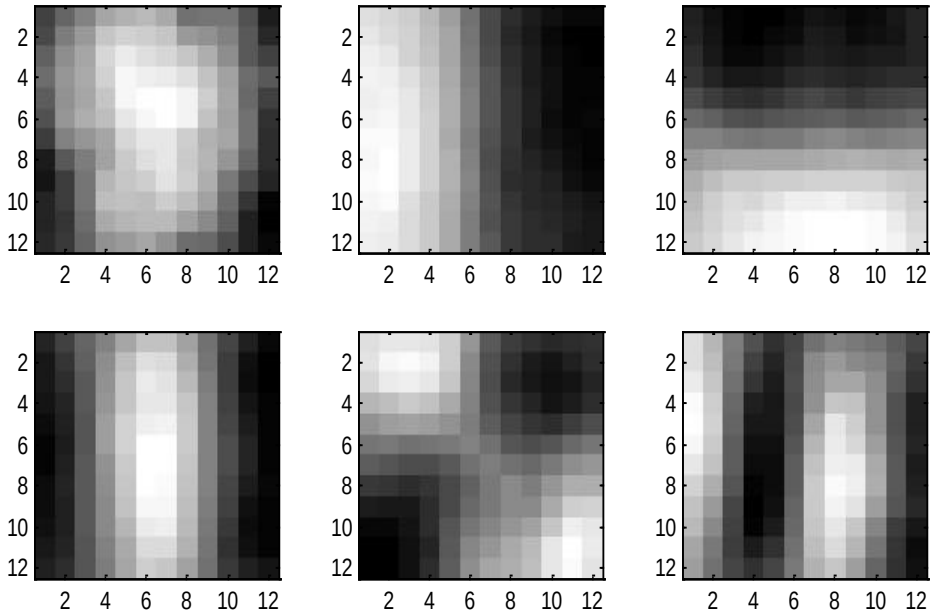
PCA compression: 144D → 16D

16 most important eigenvectors

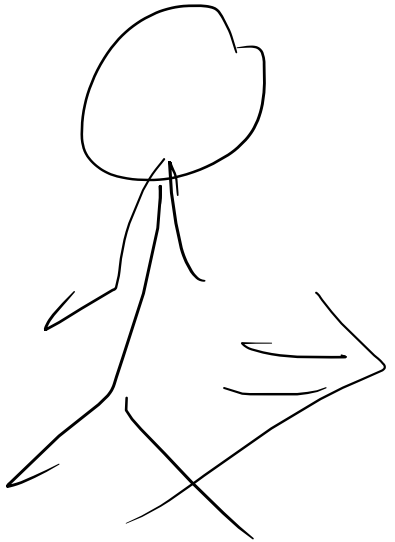


PCA compression: 144D ⟩ 6D

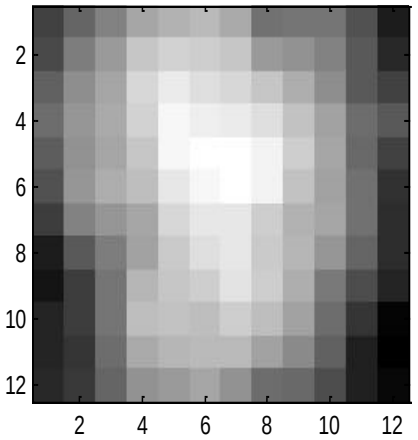
6 most important eigenvectors

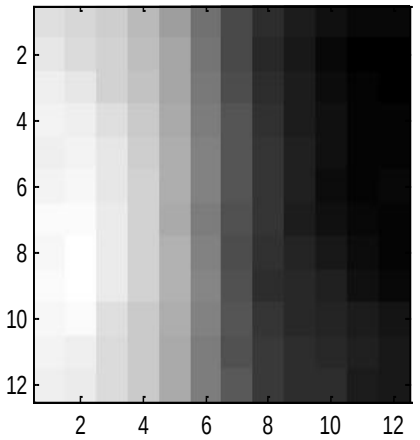


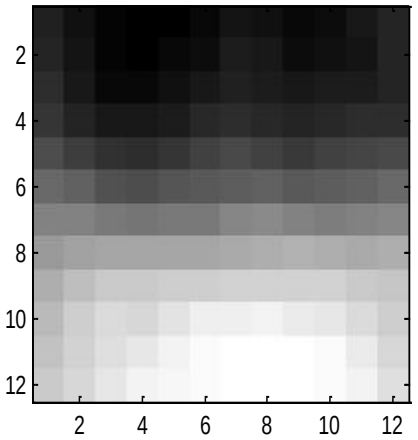
PCA compression:
144D → 3D



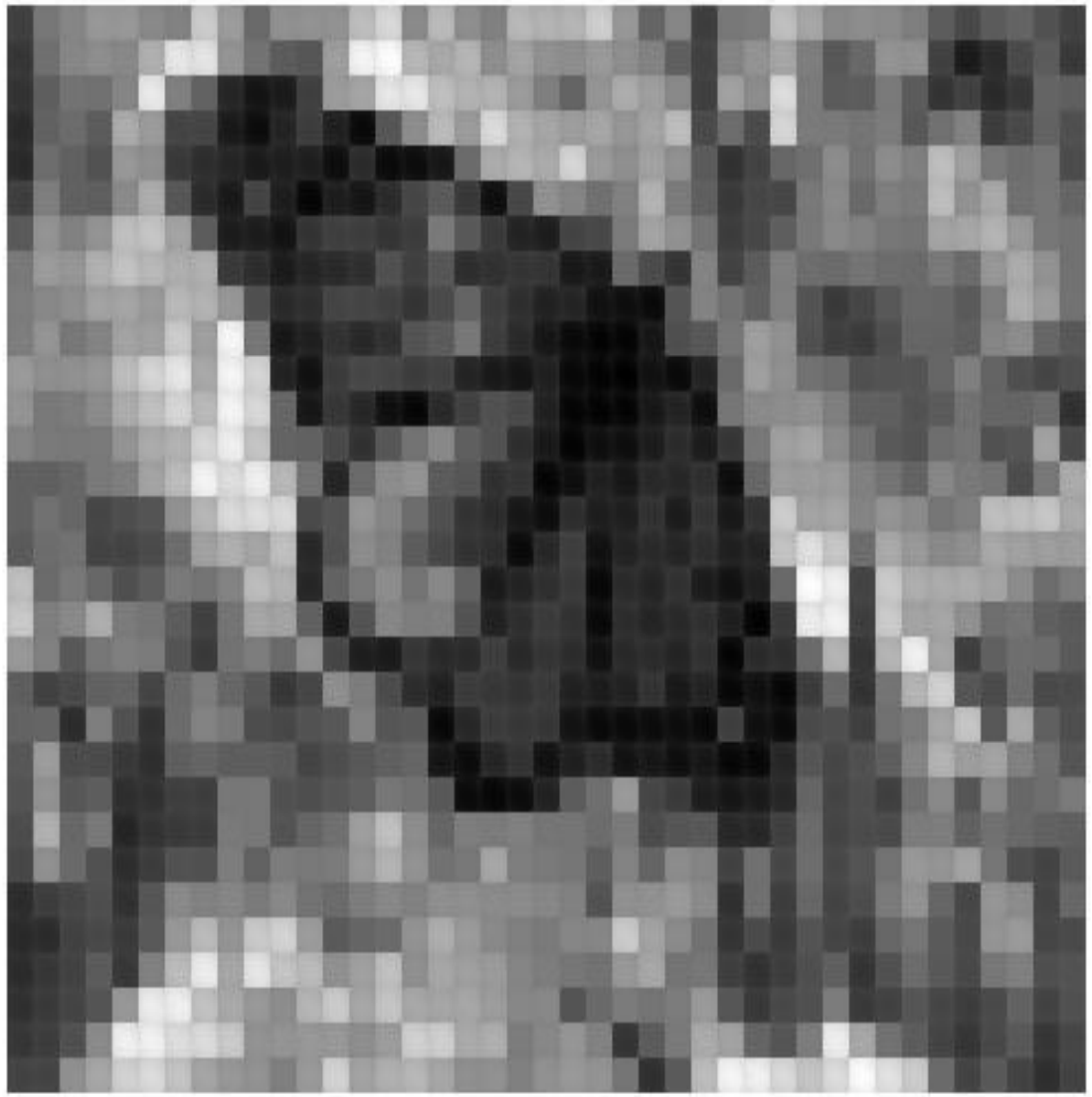
3 most important eigenvectors



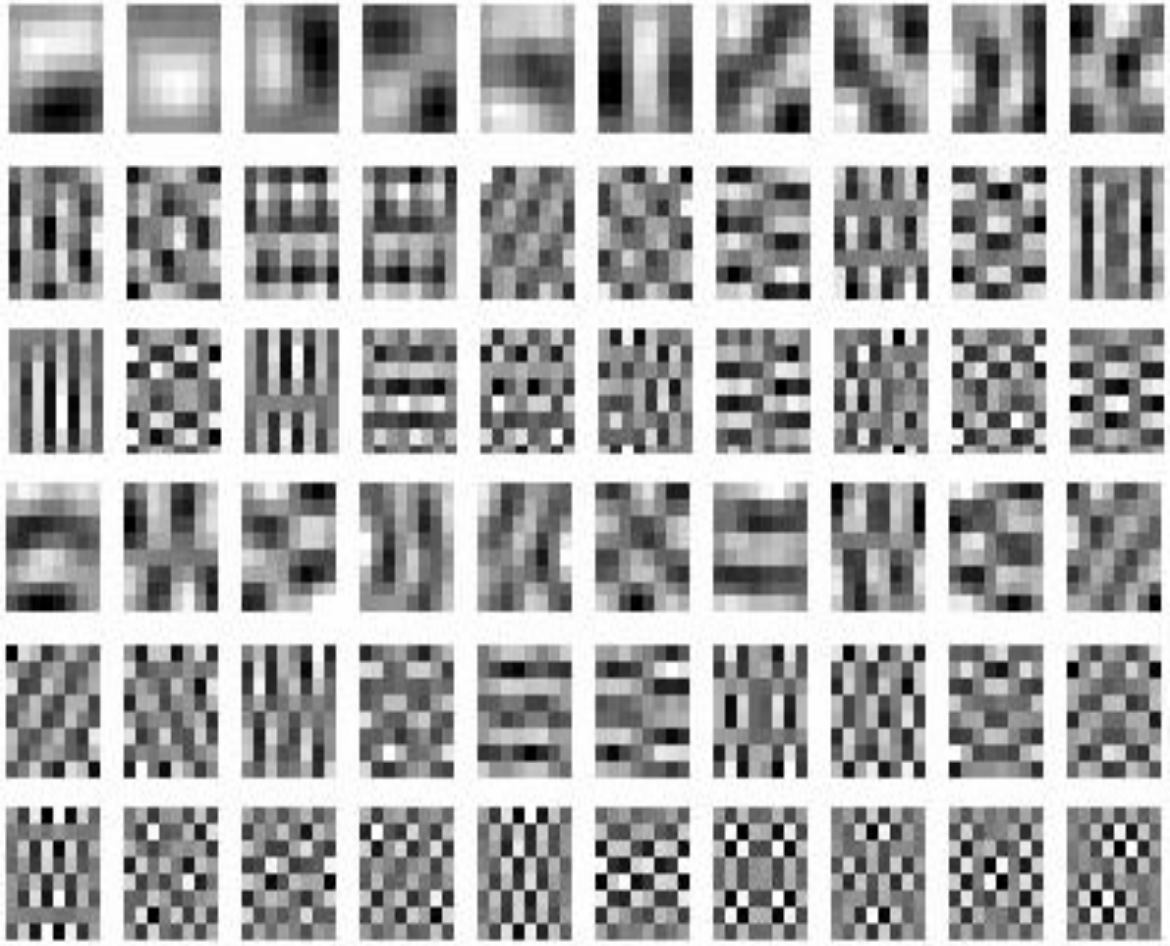




PCA compression:
144D → 1D



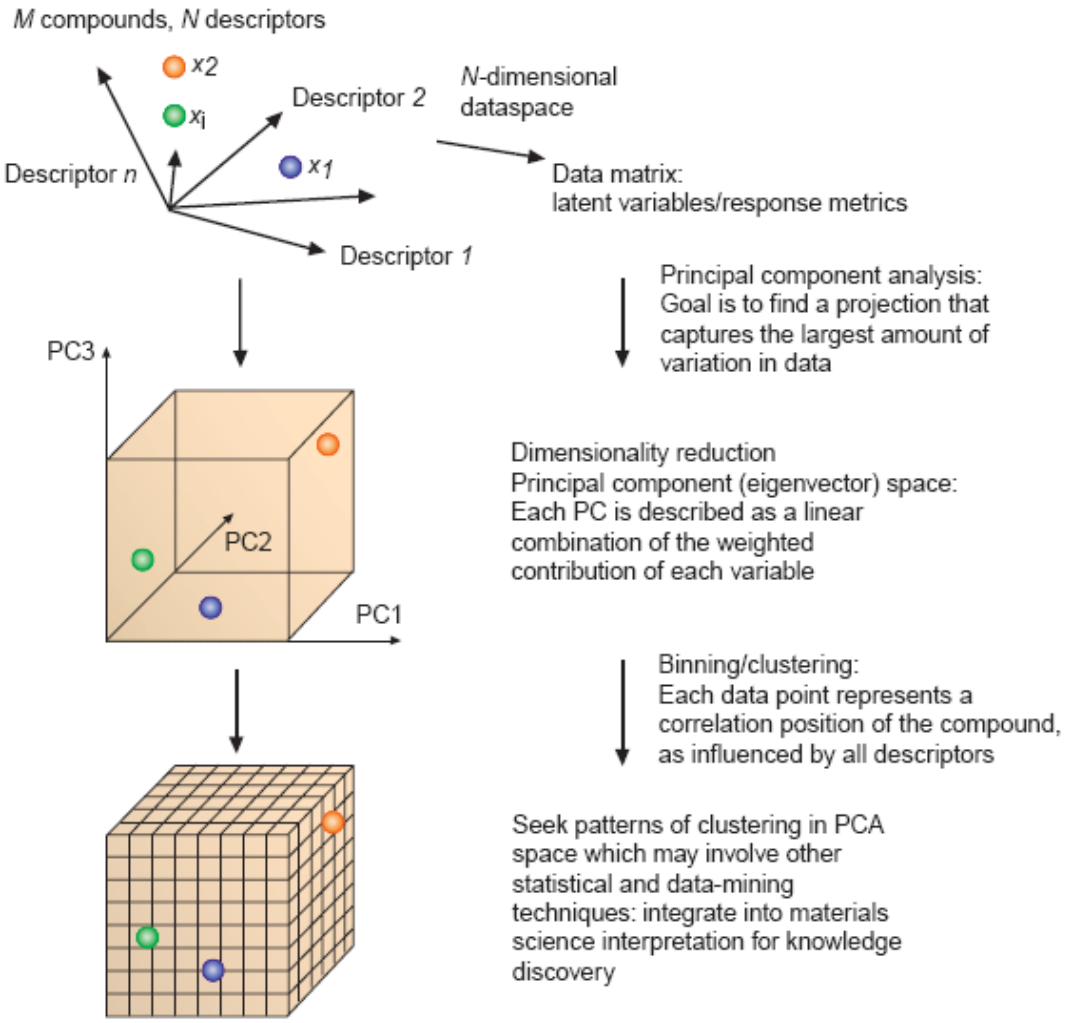
60 most important eigenvectors



Looks like the discrete cosine bases of JPG!...

PRINCIPAL COMPONENT ANALYSIS: PCA

From a set of N correlated descriptors, we can derive a set of N uncorrelated descriptors (the principal components). Each principal component (PC) is a suitable linear combination of all the original descriptors. PCA reduces the information dimensionality that is often needed from the vast arrays of data in a way so that there is minimal loss of information



(from *Nature Reviews Drug Discovery* **1**, 882-894 (2002) : INTEGRATION OF VIRTUAL AND HIGH THROUGHPUT SCREENING Jürgen Bajorath ; and *Materials Today;* MATERIALS INFORMATICS , K. Rajan , October 2005

Singular Value Decomposition – definition

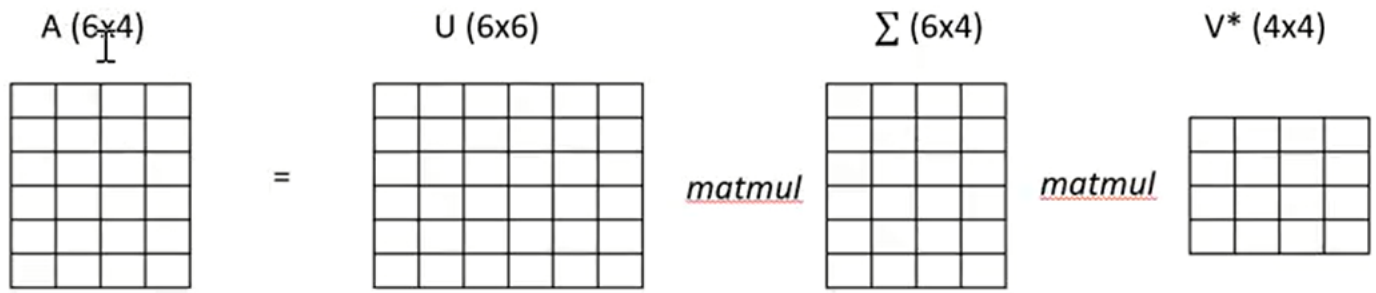
$$A = U \sum V^*$$

A – original matrix (dimensions: (m rows,n columns – m,n))
U – unitary matrix
$\sum$ - rectangular diagonal matrix of singular values
V^* - unitary matrix

A prerequisite:

$$m \geq n$$

Visualizing SVD



Visualizing $\sum$

σ^1	0	0	0	0
0	σ^2	0	0	0
0	0	σ^3	0	0
0	0	0	...	
0	0	0	0	σ^n

σ^i $-$ *ith singular value*

$\sigma^1 \geq \sigma^2 \geq \cdots \geq \sigma^n$

SVD – Singular Value Decomposition

X						U					Σ					V^T				
1	0	0	0	2		0	0	1	0		4	0	0	0		0	1	0	0	0
0	0	3	0	0	=	0	1	0	0	x	0	3	0	0	x	0	0	1	0	0
0	0	0	0	0		0	0	0	-1		0	0	2.24	0		0.45	0	0	0	0.89
0	4	0	0	0		1	0	0	0		0	0	0	0		0	0	0	1	0

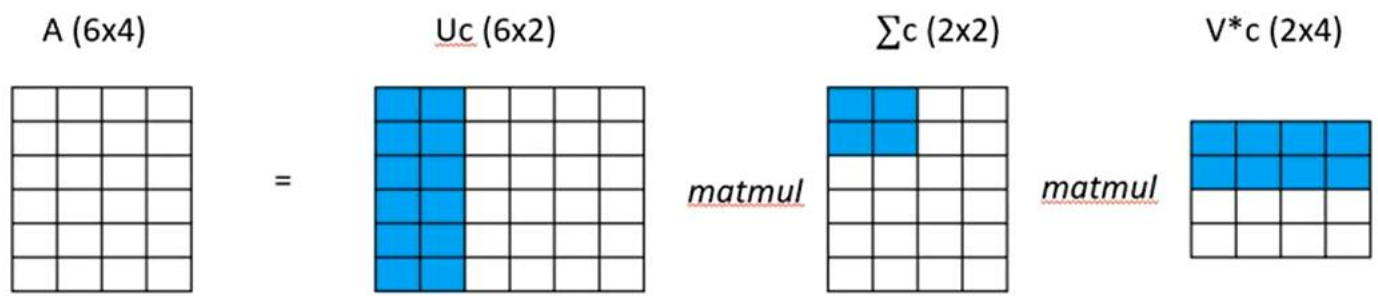
*Original Size = 4 * 5 = 20 bytes*

k = 1

0	x	4	x	0	1	0	0	0	=	0	0	0	0	0
0										0	0	0	0	0
0										0	0	0	0	0
1										0	4	0	0	0

Compressed Size = 4*1 + 1 + 1*5 = 10 bytes

Compressing with SVD



For k = 2

k – number of singular values we want to use for compression

SVD – Singular Value Decomposition

k = 2

0	0	x	4	0	x	0	1	0	0	0	=	0	0	0	0	0
0	1		0	3		0	0	1	0	0		0	0	3	0	0
0	0											0	0	0	0	0
1	0											0	4	0	0	0

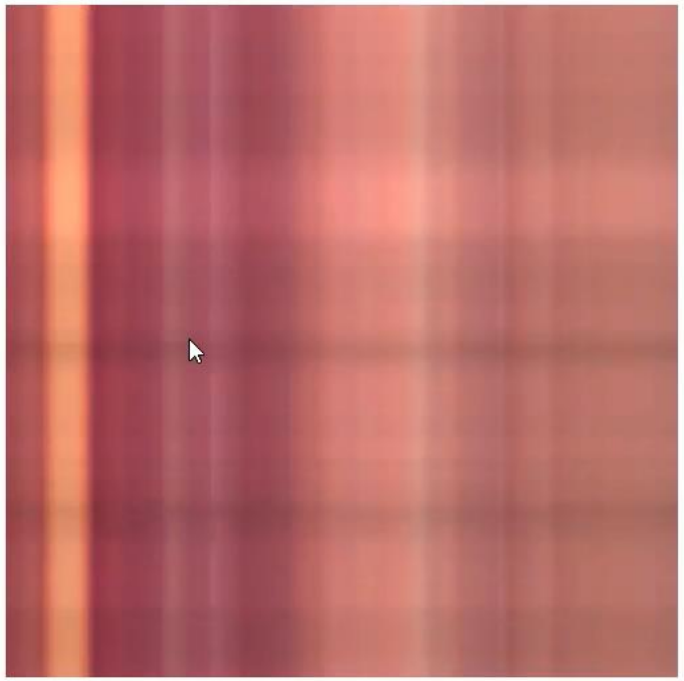
*Compressed Size = 4*2 + 2 + 2*5 = 20 bytes*

- Original size = 384*384 bytes = 147,456 bytes
- k = 1 : 384*1 + 1 + 1*384 = 769 bytes
- k = 10 : 384*10 + 10 + 10*384 = 7,690 bytes
- k = 20 : 384*20 + 20 + 20*384 = 15,380 bytes
- k = 50 : 384*50 + 50 + 50*384 = 38,450 bytes
- k = 100 : 384*100 + 100 + 100*384 = 76,900 bytes
- k = 200 : 384*200 + 200 + 200*384 = 153,800 bytes

http://journal.batard.info/post/2009/04/08/svd-fun-profit

Let's try on real data. If we load an image as a matrix of colours, how well can we compress it using the SVD? Pretty well, even if it's not JPEG. Take a look at "Lenna", using different values for k:

$k = 1$, image generated from $(1 \times 384 + 1 + 384 \times 1) = 769$ bytes.

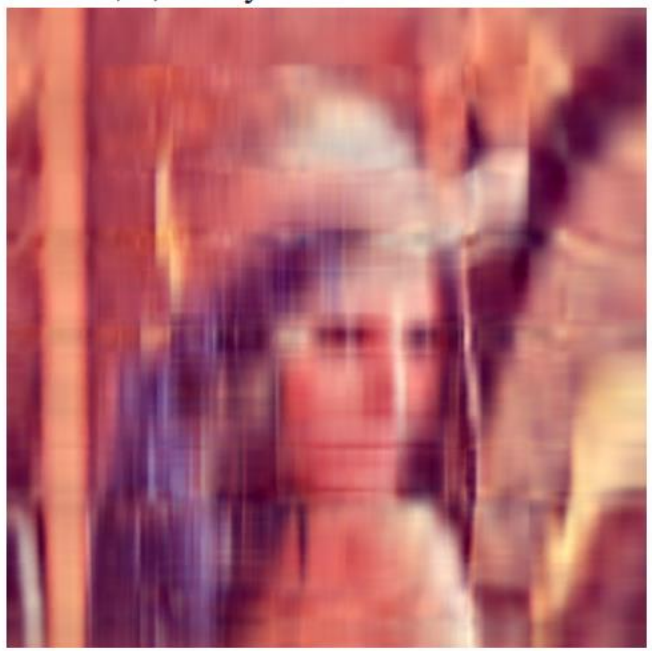


$k = 10$; 7,690 bytes.

The image size=147kb

If k=1 then size is 769 bytes

$k = 10$; 7,690 bytes.



Size is 7.69 kb

$k = 20$; 15,380 bytes.

Size is 15.30 kb