Principal Component Analysis
Dimensionality Reduction

Dr. Dipak Kumar Mohanty
KIIT University, Bhubaneswar

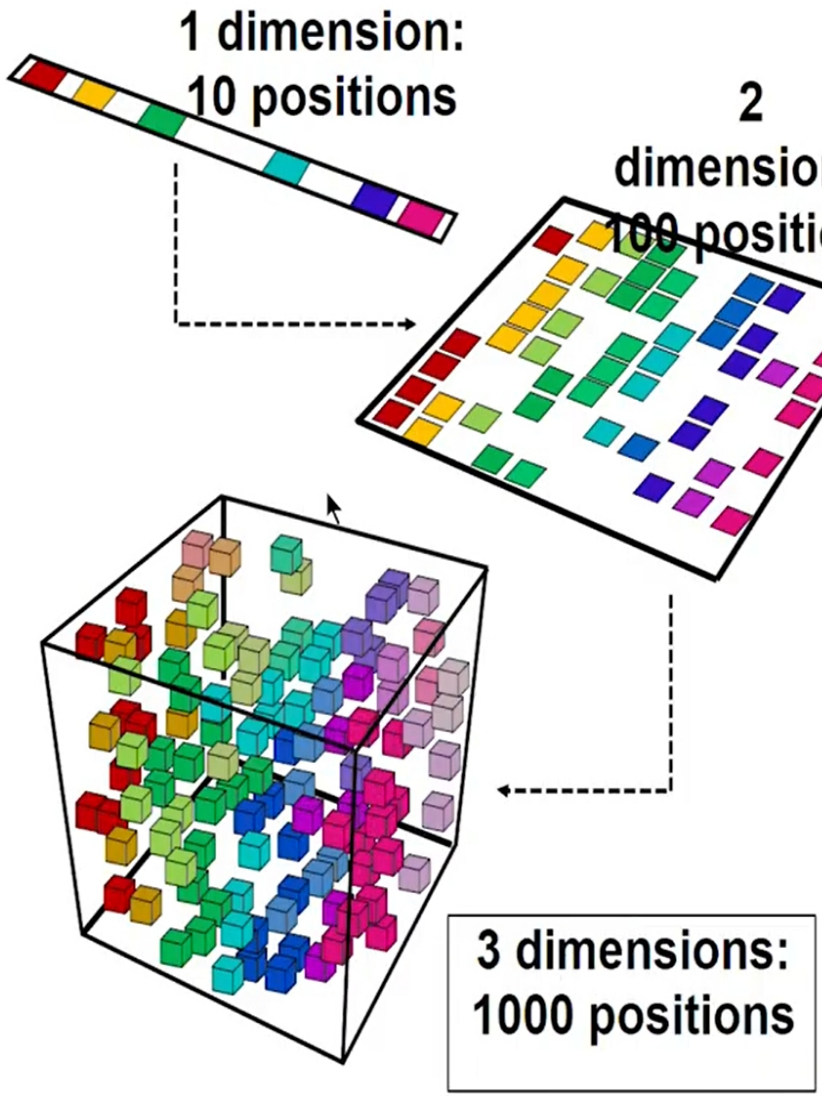
Dimensions So Far…

- 1 cell = 1-D graph (number line)
- 2 cells = 2-D graph (normal x/y graph)
- 3 cells = 3-D graph (fancy graph with depth)
- 4 cells = 4-D graph (you can't draw it)
- 200 cells = 200-D graph (etc..)

Are all those dimensions super important? Or are some more important than others?

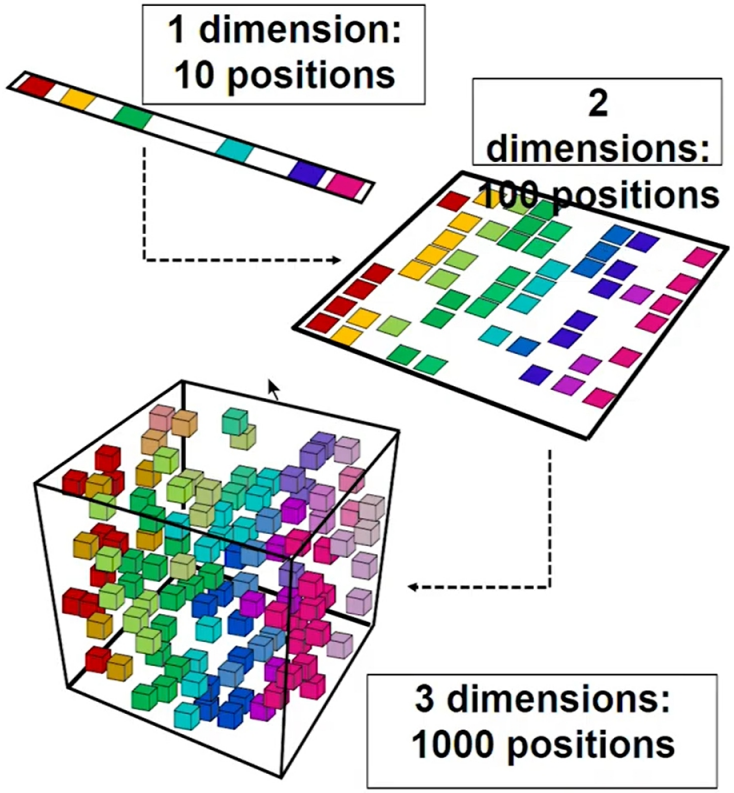
Curse of Dimensionality

- Theoretically, increasing features should improve performance

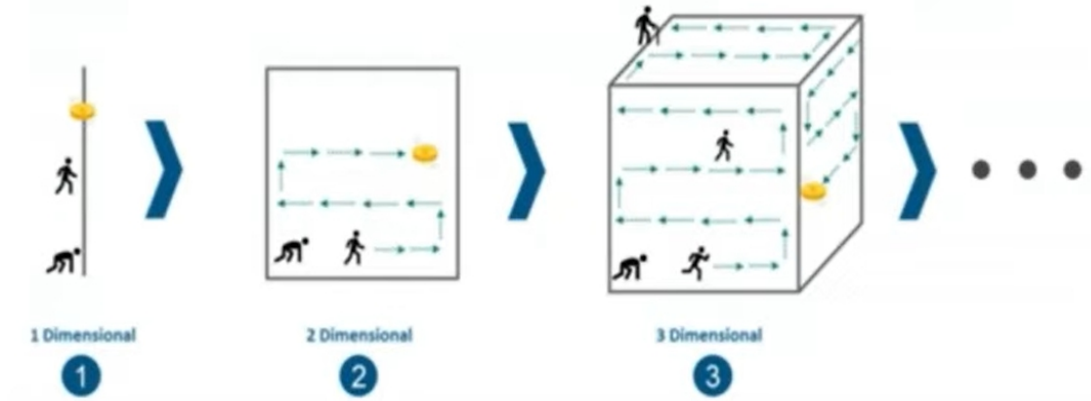


Curse of Dimensionality

- Theoretically, increasing features should improve performance
- In practice, too many features leads to worse performance
- Number of training examples required increases exponentially with dimensionality



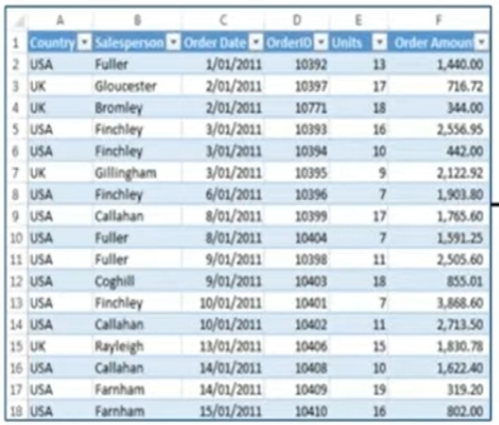
Coin dropping and seaching Scenarios....



Curse of dimensionality

Need For PCA

High dimension data is extremely complex to process due to inconsistencies in the features which increase the computation time and make data processing and EDA more convoluted.



	A	B	C	D	E	F
1	Country	Salesperson	Order Date	OrderID	Units	Order Amount
2	USA	Fuller	1/01/2011	10392	13	1,440.00
3	UK	Gloucester	2/01/2011	10397	17	716.72
4	UK	Bromley	2/01/2011	10771	18	344.00
5	USA	Finchley	3/01/2011	10393	16	2,556.95
6	USA	Finchley	3/01/2011	10394	10	442.00
7	UK	Gillingham	3/01/2011	10395	9	2,122.92
8	USA	Finchley	6/01/2011	10396	7	1,903.80
9	USA	Callahan	8/01/2011	10399	17	1,765.60
10	USA	Fuller	8/01/2011	10404	7	1,591.25
11	USA	Fuller	9/01/2011	10398	11	2,505.60
12	USA	Coghill	9/01/2011	10403	18	855.01
13	USA	Finchley	10/01/2011	10401	7	3,868.60
14	USA	Callahan	10/01/2011	10402	11	2,713.50
15	UK	Rayleigh	13/01/2011	10406	15	1,830.78
16	USA	Callahan	14/01/2011	10408	10	1,622.40
17	USA	Farnham	14/01/2011	10409	19	319.20
18	USA	Farnham	15/01/2011	10410	16	802.00

Original data

Remove inconsistencies

Redundant data

Highly-correlated features

D	E	F
OrderID	Units	Order Amount
10392	13	1,440.00
10397	17	716.72
10771	18	344.00
10393	16	2,556.95
10394	10	442.00
10395	9	2,122.92
10396	7	1,903.80
10399	17	1,765.60
10404	7	1,591.25
10398	11	2,505.60
10403	18	855.01
10401	7	3,868.60
10402	11	2,713.50
10406	15	1,830.78
10408	10	1,622.40
10409	19	319.20
10410	16	802.00

New data space with lesser features that retain most of the info

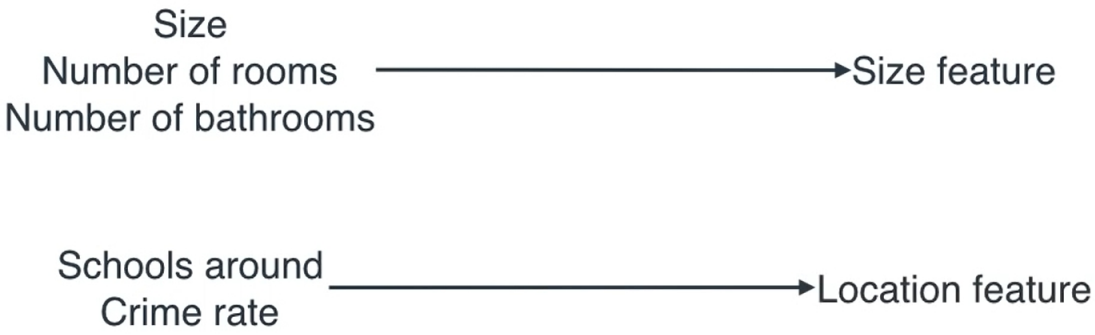
What Is Principal Component Analysis (PCA)?

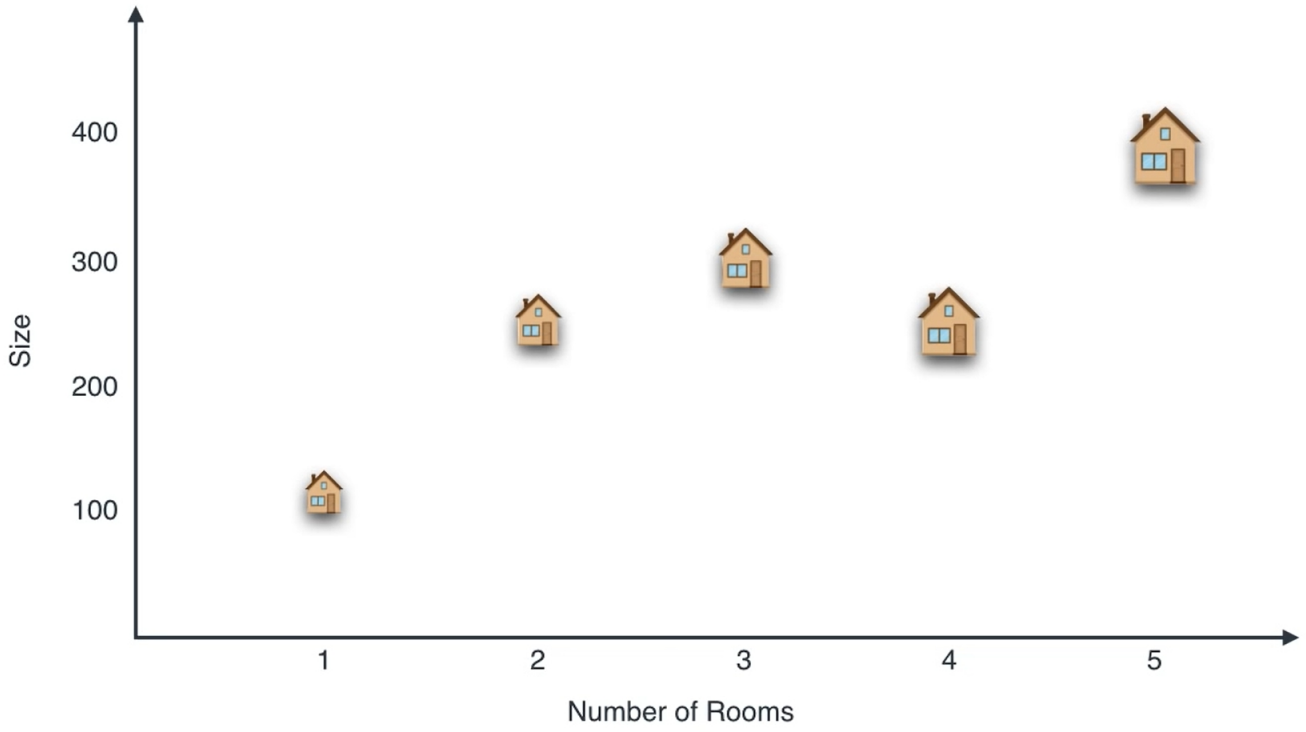
Principal components analysis (PCA) is a dimensionality reduction technique that enables you to identif correlations and patterns in a data set so that it can be transformed into a data set of significantly lowe dimension without loss of any important information.

Housing Data

Size
Number of rooms
Number of bathrooms
Schools around
Crime rate

Housing Data




400
300
200
100
Size
1
2
3
4
5
Number of Rooms

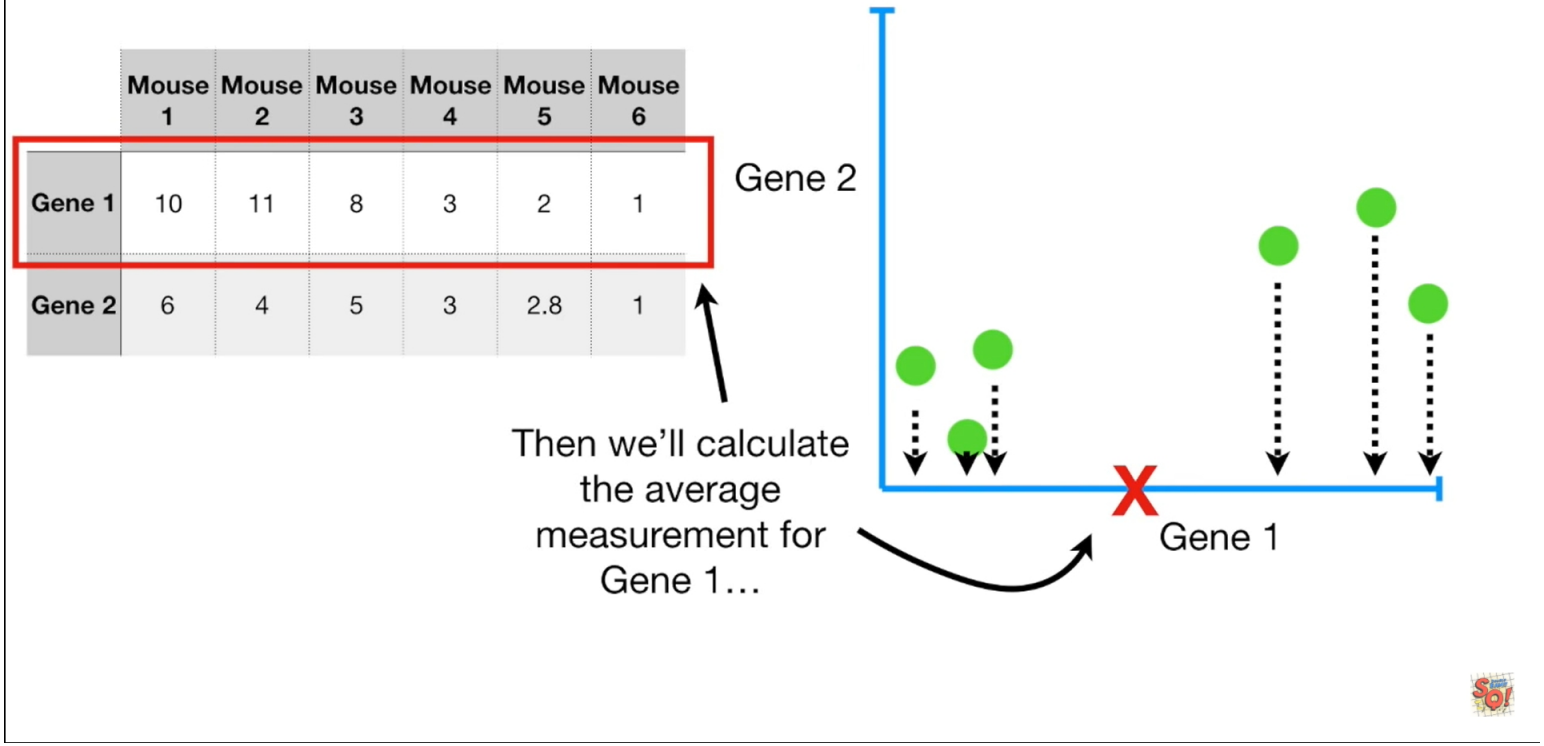
Housing Data

5 dimensions	2 dimensions
Size Number of rooms Number of bathrooms	Size feature
Schools around Crime rate	Location feature

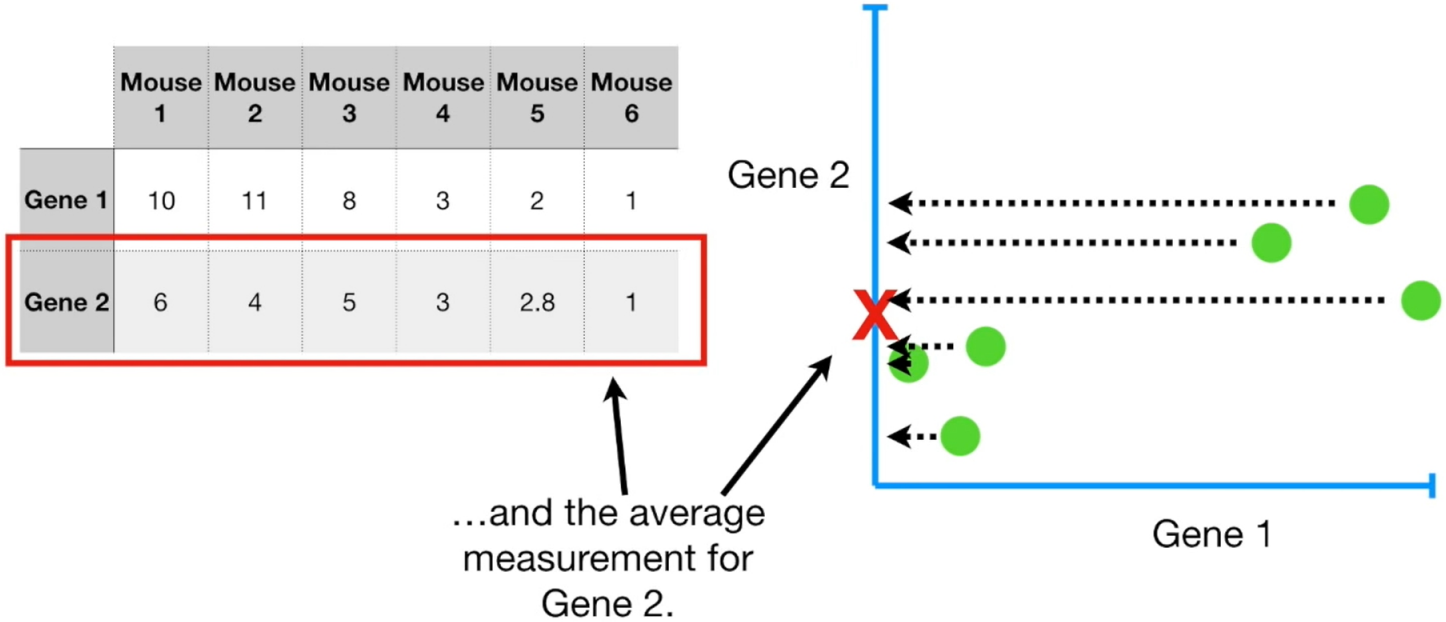
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

To understand what PCA does and how it works, let's go back to the dataset that only had 2 genes…

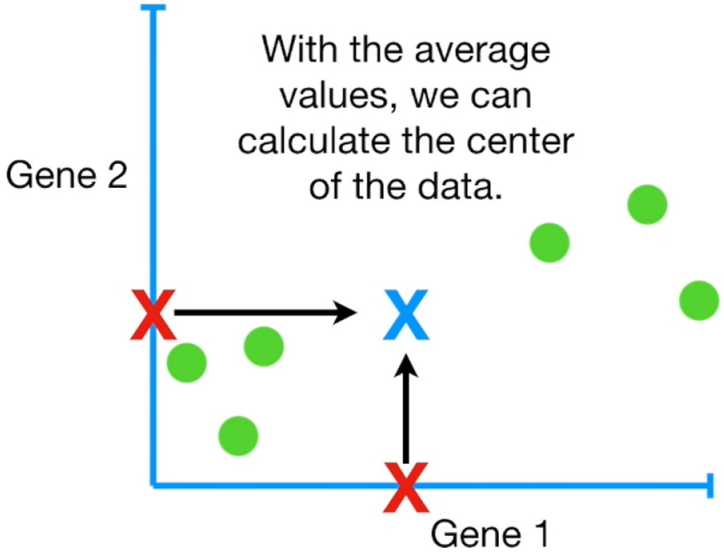
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1



	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

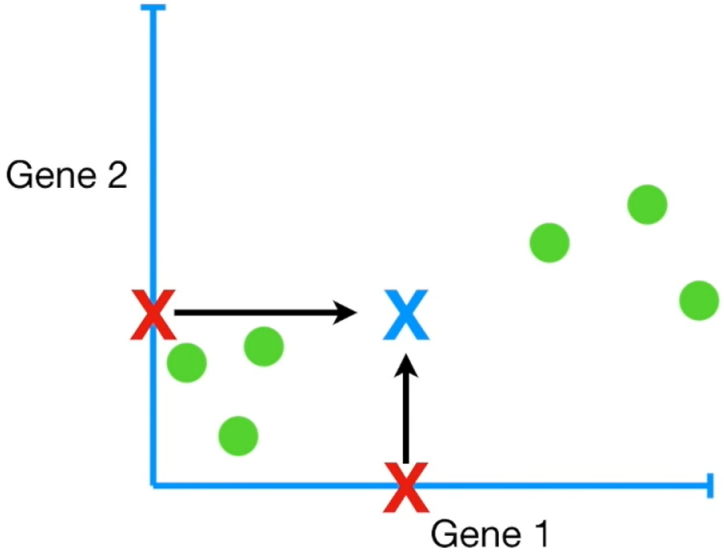


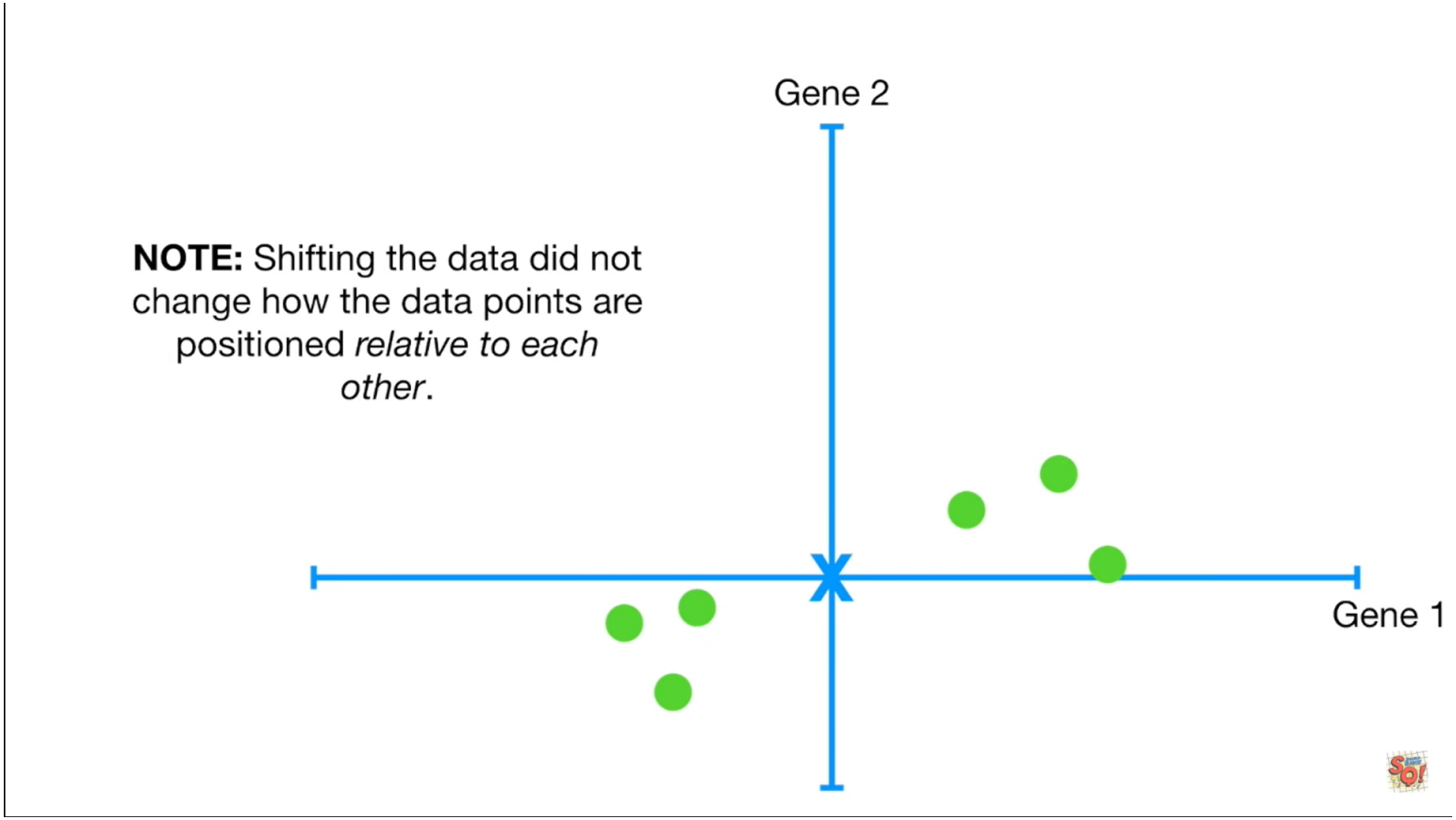
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

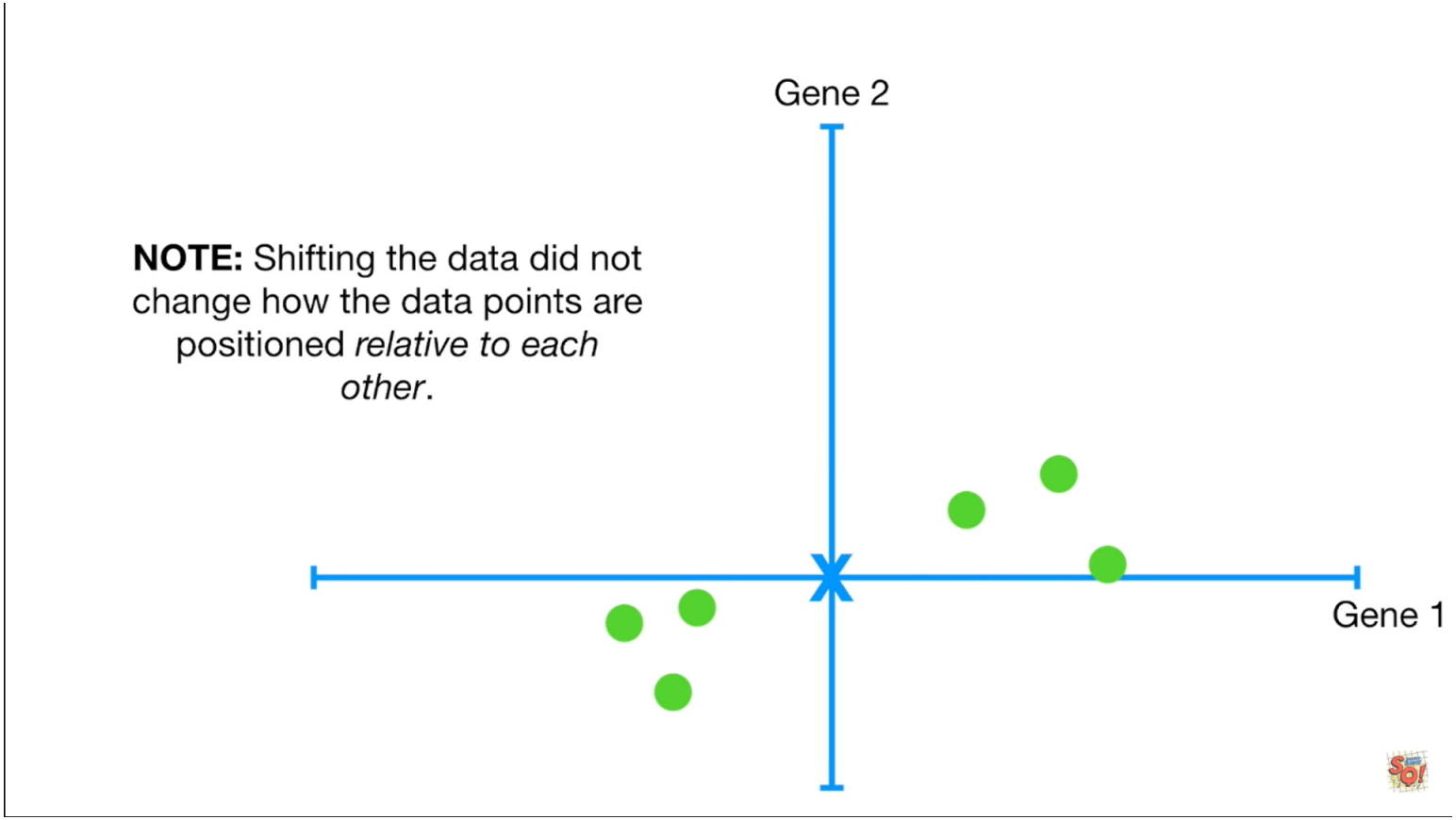


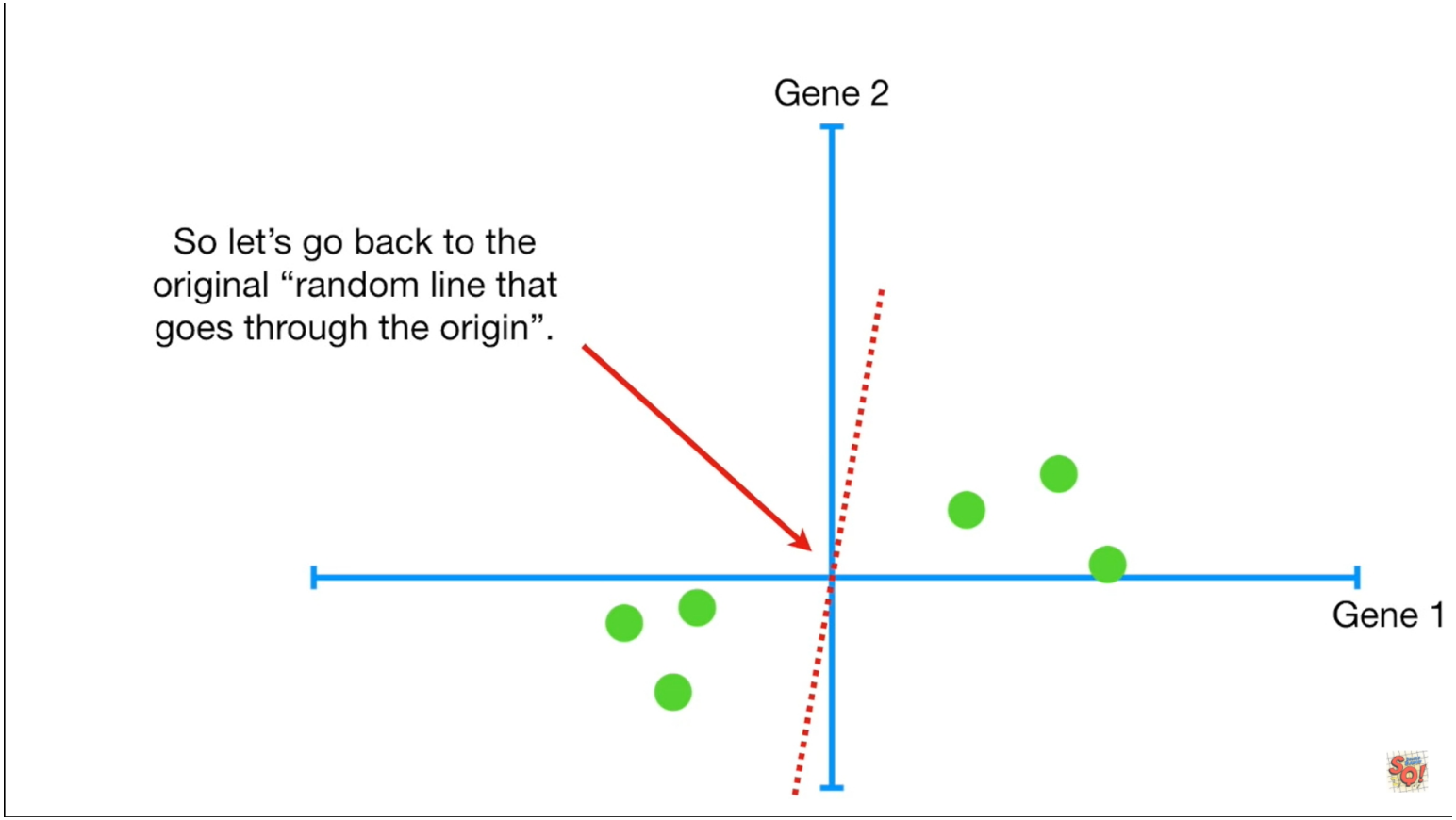
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

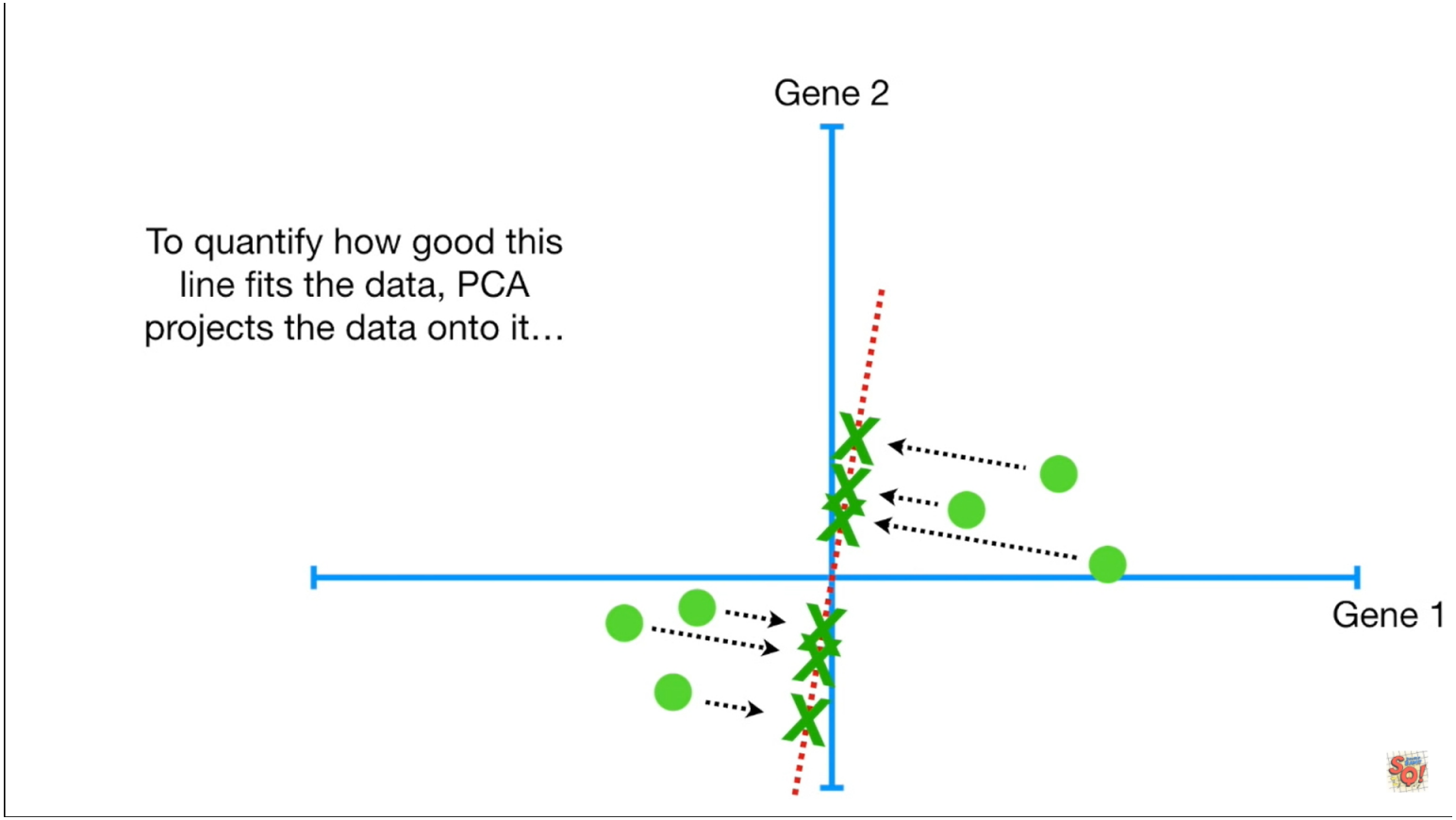
From this point on, we'll focus on what happens in the graph; we no longer need the original data…

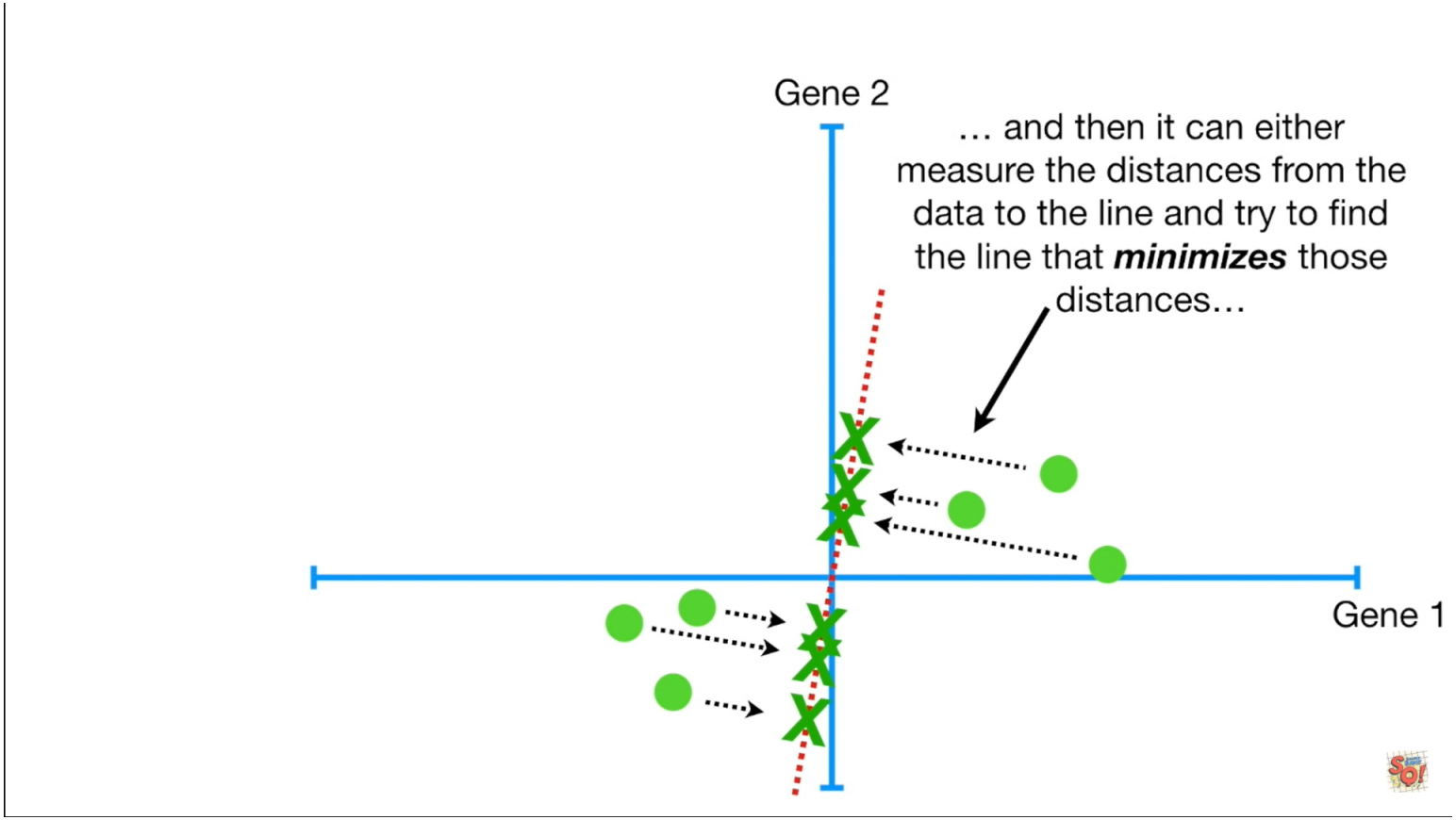


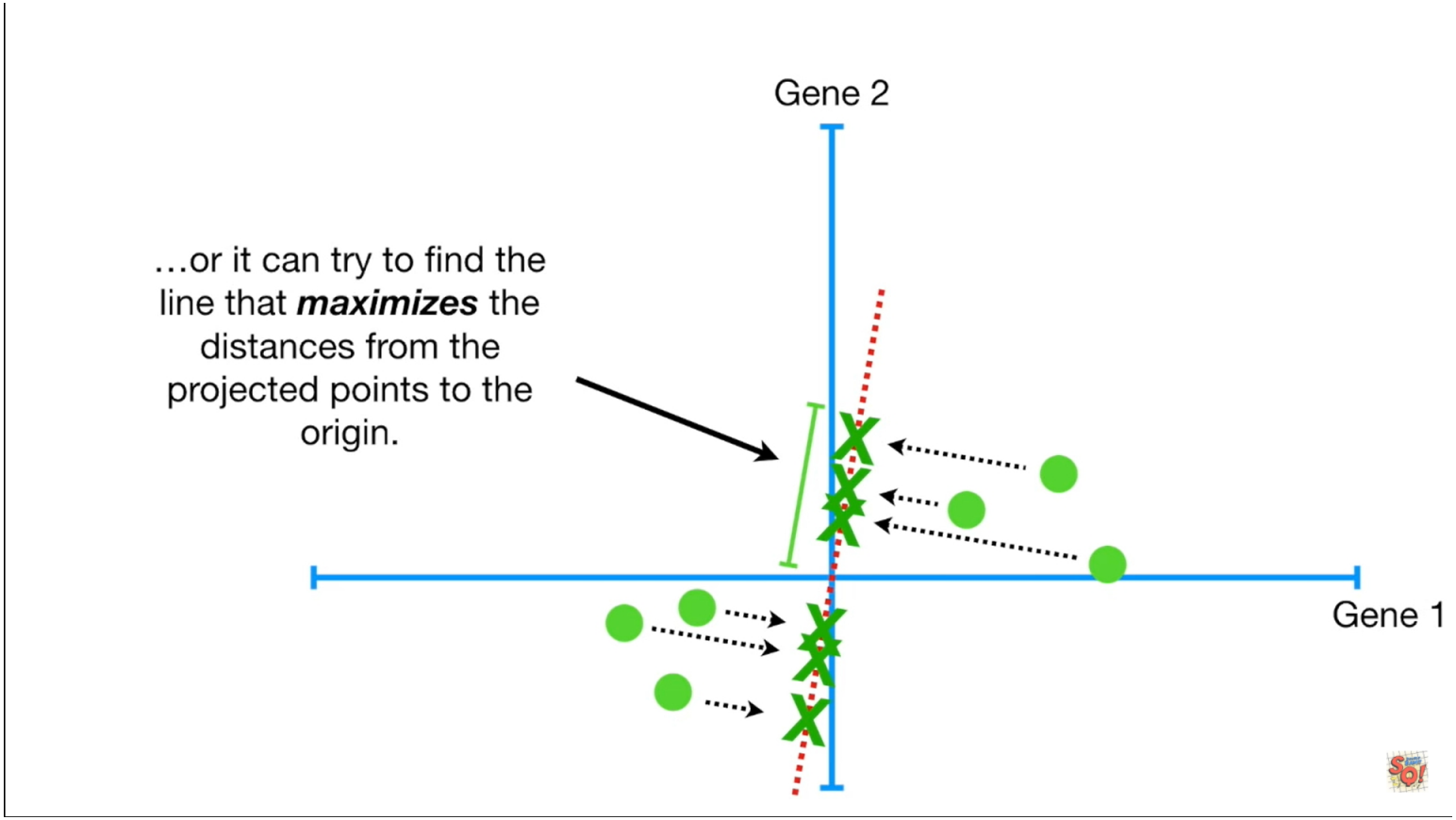

Gene 2
NOTE: Shifting the data did not change how the data points are positioned relative to each other.
Gene 1

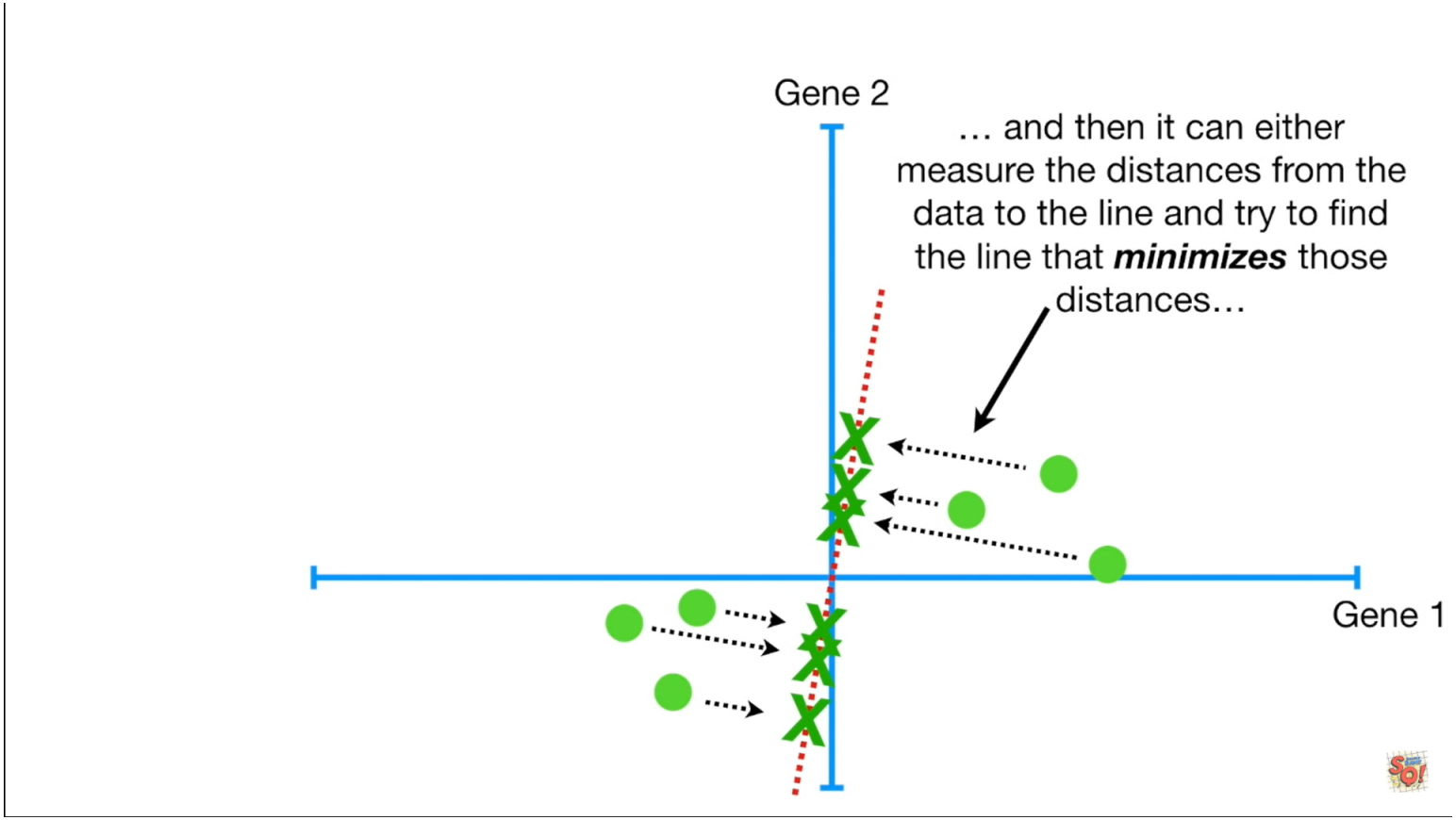

Gene 2
NOTE: Shifting the data did not change how the data points are positioned relative to each other.
Gene 1

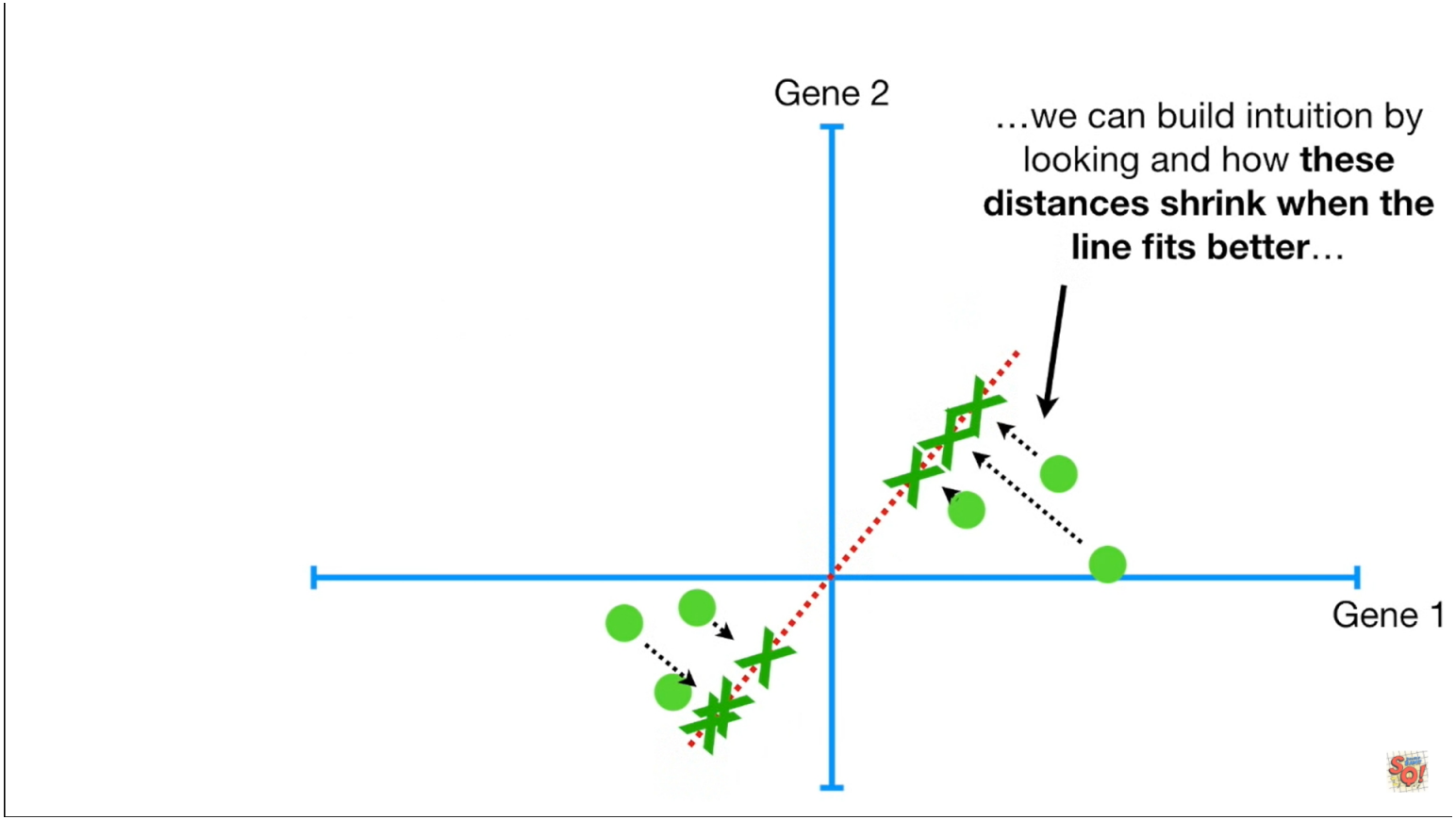

Gene 2
So let's go back to the original "random line that goes through the origin".
Gene 1


To quantify how good this line fits the data, PCA projects the data onto it…
Gene 2
Gene 1


Gene 2
… and then it can either measure the distances from the data to the line and try to find the line that ***minimizes*** those distances…
Gene 1

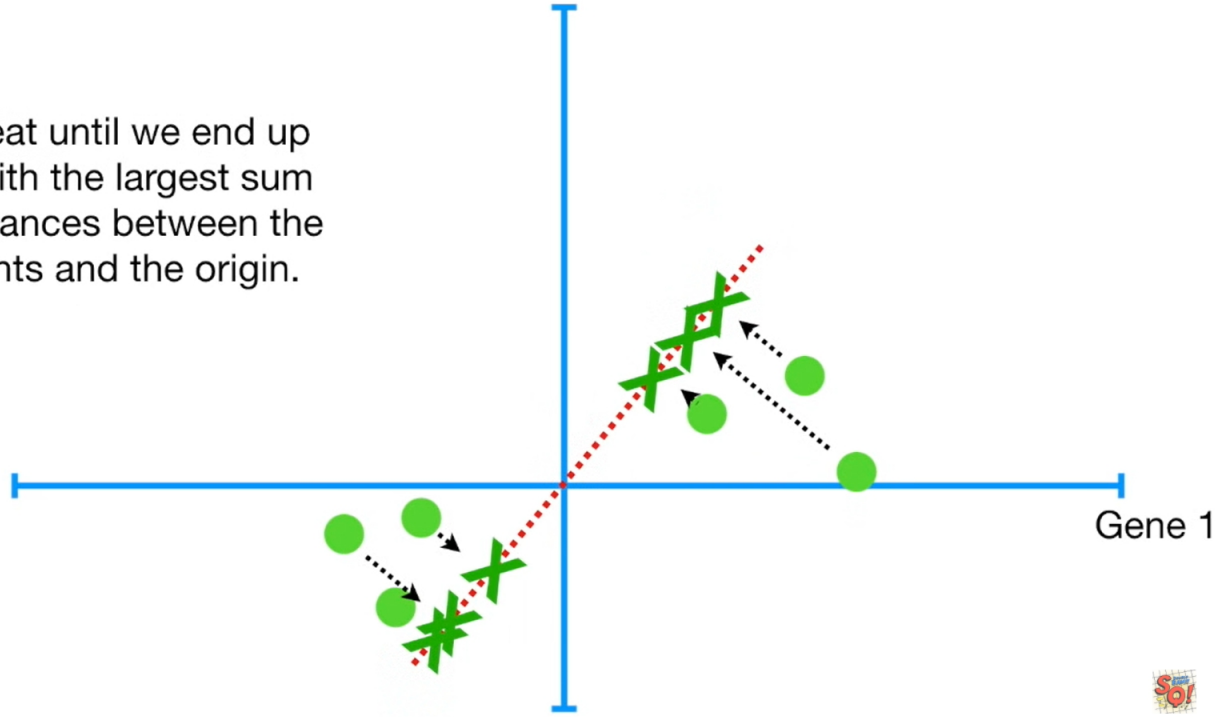

Gene 2
…or it can try to find the line that ***maximizes*** the distances from the projected points to the origin.
Gene 1


Gene 2
… and then it can either measure the distances from the data to the line and try to find the line that ***minimizes*** those distances…
Gene 1

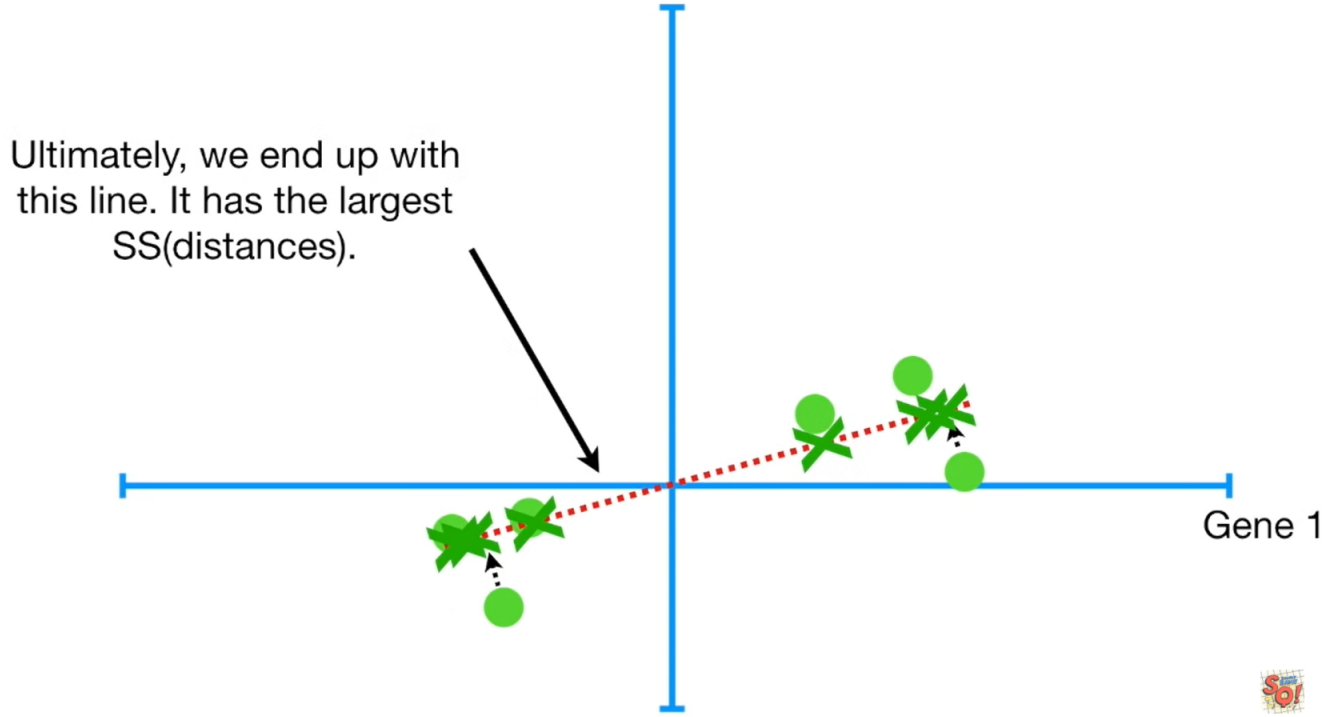

Gene 2
…we can build intuition by looking and how **these distances shrink when the line fits better**…
Gene 1

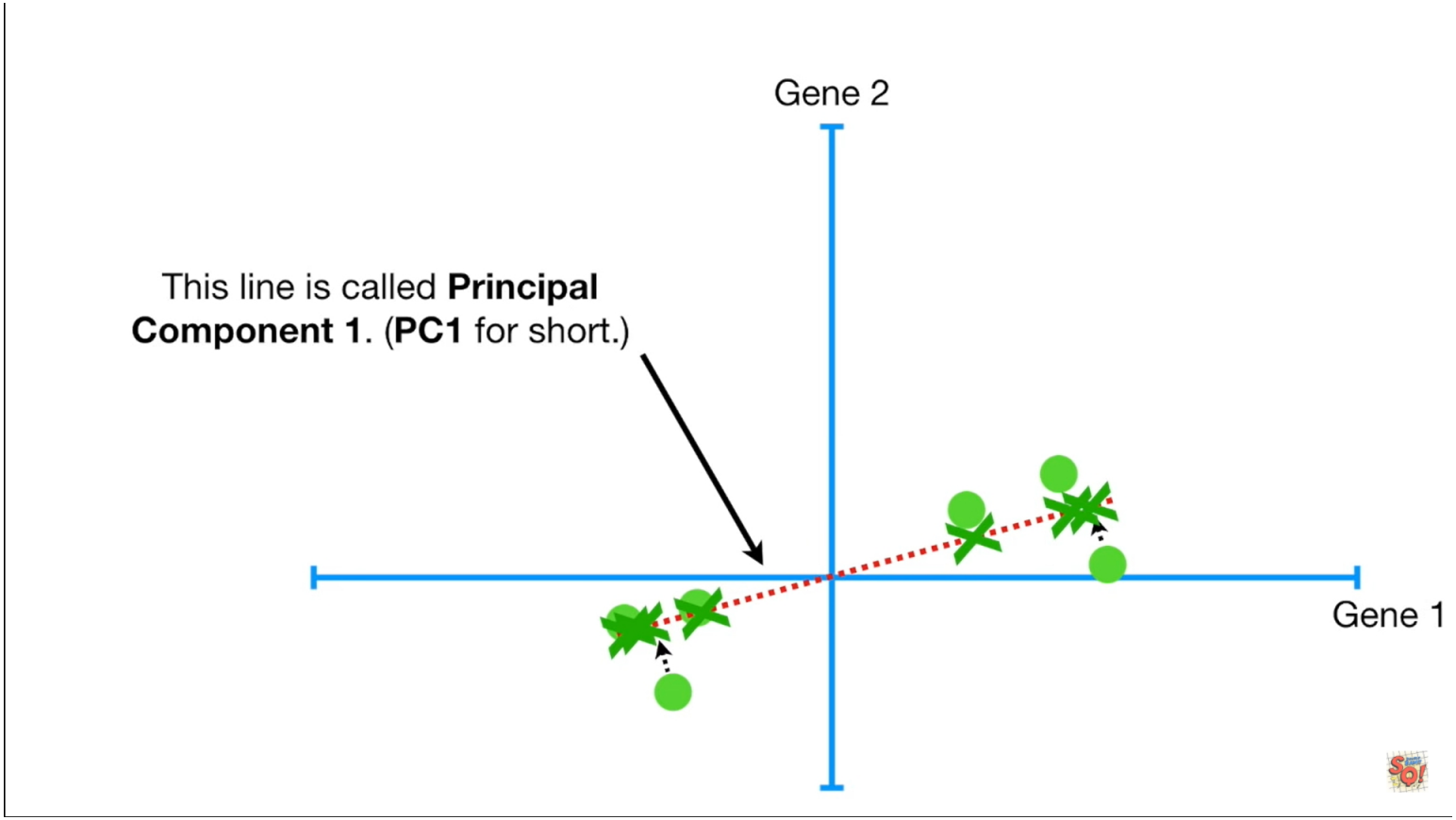
$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)

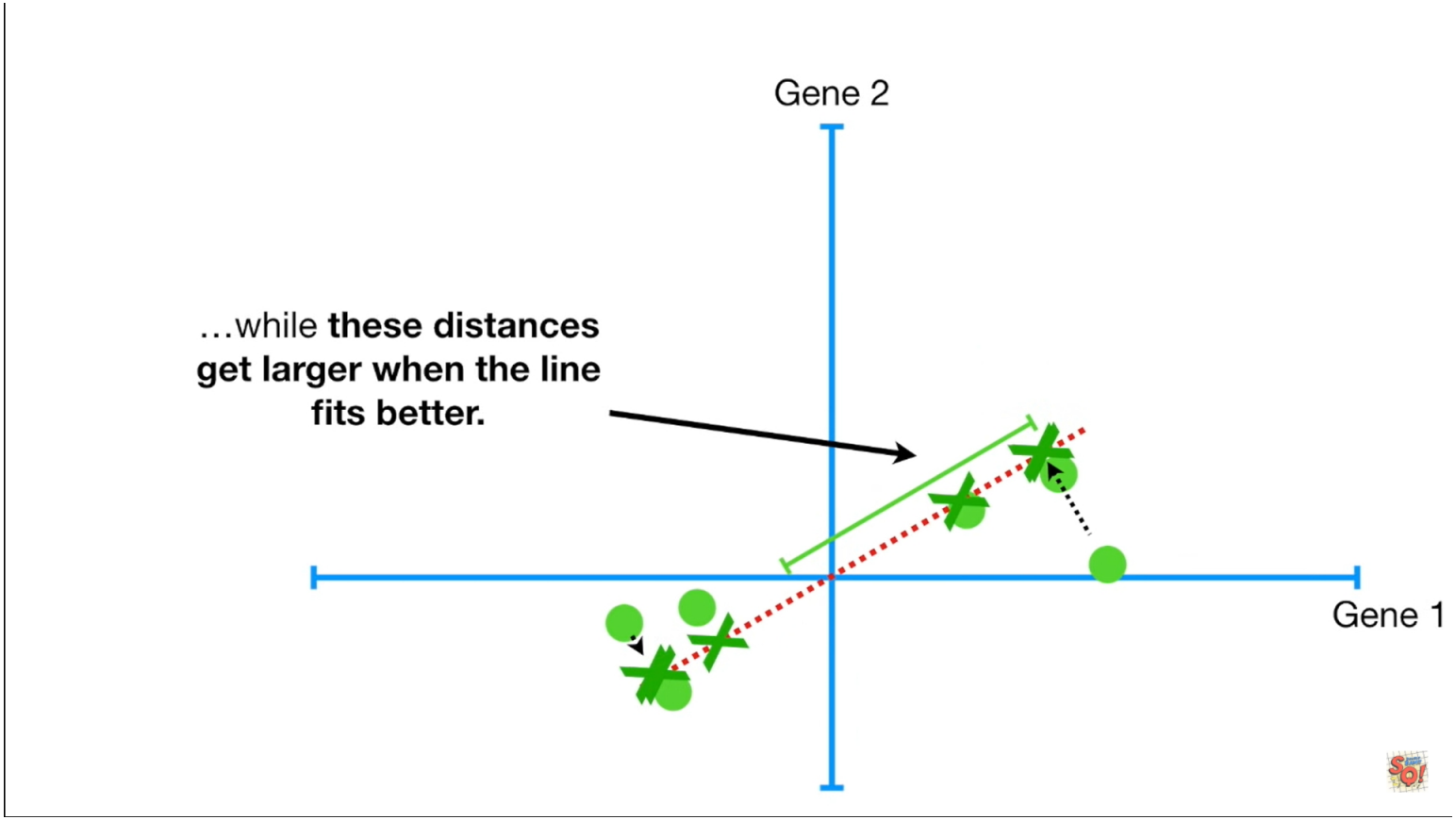
…and we repeat until we end up with the line with the largest sum of squared distances between the projected points and the origin.



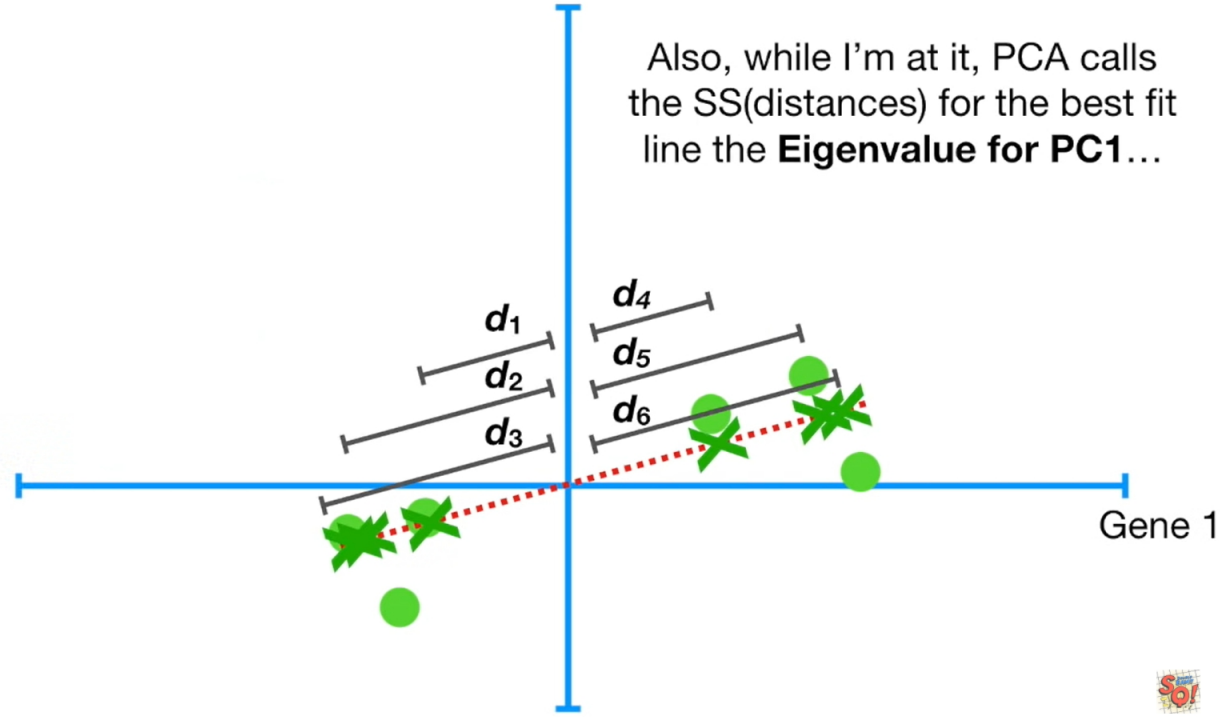
$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)




Gene 2
This line is called **Principal Component 1**. (**PC1** for short.)
Gene 1


Gene 2
…while these distances get larger when the line fits better.
Gene 1

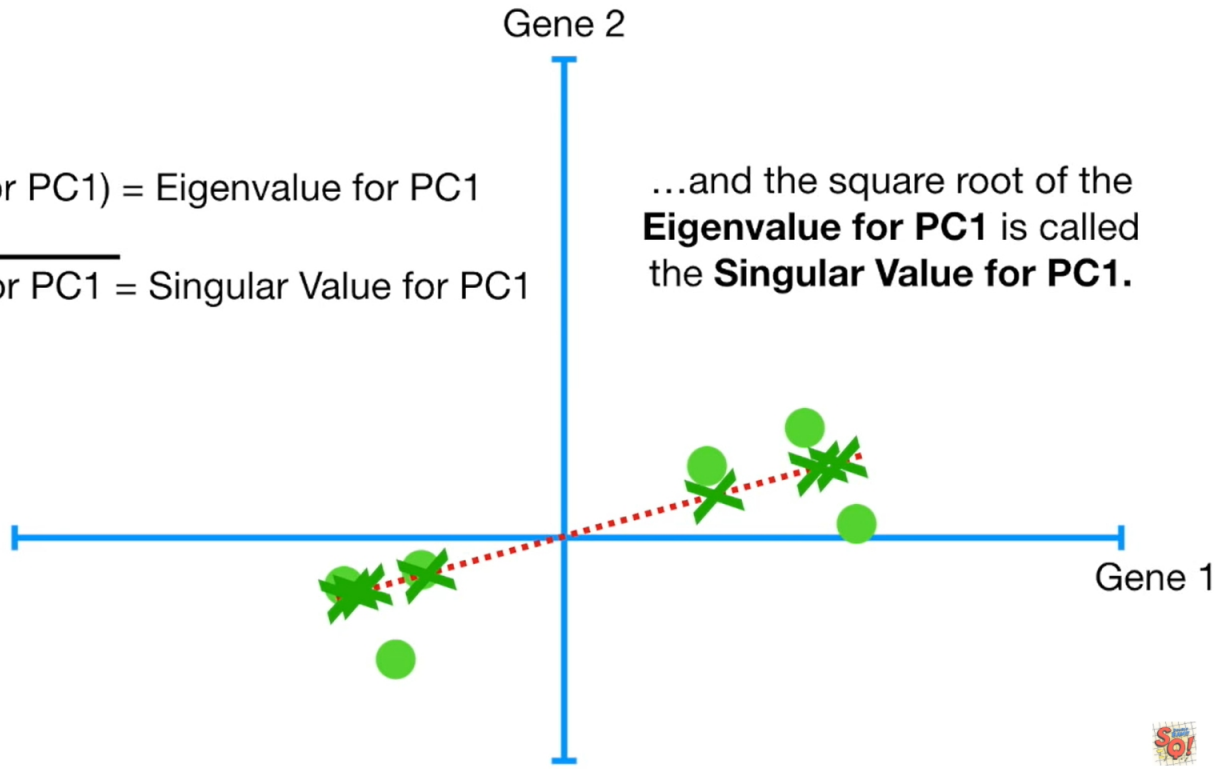
$d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = sum of squared distances = SS(distances)

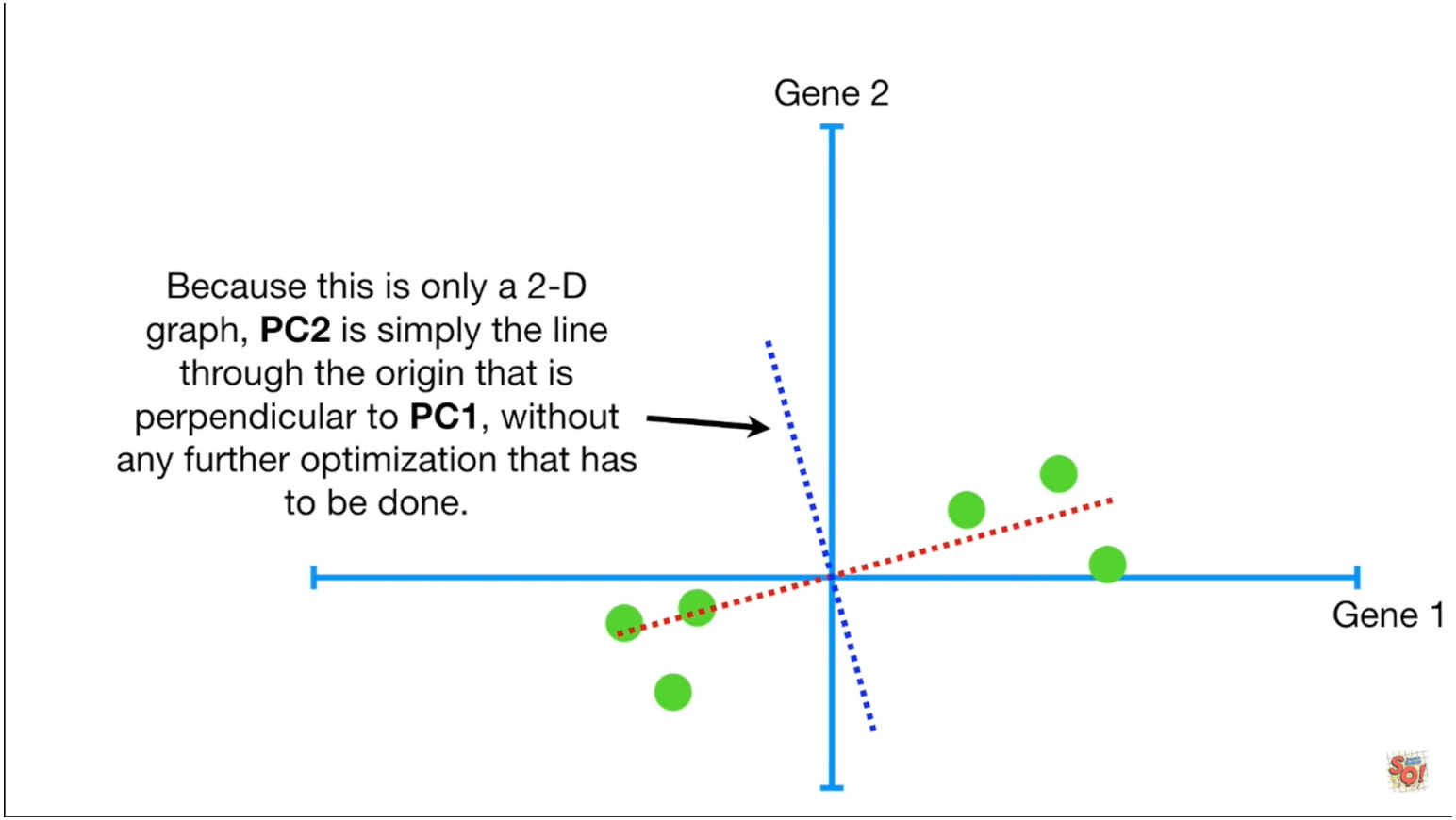


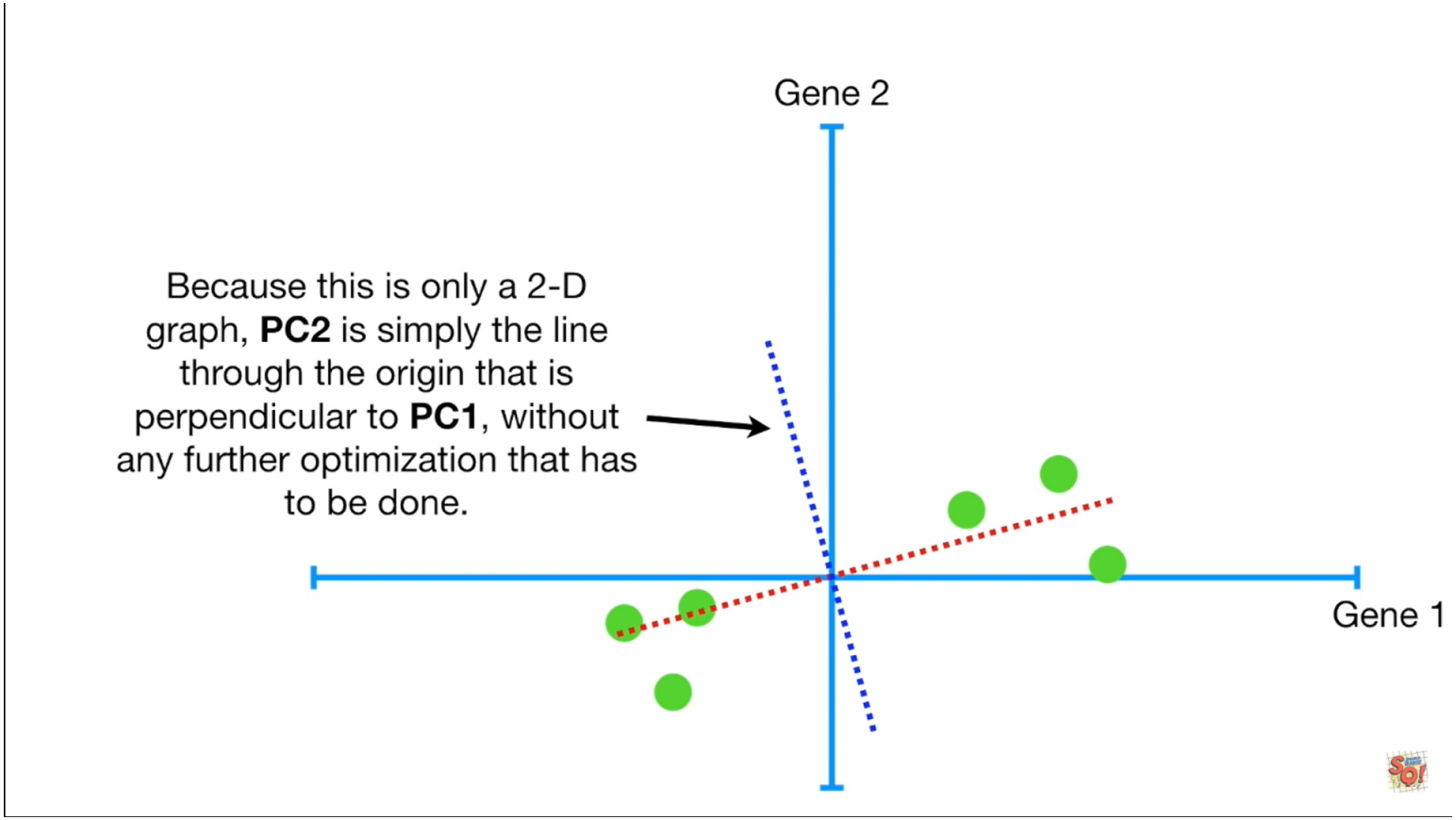
SS(distances for PC1) = Eigenvalue for PC1

$\sqrt{\text{Eigenvalue for PC1}}$ = Singular Value for PC1

…and the square root of the **Eigenvalue for PC1** is called the **Singular Value for PC1.**




Gene 2
Because this is only a 2-D graph, **PC2** is simply the line through the origin that is perpendicular to **PC1**, without any further optimization that has to be done.
Gene 1


Gene 2
Because this is only a 2-D graph, **PC2** is simply the line through the origin that is perpendicular to **PC1**, without any further optimization that has to be done.
Gene 1

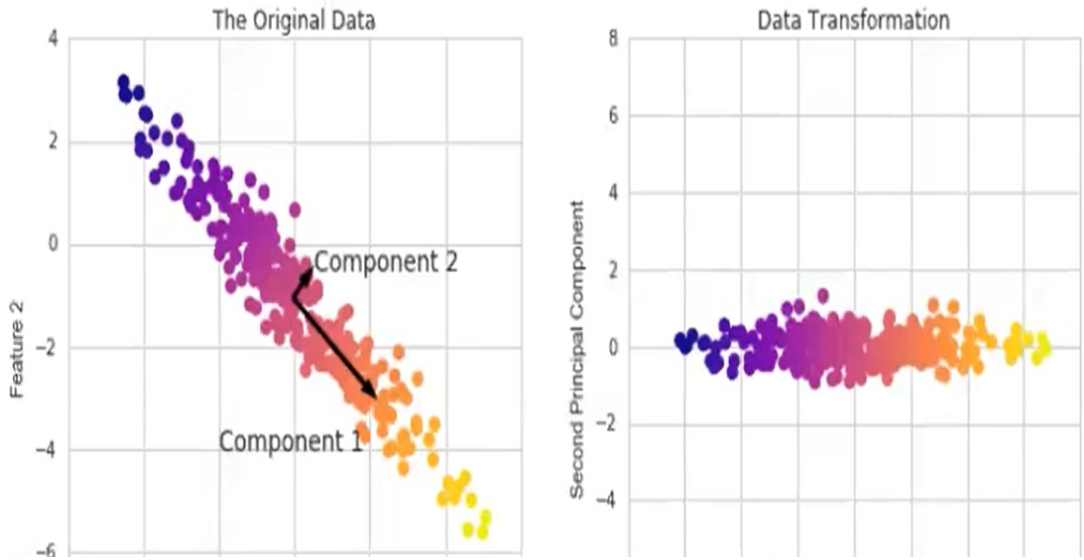
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Markdown

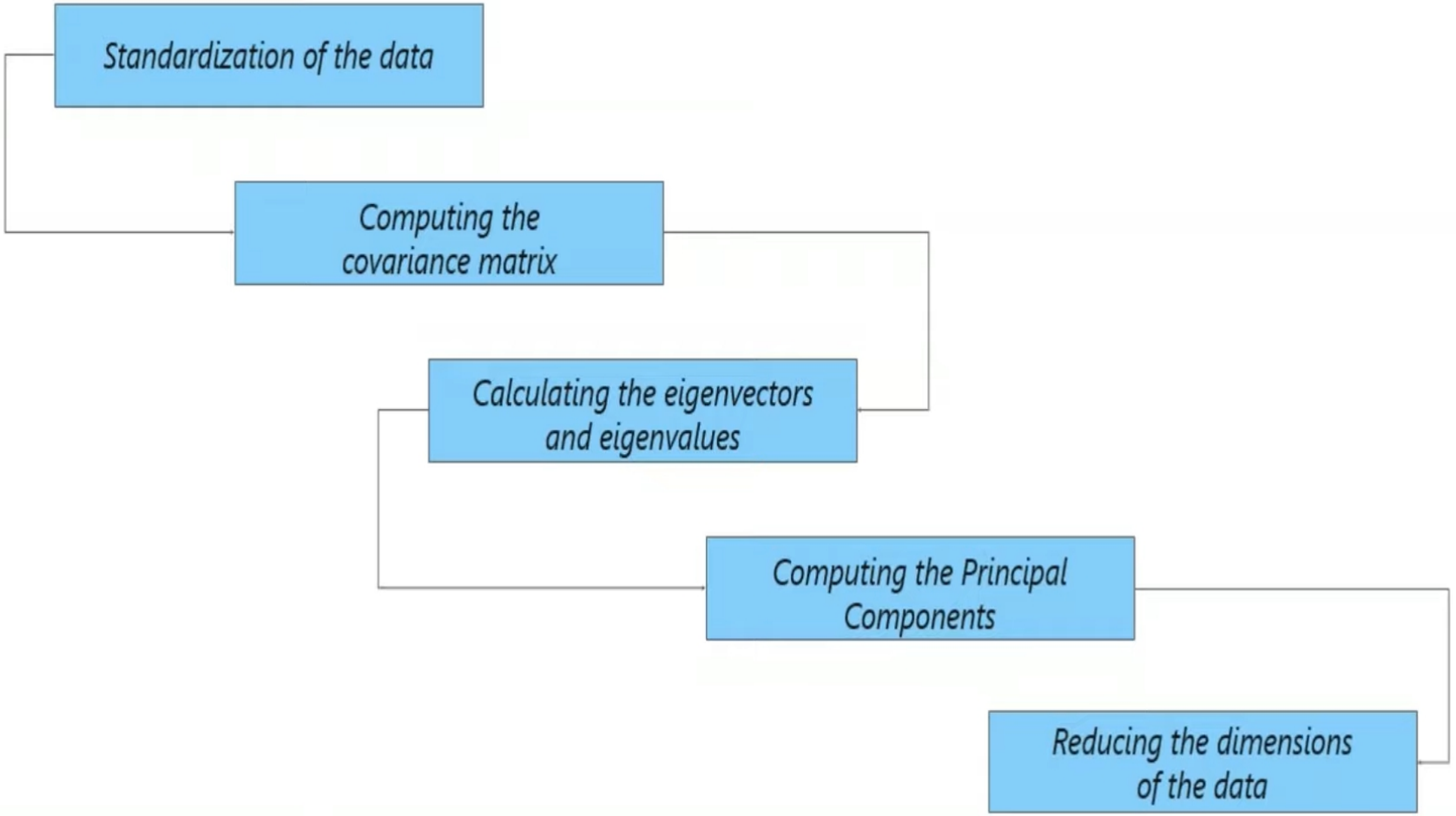
Principal Component Analysis

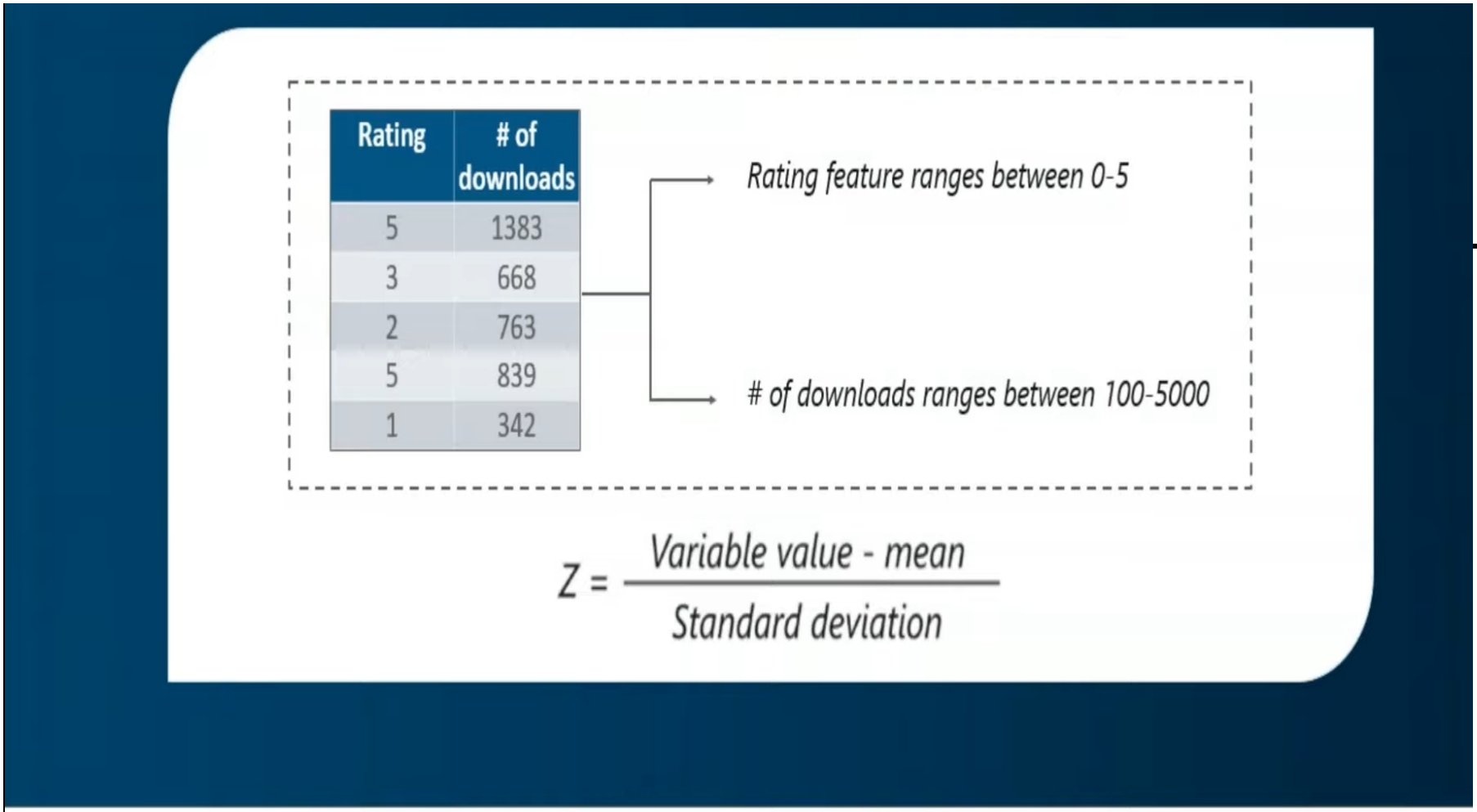
Let's discuss PCA! Since this isn't exactly a full machine learning algorithm, but instead an unsupervised learning algorithm, we will just have a lecture on this topic, but no full machine learning project (although we will walk through the cancer set with PCA).

PCA Review



STEP BY STEP PCA

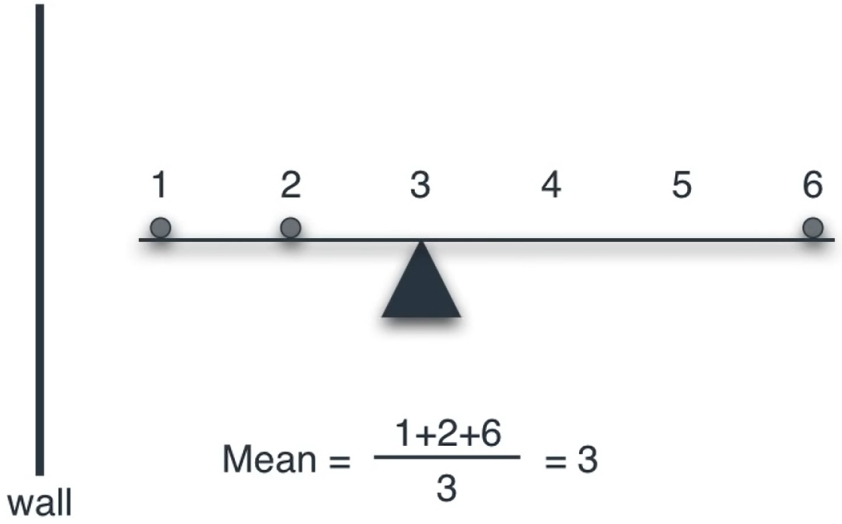




Step 1: Standardization of the data

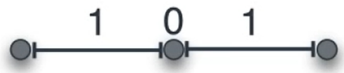
Standardization is all about scaling your data in such a way that all the variables and their values lie within a similar range.

Mean



$$\text{Mean} = \frac{1+2+6}{3} = 3$$

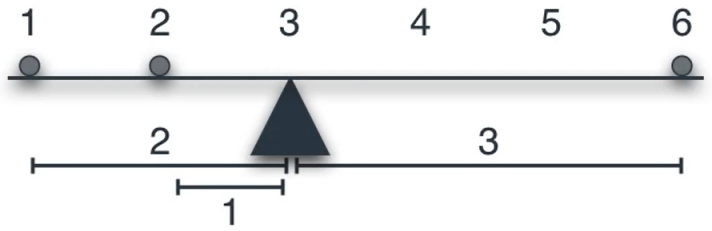
Variance



$$\text{Variance} = \frac{1^2+0^2+1^2}{3} = 2/3$$

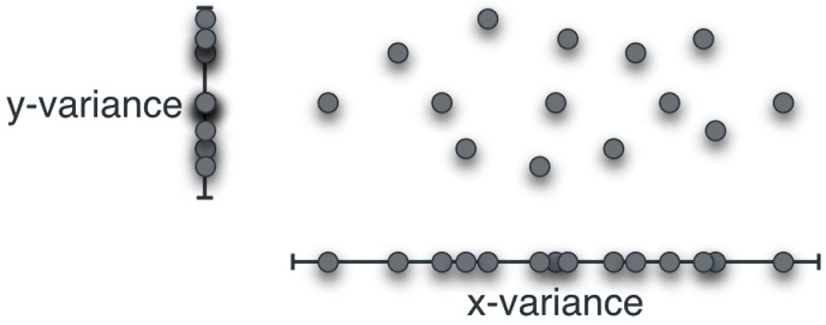
$$\text{Variance} = \frac{5^2+0^2+5^2}{3} = 50/3$$

Mean



$$\text{Variance} = \frac{2^2+1^2+3^2}{3} = 14/3$$

Variance?

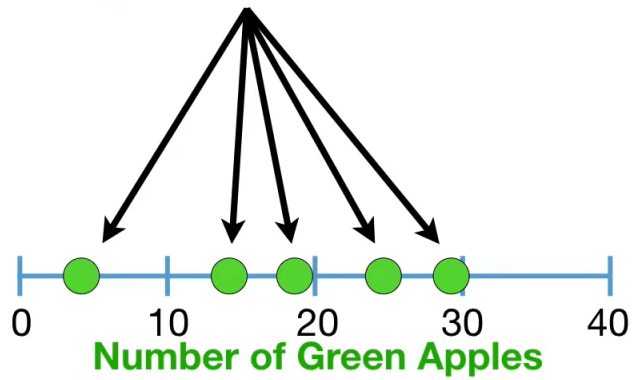


Variance?

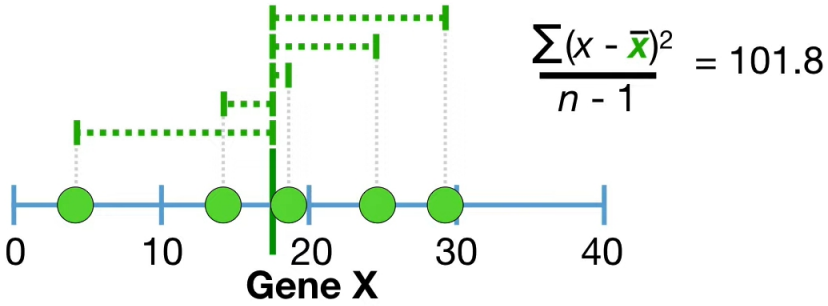
$$\text{x-variance} = \frac{2^2+0^2+2^2}{3} = 8/3$$

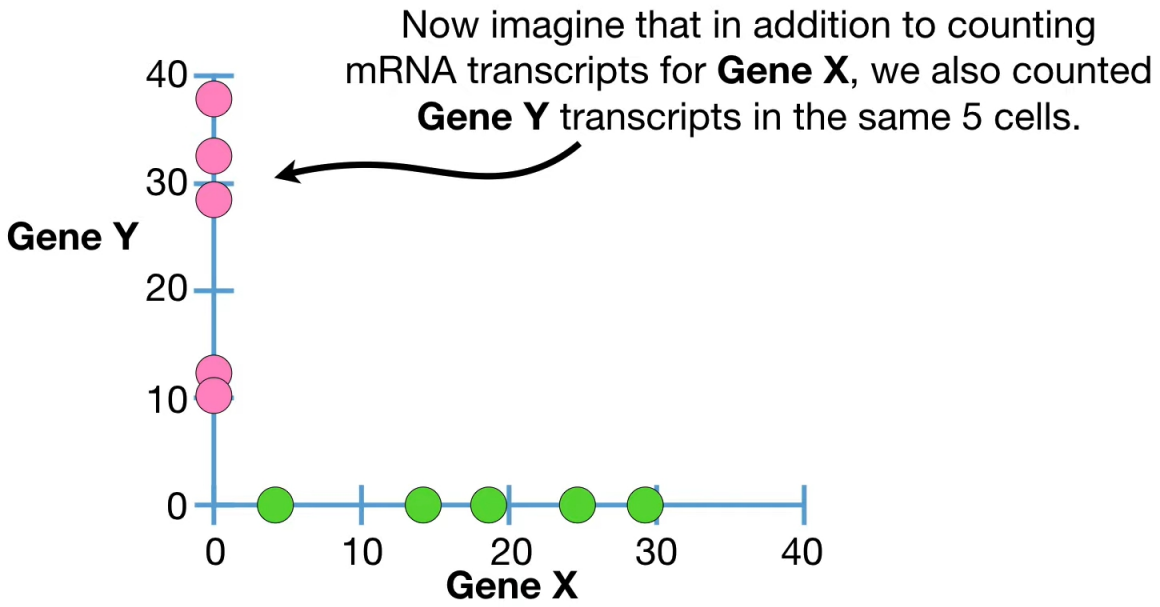
$$\text{y-variance} = \frac{1^2+0^2+1^2}{3} = 2/3$$

And if mRNA transcripts aren't your thing, you could think that we counted the number of **green apples** in **5** different grocery stores.

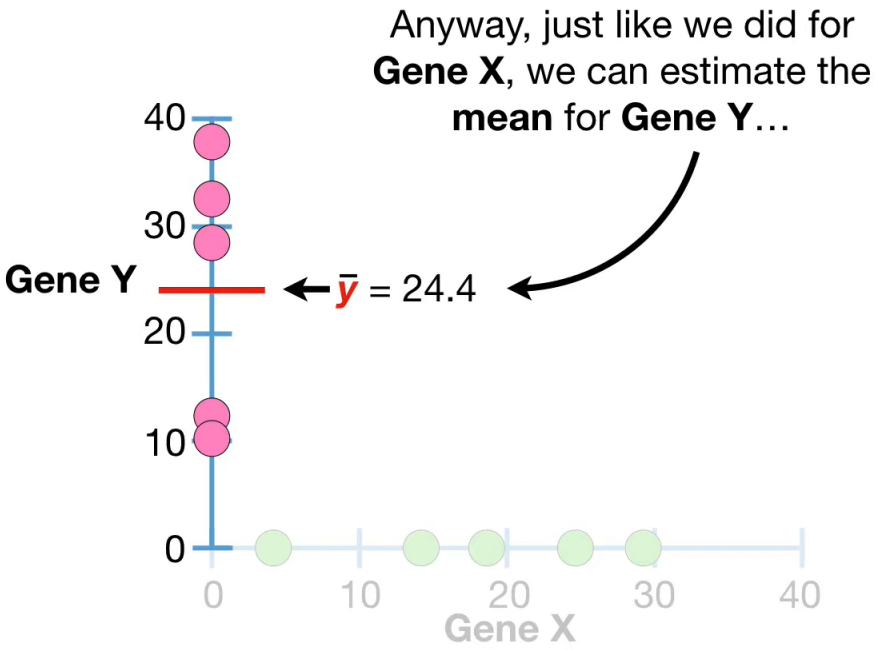


And that's our review of **Variance**.



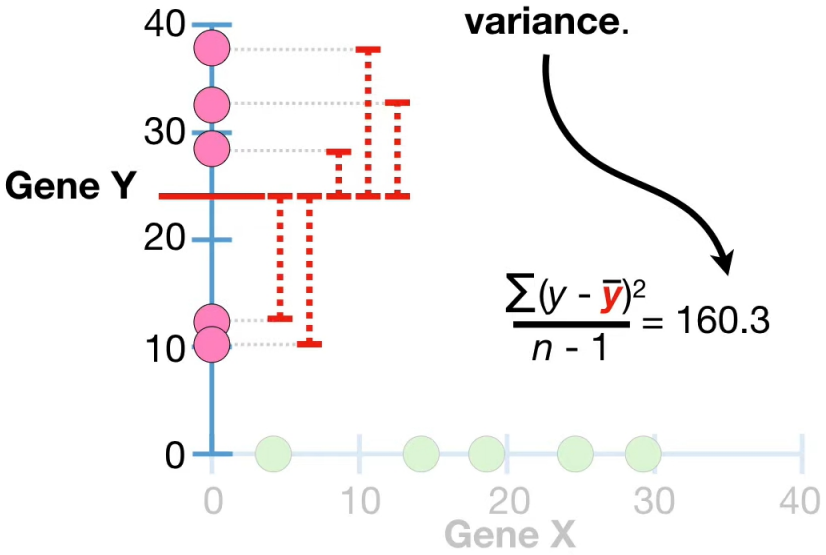

Now imagine that in addition to counting mRNA transcripts for **Gene X**, we also counted **Gene Y** transcripts in the same 5 cells.
Gene Y
40
30
20
10
0
0
10
20
30
40
Gene X


Alternatively, you can imagine we counted the number of red apples in the same 5 grocery stores.
Number of Red Apples
40
30
20
10
0
0
10
20
30
40
Number of Green Apples

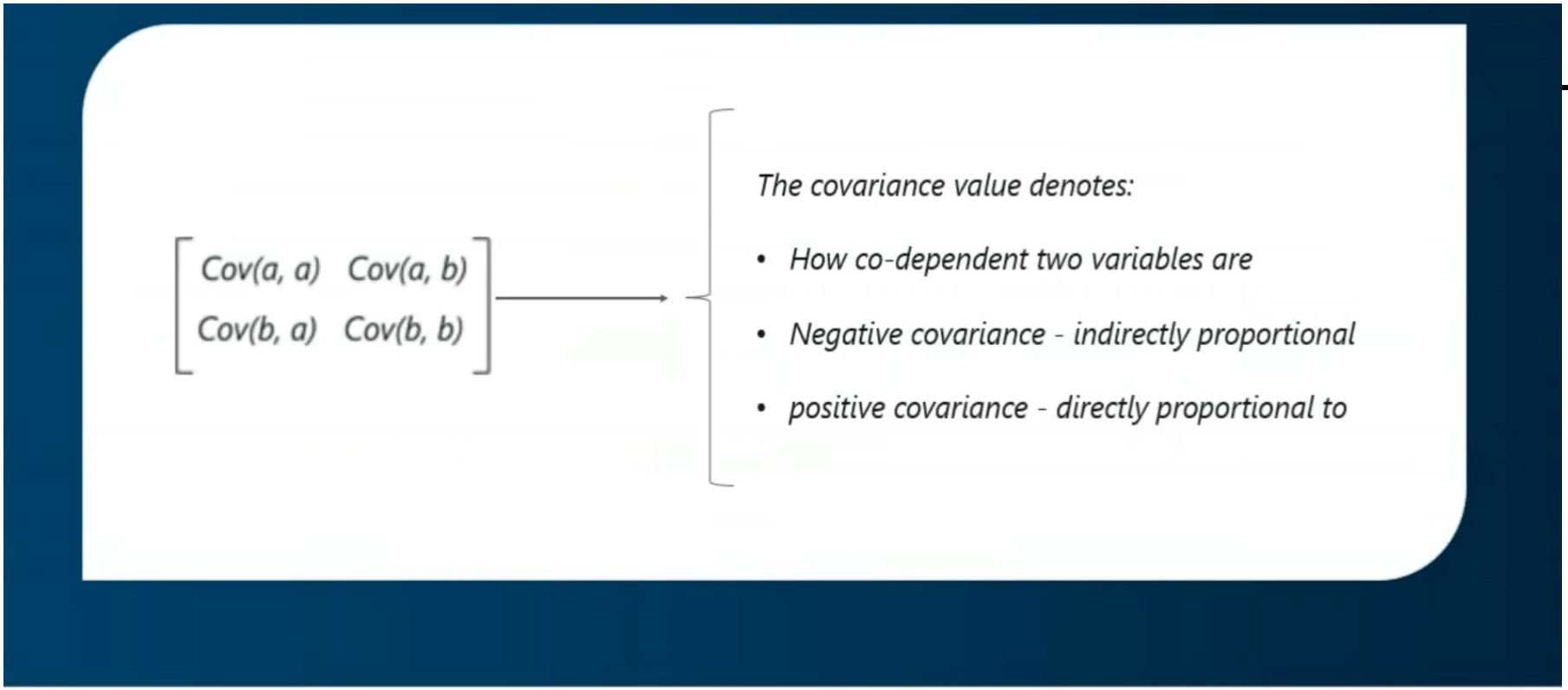

Anyway, just like we did for
Gene X, we can estimate the
mean for Gene Y…
40
30
Gene Y
$\bar{y}$ = 24.4
20
10
0
0
10
20
30
40
Gene X

…and we can estimate the **variance**.

bam.



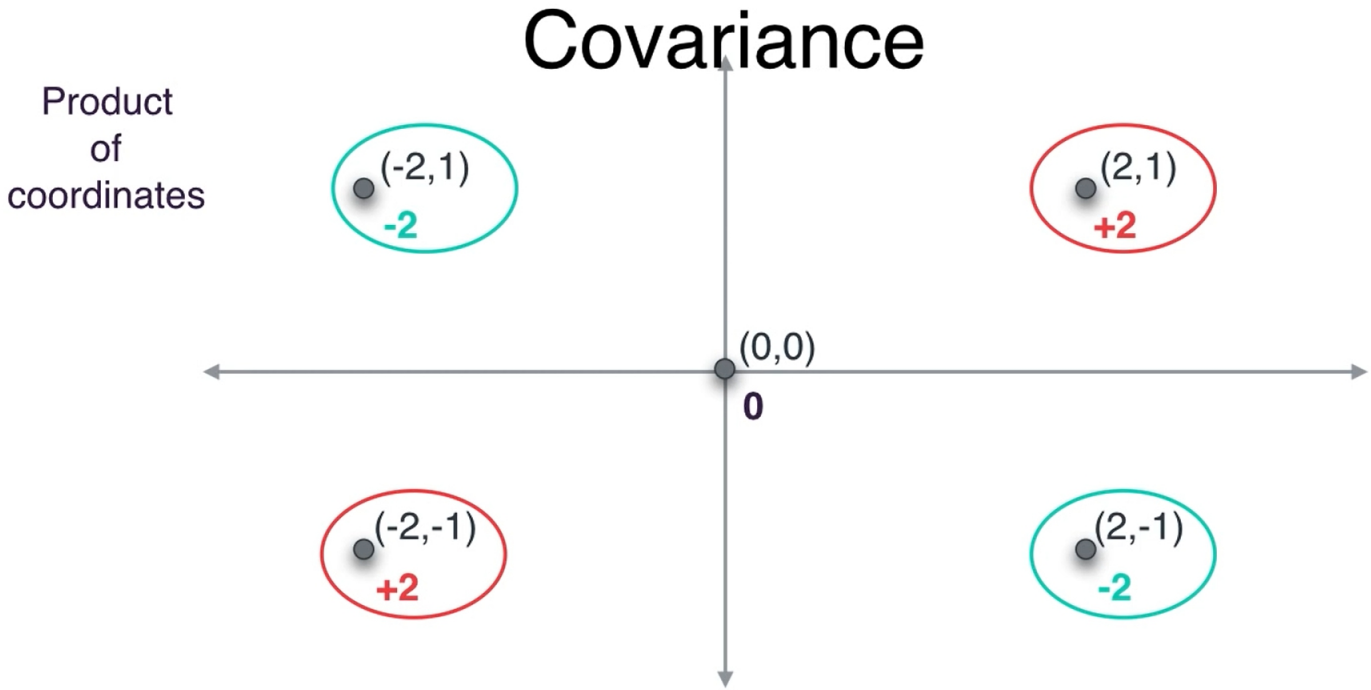
Covariance Matrix



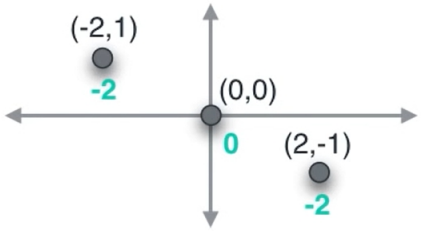
Step 2: Computing the covariance matrix

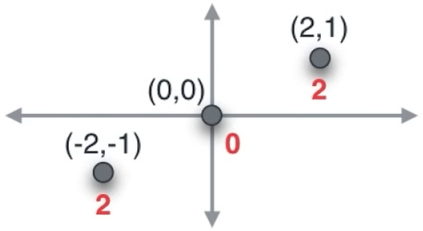
A covariance matrix expresses the correlation between the different variables in the data set. It is essential to identify heavily dependent variables because they contain biased and redundant information which reduces the overall performance of the model.

Covariance



Covariance

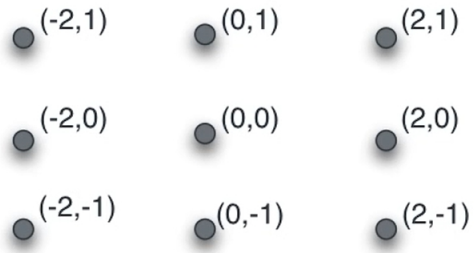




$$\text{covariance} = \frac{(-2) + 0 + (-2)}{3} = -4/3$$

$$\text{covariance} = \frac{2 + 0 + 2}{3} = 4/3$$

Covariance



$$\text{covariance} = \frac{-2 + 0 + 2 + 0 + 0 + 0 + 2 + 0 + -2}{9} = 0$$

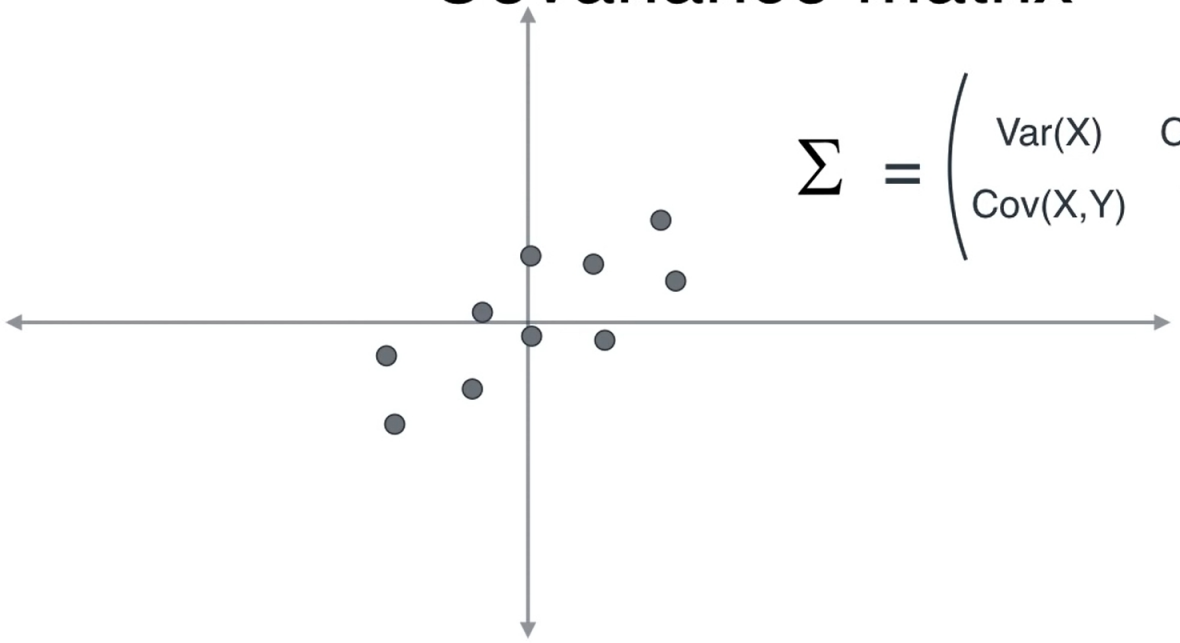
Covariance

negative
covariance

covariance zero
(or very small)

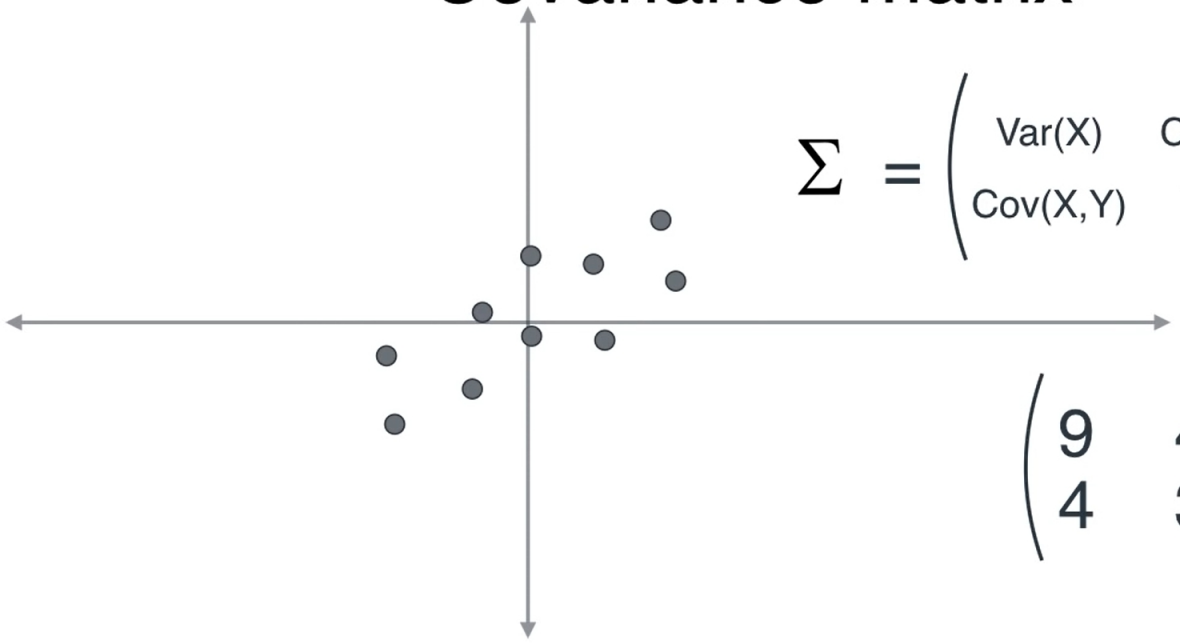
positive
covariance

Covariance matrix



$$\Sigma = \begin{pmatrix} \mathrm{Var}(X) & \mathrm{Cov}(X,Y) \\ \mathrm{Cov}(X,Y) & \mathrm{Var}(Y) \end{pmatrix}$$

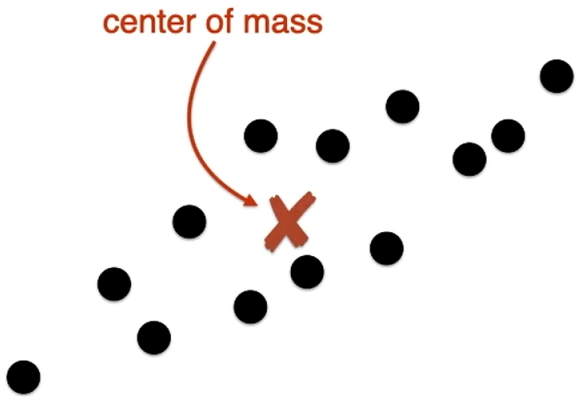
Covariance matrix



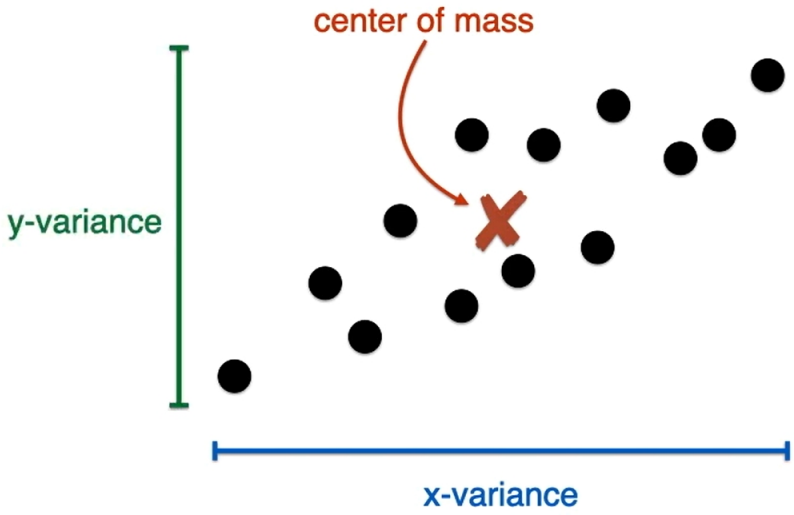
$$\Sigma = \begin{pmatrix} \text{Var}(X) & \text{Cov}(X,Y) \\ \text{Cov}(X,Y) & \text{Var}(Y) \end{pmatrix}$$

$$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

The covariance matrix

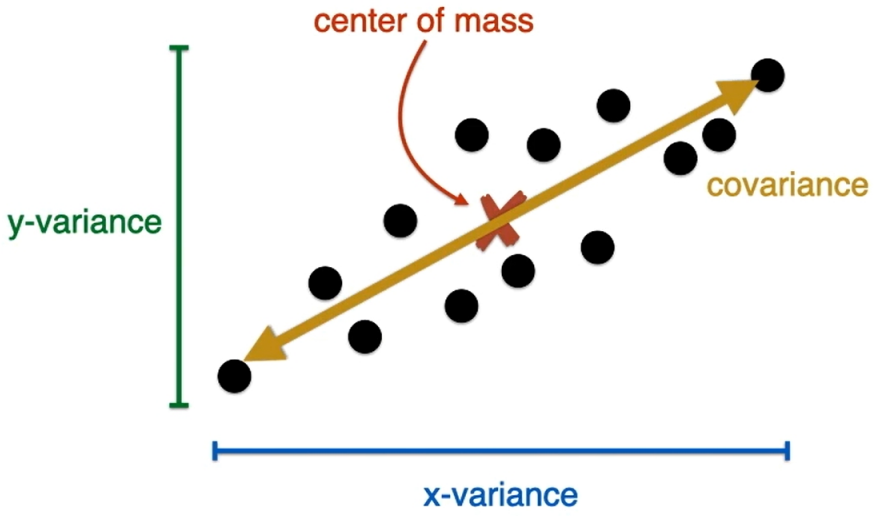


The covariance matrix



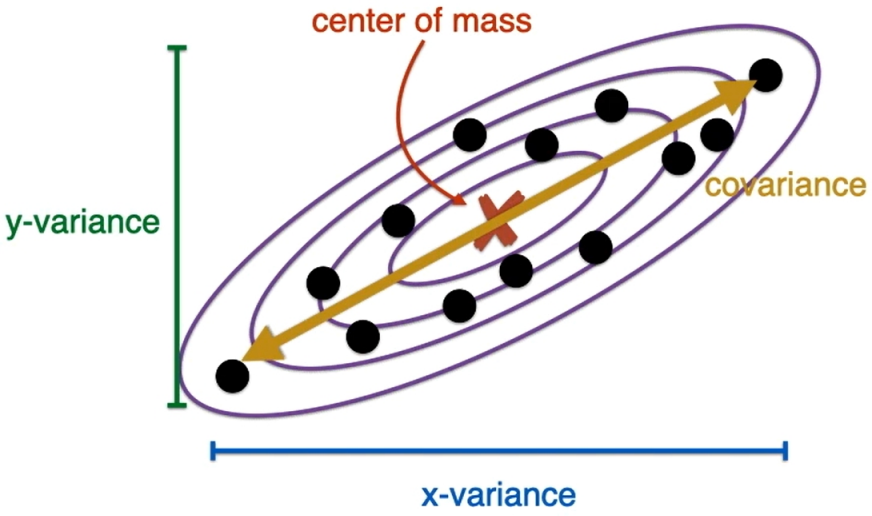
$\mu = Average$

The covariance matrix



$\mu = Average$

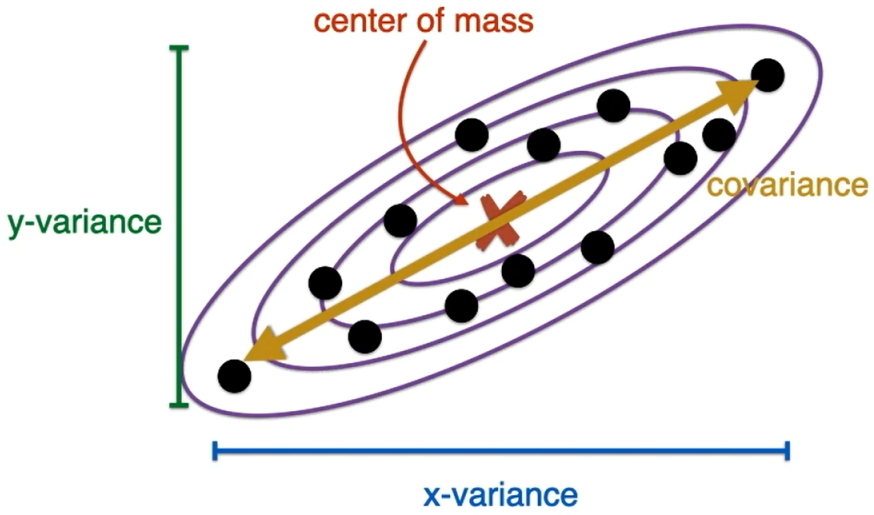
The covariance matrix



$\mu = Average$

Σ

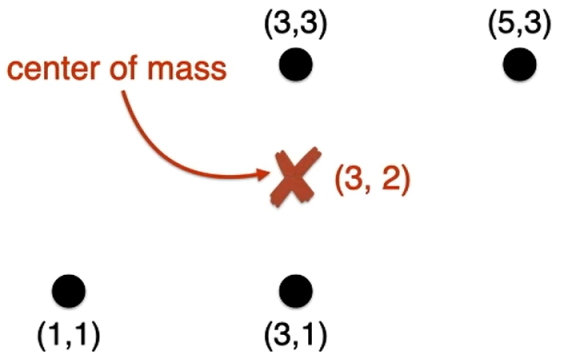
The covariance matrix



$$\mu = Average$$

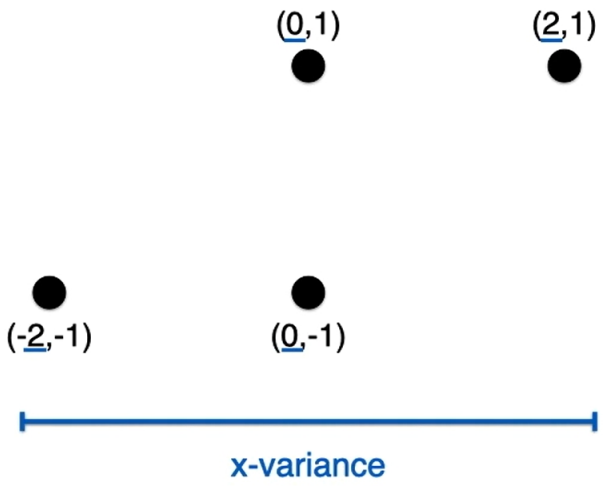
$$\Sigma = \begin{pmatrix} Var(x) & Cov(x,y) \\ Cov(x,y) & Var(y) \end{pmatrix}$$

Average



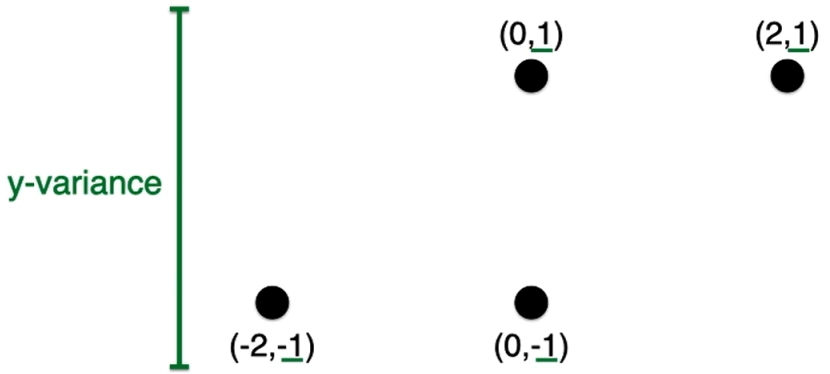
$$\left(\frac{1+3+3+5}{4}, \frac{1+1+3+3}{4}\right) = (3,2)$$

X-variance

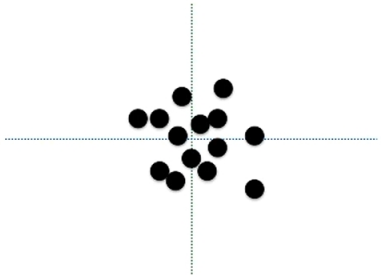


$$var(x) = \frac{1}{4}((-2)^2 + 0^2 + 0^2 + 2^2) = 2$$

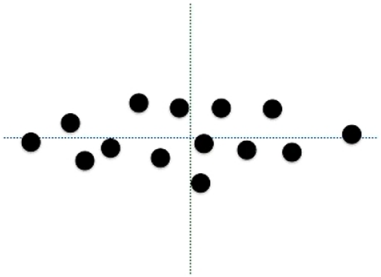
Y-variance



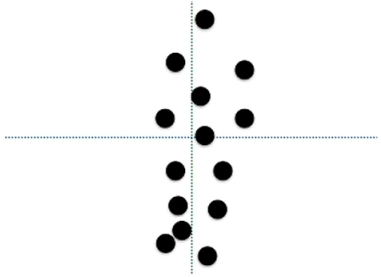
$$var(y) = \frac{1}{4}((-1)^2 + (-1)^2 + 1^2 + 1^2) = 1$$



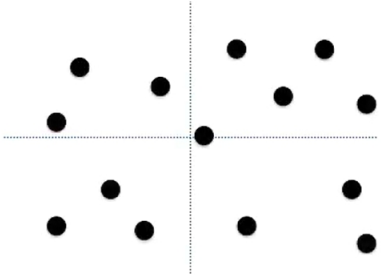
Small x-variance
Small y-variance



Large x-variance
Small y-variance

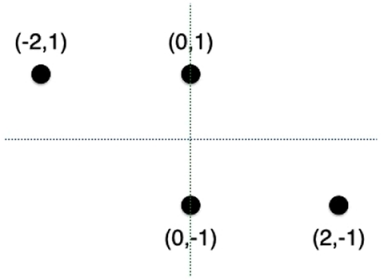


Small x-variance
Large y-variance



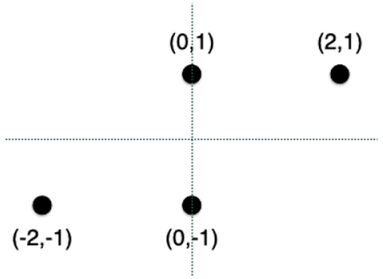
Large x-variance
Large y-variance

Problem: Same variances



$$var(x) = \frac{1}{4}((-2)^2 + 0^2 + 0^2 + 2^2) = 2$$

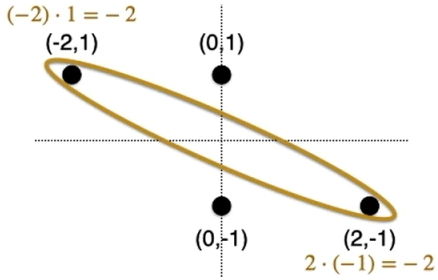
$$var(y) = \frac{1}{4}(1^2 + 1^2 + (-1)^2 + (-1)^2) = 1$$

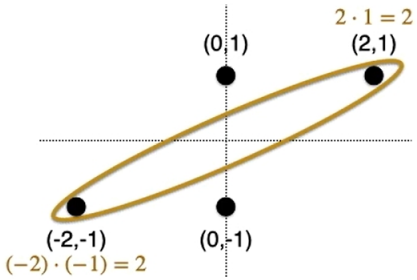


$$var(x) = \frac{1}{4}((-2)^2 + 0^2 + 0^2 + 2^2) = 2$$

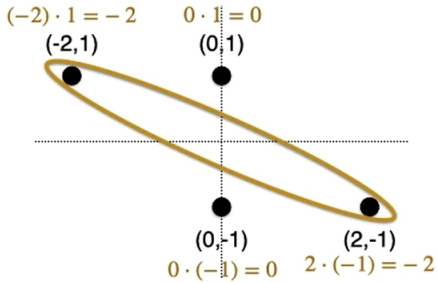
$$var(y) = \frac{1}{4}((-1)^2 + (-1)^2 + 1^2 + 1^2) = 1$$

Solution: Covariance



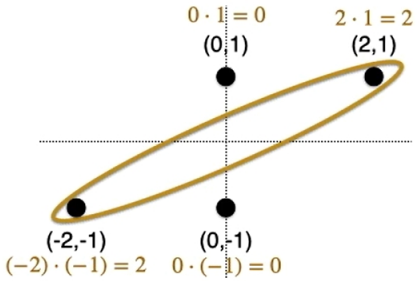


Solution: Covariance



$$cov(x,y) = \frac{1}{4}(-2+0+0-2)$$

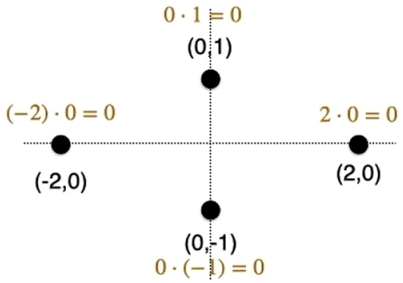
$$cov(x,y) = -1$$



$$cov(x,y) = \frac{1}{4}(2+0+0+2)$$

$$cov(x,y) = 1$$

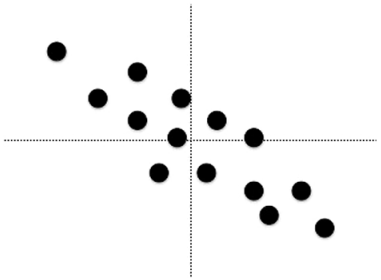
Solution: Covariance



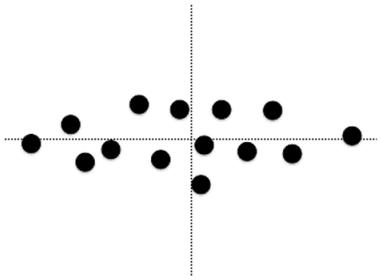
$$cov(x, y) = \frac{1}{4}(0 + 0 + 0 + 0)$$

$$cov(x, y) = 0$$

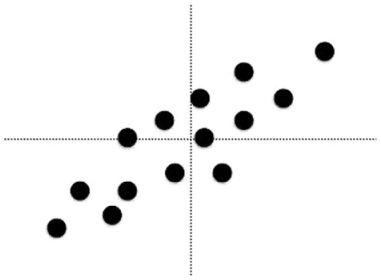
In general



Negative covariance

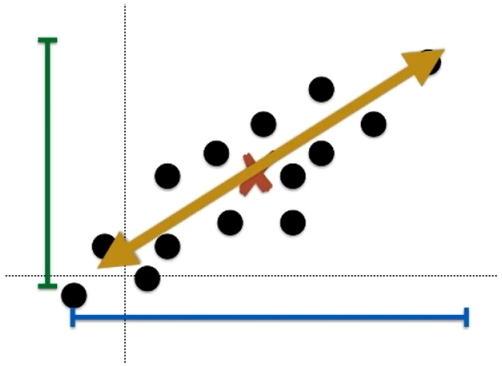


Zero covariance
(or very small)



Positive covariance

Formulas



Points

(x_1, y_1)
(x_2, y_2)
$\vdots$
(x_n, y_n)

Mean

$$(\mu_x, \mu_y) = (\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i)$$

x-Variance

$$var(x) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu_x)^2$$

y-Variance

$$var(y) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \mu_y)^2$$

Covariance

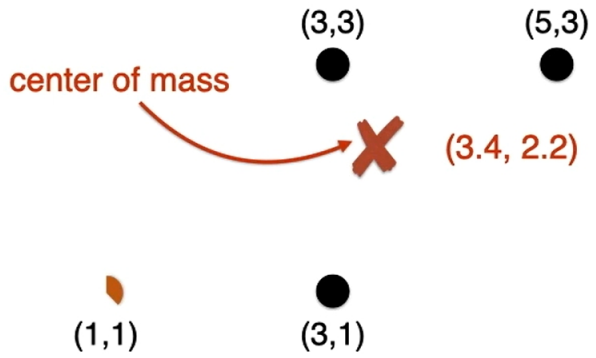
$$cov(x, y) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)$$

Covariance matrix

$$\Sigma = \begin{pmatrix} var(x) & cov(x, y) \\ cov(x, y) & var(y) \end{pmatrix}$$

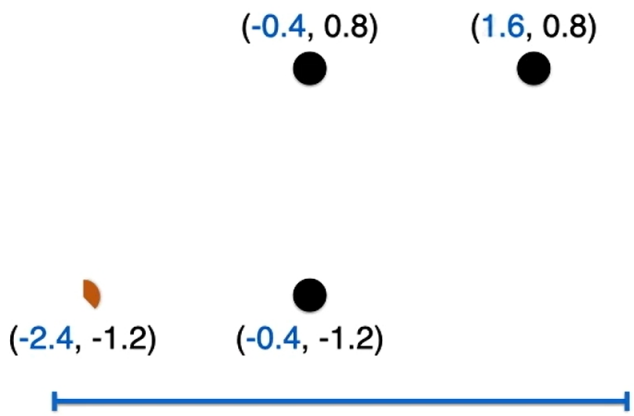
When points are weighted differently

Average



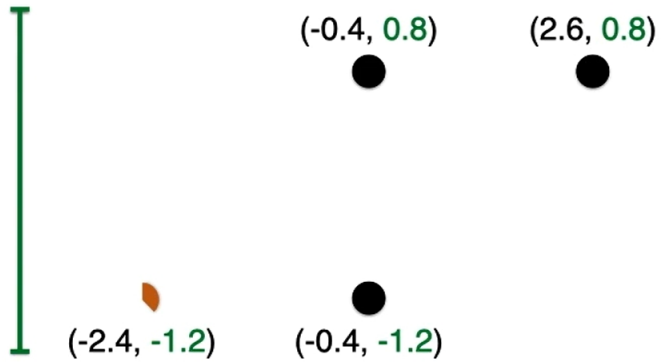
$$\left(\frac{\frac{1}{3}1+3+3+5}{\frac{1}{3}1+1+1+1}, \frac{\frac{1}{3}1+1+3+3}{\frac{1}{3}1+1+1+1}\right) = (3.4,2.2)$$

X-variance



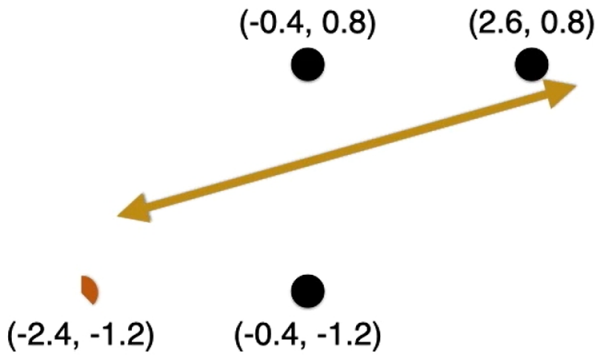
$$var(x) = \frac{1}{4}(\left(\frac{1}{3}\right)^2(-2.4)^2 + (-0.4)^2 + (-0.4)^2 + 1.6^2) = 0.88$$

Y-variance



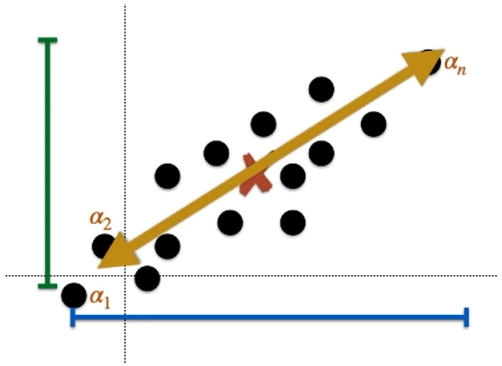
$$var(y) = \frac{1}{4}(\left(\frac{1}{3}\right)^2(-1.2)^2 + (-1.2)^2 + 0.8^2 + 0.8^2) = 0.72$$

Covariance



$$cov(x, y) = \frac{1}{4}(\left(\frac{1}{3}\right)^2(-2.4)(-1.2) + (-0.4)(-1.2) + (-0.4)(0.8) + (2.6)(0.8)) = 0.64$$

Formulas



Weights	Points
α_1	(x_1, y_1)
α_2	(x_2, y_2)
$\vdots$	$\vdots$
α_n	(x_n, y_n)

Mean

$$(\mu_x, \mu_y) = (\frac{1}{\sum \alpha_i} \sum_{i=1}^{n} \alpha_i \; x_i, \frac{1}{\sum \alpha_i} \sum_{i=1}^{n} \alpha_i \; y_i)$$

x-Variance

$$var(x) = \frac{1}{\sum \alpha_i^2} \sum_{i=1}^{n} \alpha_i^2 \, (x_i - \mu_x)^2$$

y-Variance

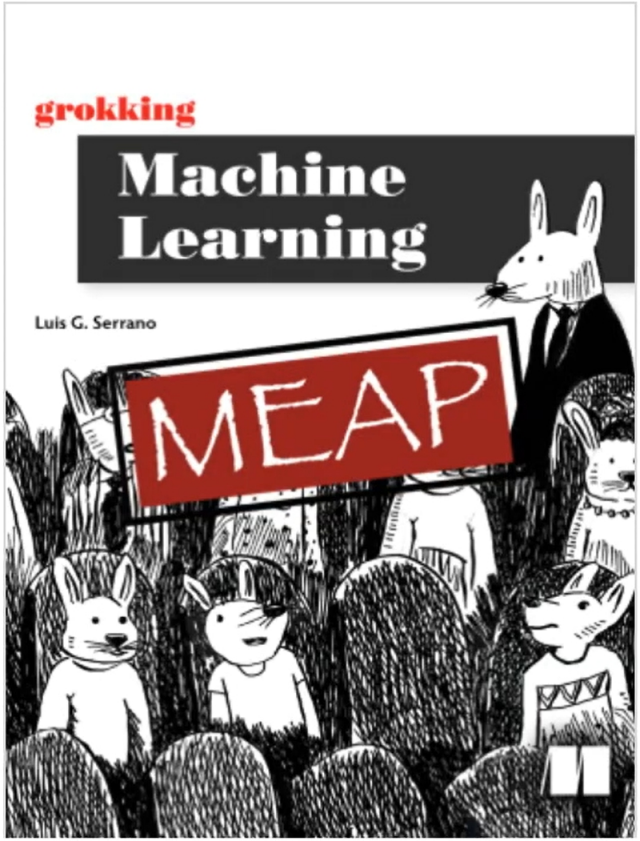
$$var(y) = \frac{1}{\sum \alpha_i^2} \sum_{i=1}^{n} \alpha_i^2 \, (y_i - \mu_y)^2$$

Covariance

$$cov(x, y) = \frac{1}{\sum \alpha_i^2} \sum_{i=1}^{n} \alpha_i^2 \, (x_i - \mu_x)(y_i - \mu_y)$$

Covariance matrix

$$\Sigma = \begin{pmatrix} var(x) & cov(x, y) \\ cov(x, y) & var(y) \end{pmatrix}$$



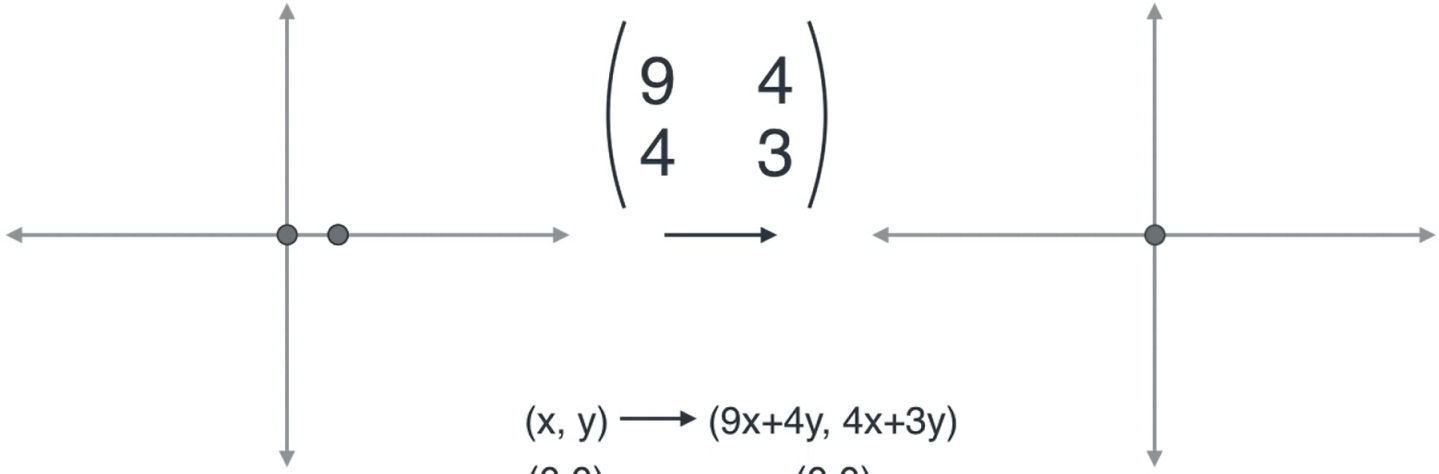
Grokking Machine Learning

By Luis G. Serrano

https://www.manning.com/books/grokking-machine-learning
Discount code: serranoyt

Linear Transformations

$$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$



$(x, y) \longrightarrow (9x+4y, 4x+3y)$

(0,0)	(0,0)
(1,0)	(9,4)

Eigenvalues and Eigenvectors

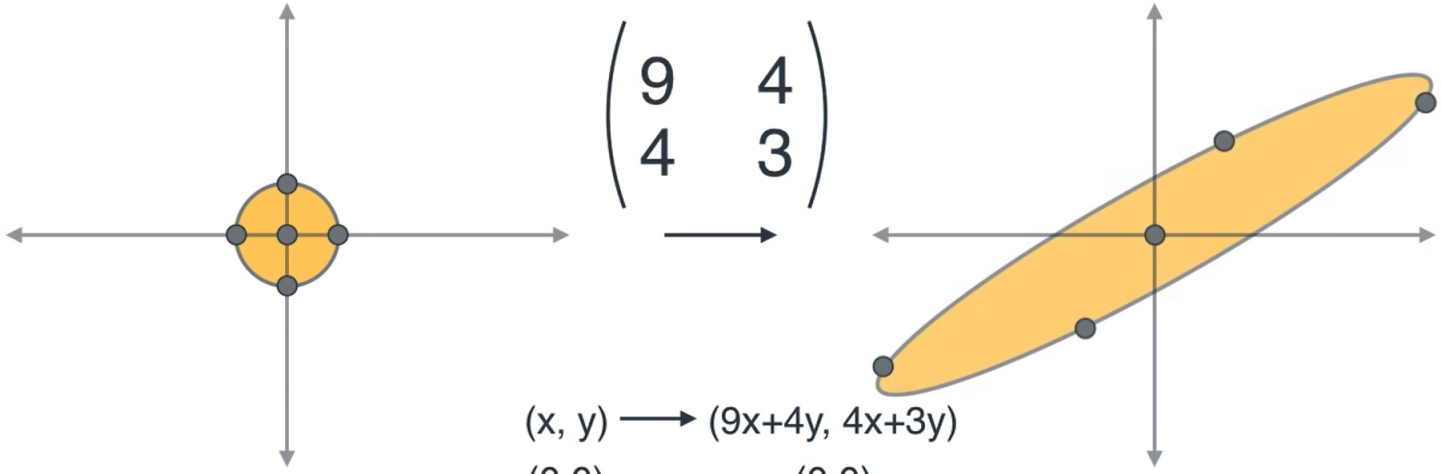
Principal components are the new set of variables that are obtained from the initial set of variables. They compress and possess most of the useful information that was scattered among the initial variables.

- *Eigenvectors are those vectors when a linear transformation is performed on them then their direction does not change.*
- *Eigenvalues simply denote the scalars of the respective eigenvectors*

Step 3: Calculating the Eigenvectors and Eigenvalues

Eigenvectors and eigenvalues are the mathematical constructs that must be computed from the covariance matrix in order to determine the principal components of the data set.

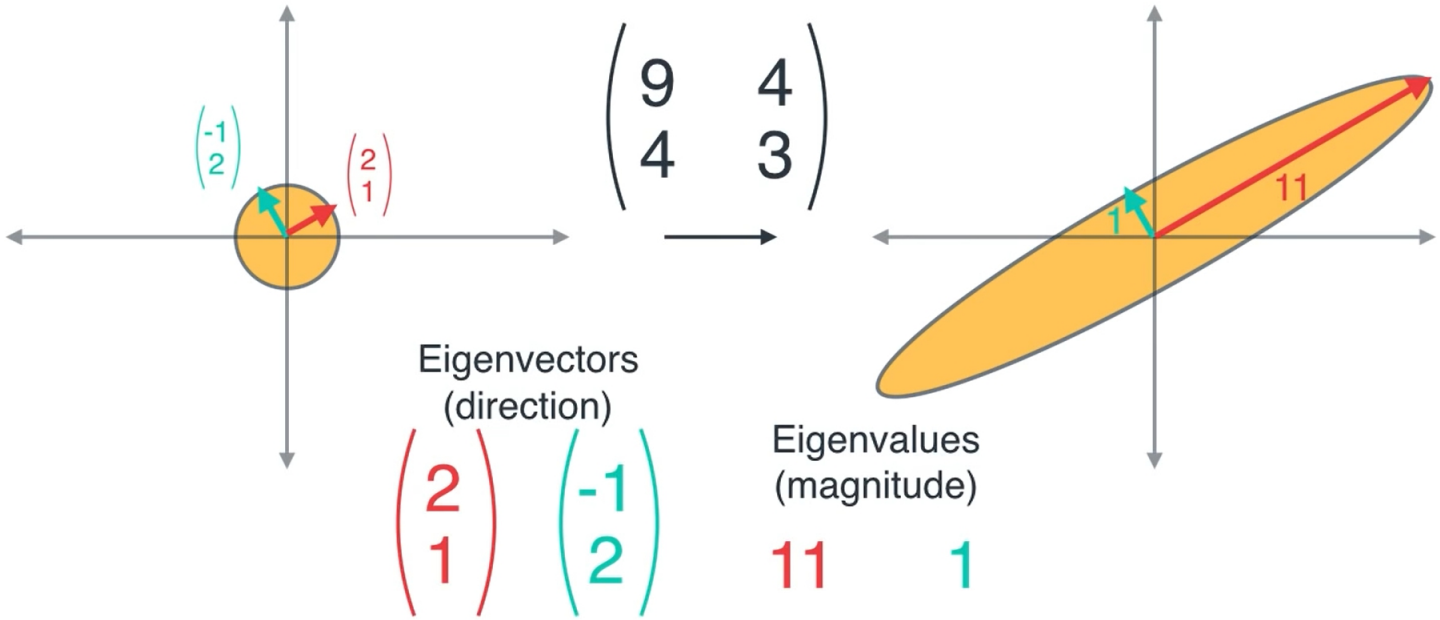
Linear Transformations



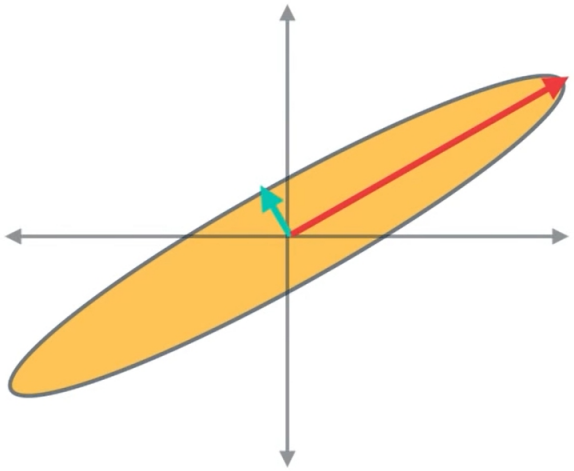
$(x, y) \rightarrow (9x+4y, 4x+3y)$

(0,0)	(0,0)
(1,0)	(9,4)
(0,1)	(4,3)
(-1,0)	(-9,-4)
(0,-1)	(-4,-3)

Linear Transformations



Linear Transformations



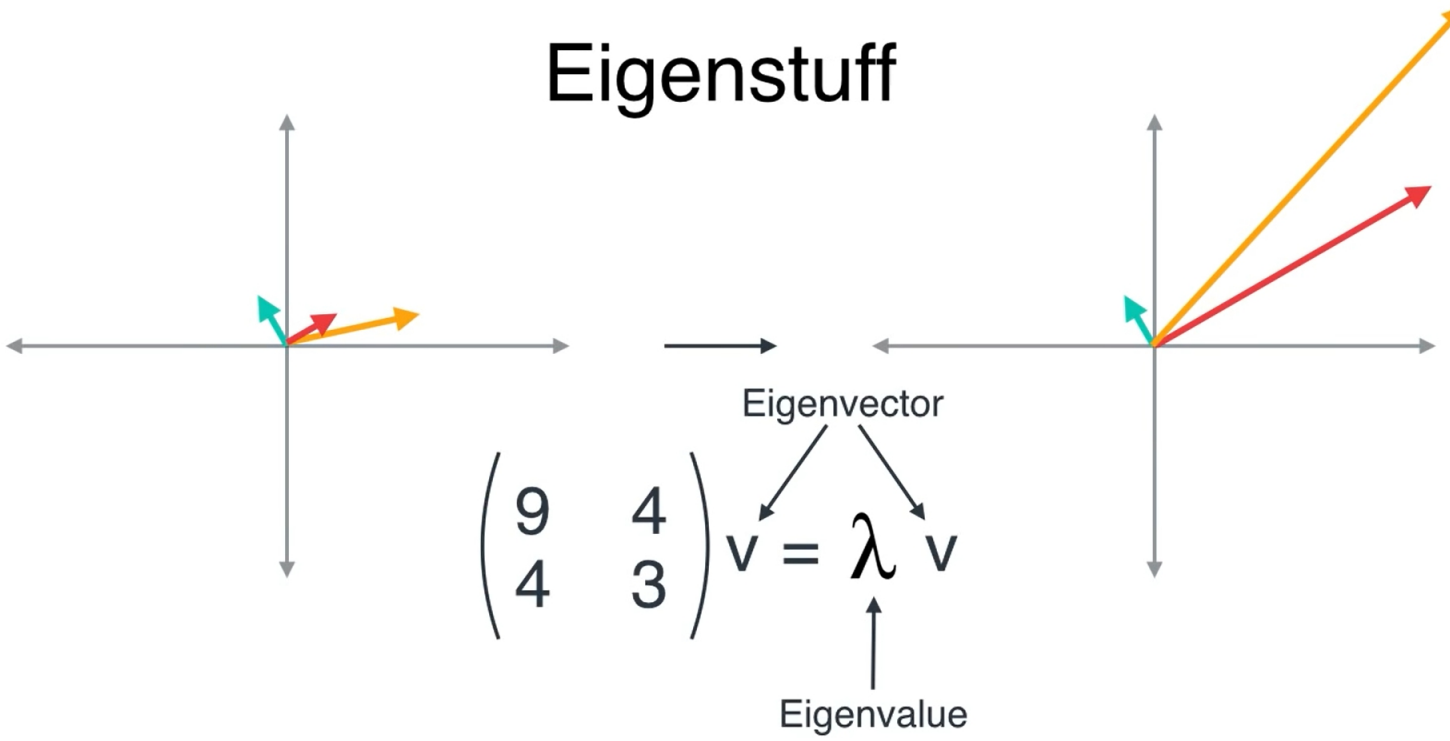
Eigenvectors
(direction)

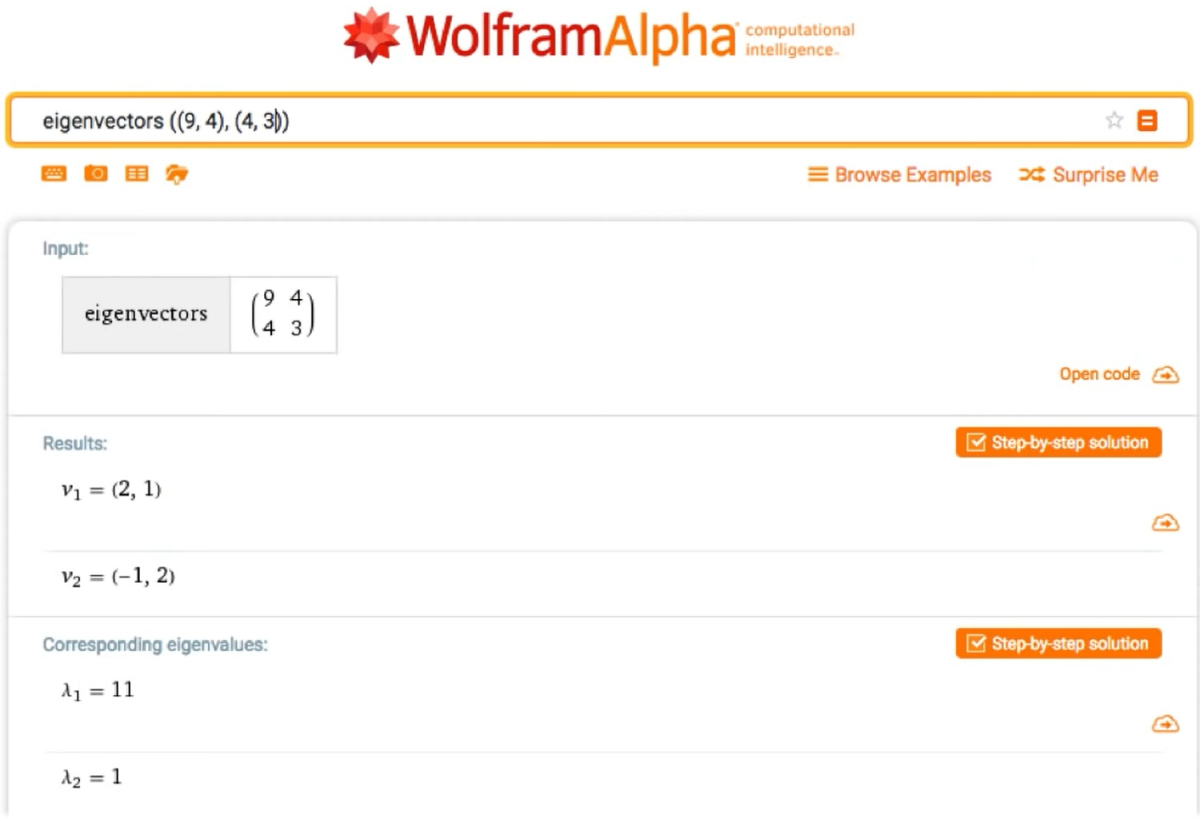
$\begin{pmatrix} 2 \\ 1 \end{pmatrix}$ $\begin{pmatrix} -1 \\ 2 \end{pmatrix}$

Eigenvalues
(magnitude)

11 1

Eigenstuff


Eigenvector
$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix} v = \lambda v$
Eigenvalue


WolframAlpha computational intelligence
eigenvectors ((9, 4), (4, 3))
Browse Examples
Surprise Me
Input:
eigenvectors
$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$
Open code
Results:
Step-by-step solution
$v_1 = (2, 1)$
$v_2 = (-1, 2)$
Corresponding eigenvalues:
Step-by-step solution
$\lambda_1 = 11$
$\lambda_2 = 1$

Eigenvalues

$$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

Characteristic Polynomial

$$\begin{vmatrix} x-9 & -4 \\ -4 & x-3 \end{vmatrix} = (x-9)(x-3) - (-4)(-4) = x^2 - 12x + 11$$

$$= (x-11)(x-1)$$

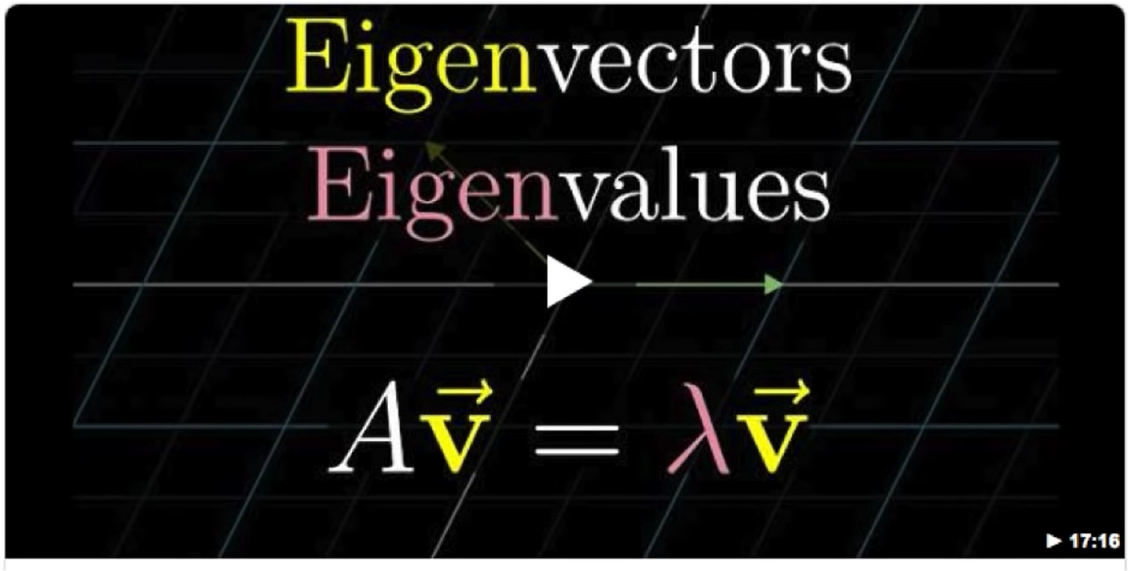
Eigenvalues 11 and 1

Eigenvectors

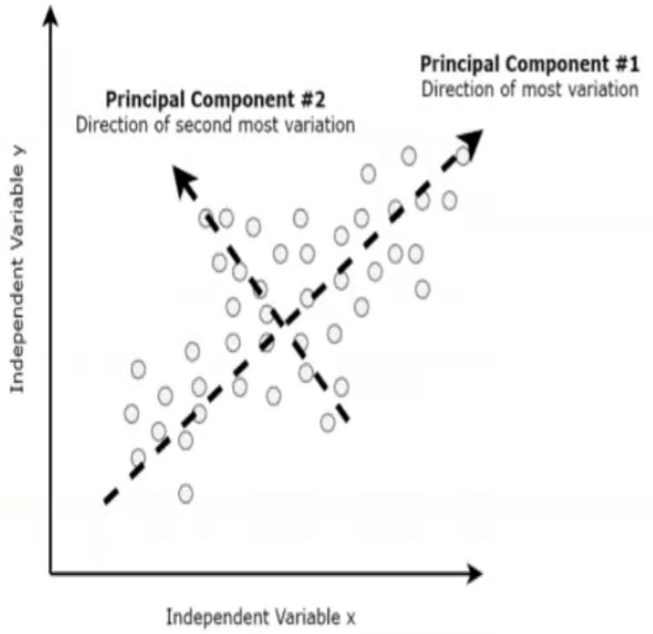
$$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = 11\begin{pmatrix} u \\ v \end{pmatrix} \qquad \begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = 1\begin{pmatrix} u \\ v \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

3blue1brown

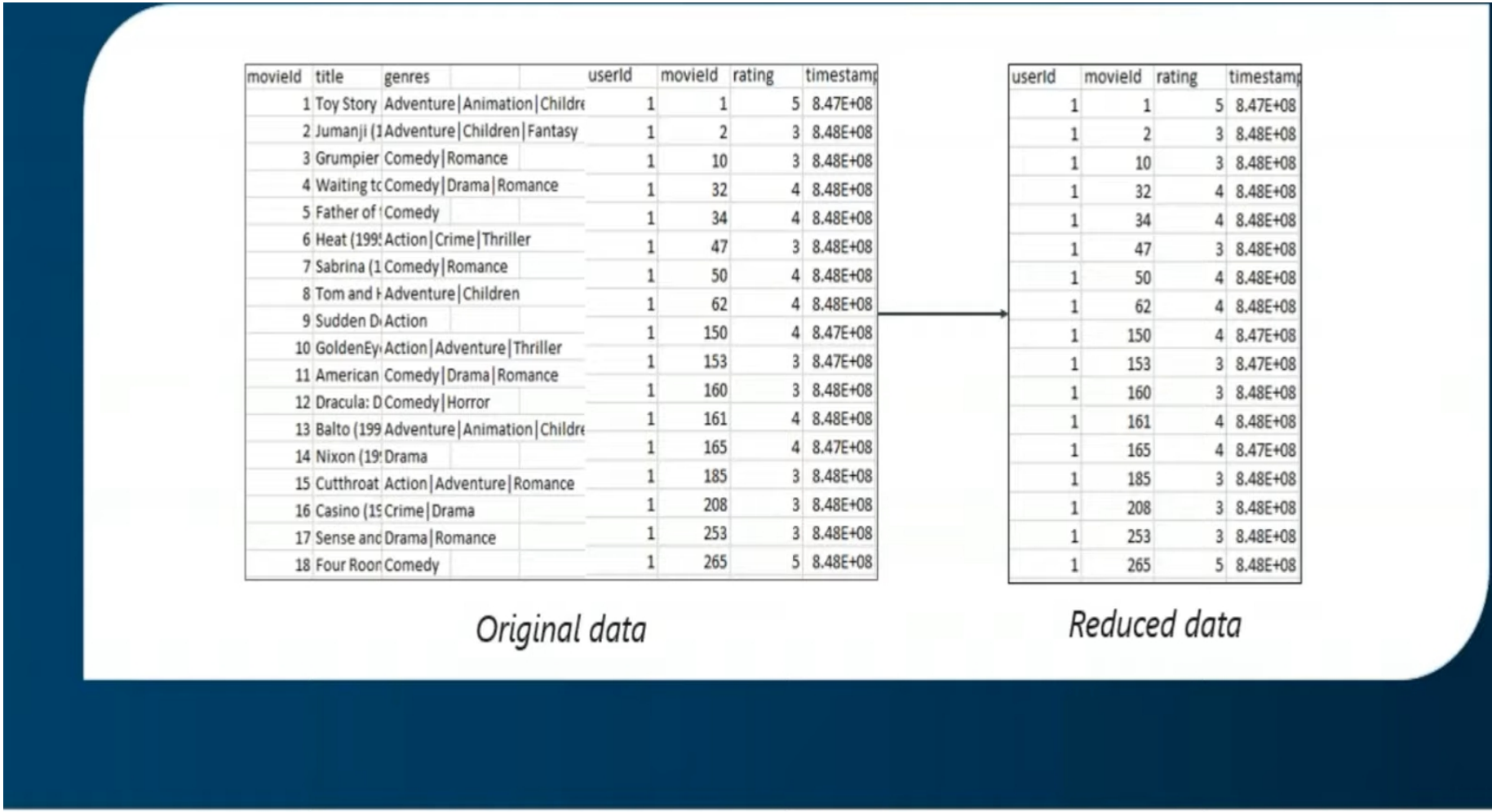


- *PC1 is the most significant and stores the maximum possible information.*
- *PC2 is the second most significant PC and stores remaining maximum info and so on*



Step 4: Computing the Principal Components

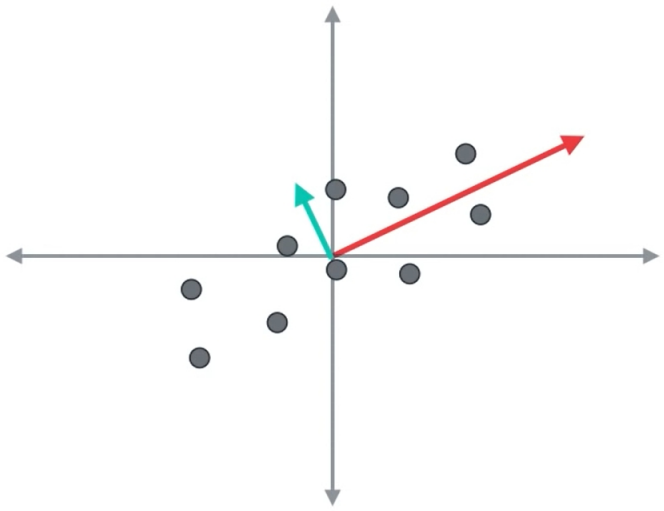
Once we have computed the Eigenvectors and eigenvalues, all we have to do is order them in the descending order, where the eigenvector with the highest eigenvalue is the most significant and thus forms the first principal component.



Step 5: Reducing the dimensions of the data set

The last step in performing PCA is to re-arrange the original data with the final principal components which represent the maximum and the most significant information of the data set.

Principal Component Analysis (PCA)



$$\Sigma = \begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

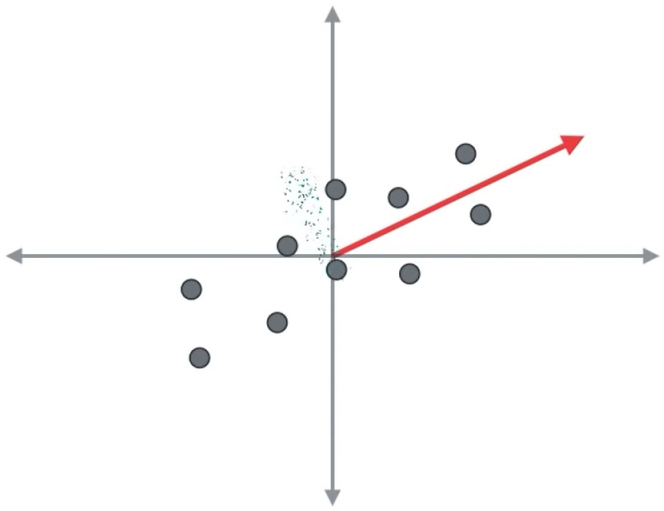
$$\begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

Eigenvectors (direction)

11 1

Eigenvalues (magnitude)

Principal Component Analysis (PCA)



$$\Sigma = \begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

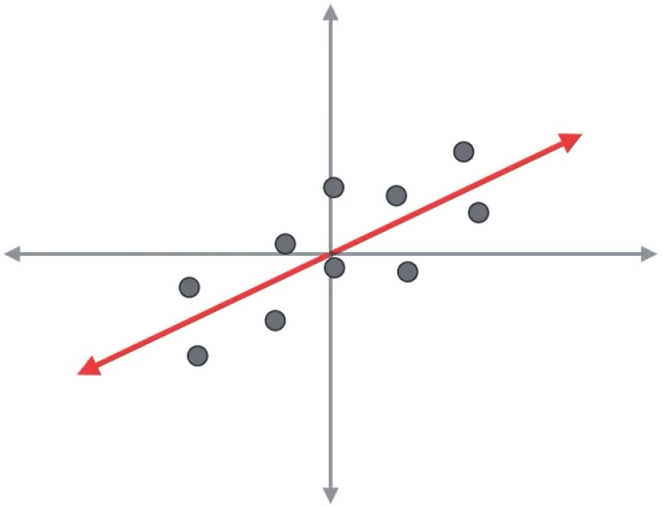
$$\begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Eigenvectors
(direction)

11

Eigenvalues
(magnitude)

Principal Component Analysis (PCA)



$$\Sigma = \begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

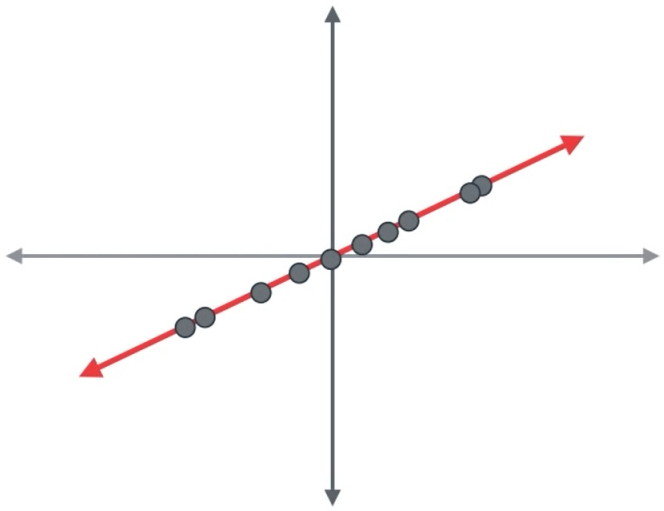
$$\begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Eigenvectors
(direction)

11

Eigenvalues
(magnitude)

Principal Component Analysis (PCA)



$$\Sigma = \begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

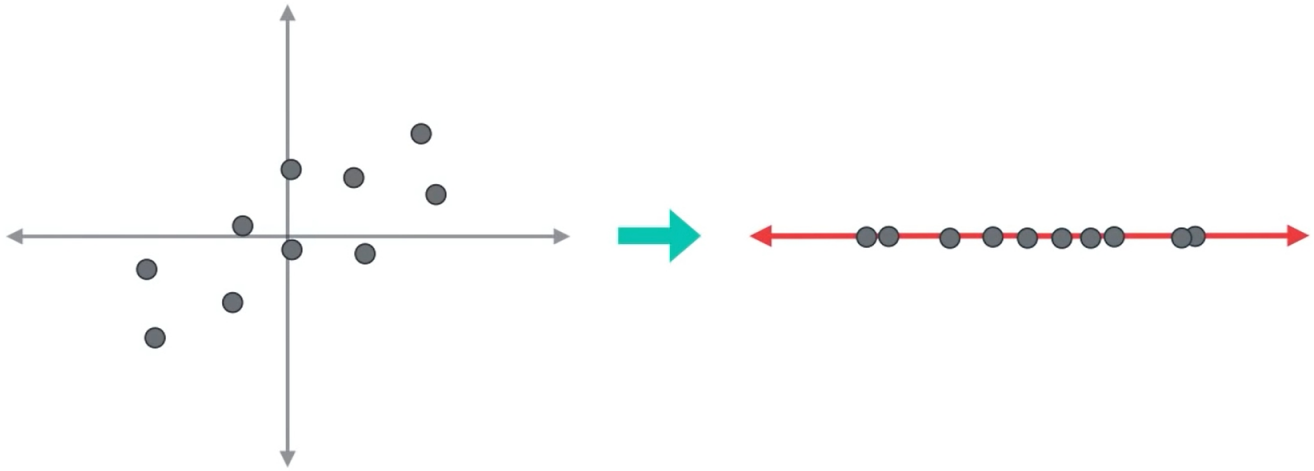
$$\begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

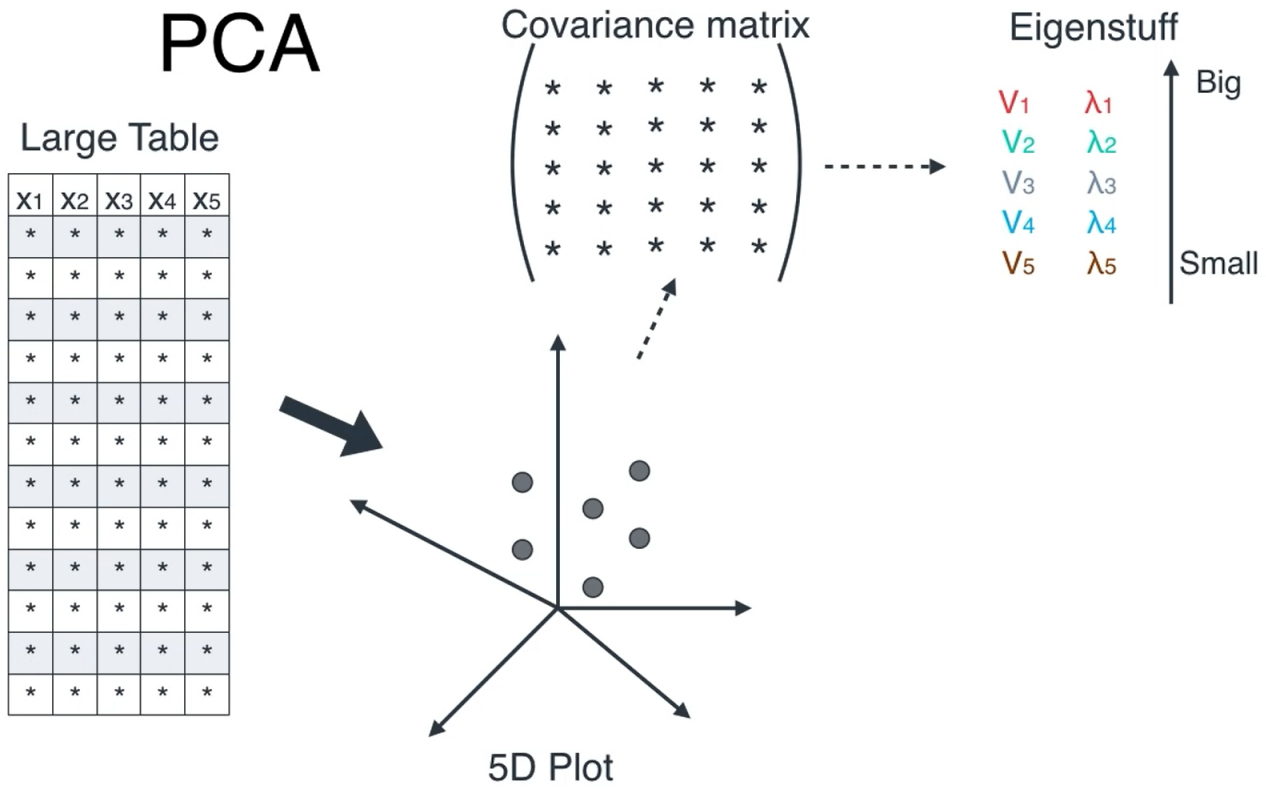
Eigenvectors (direction)

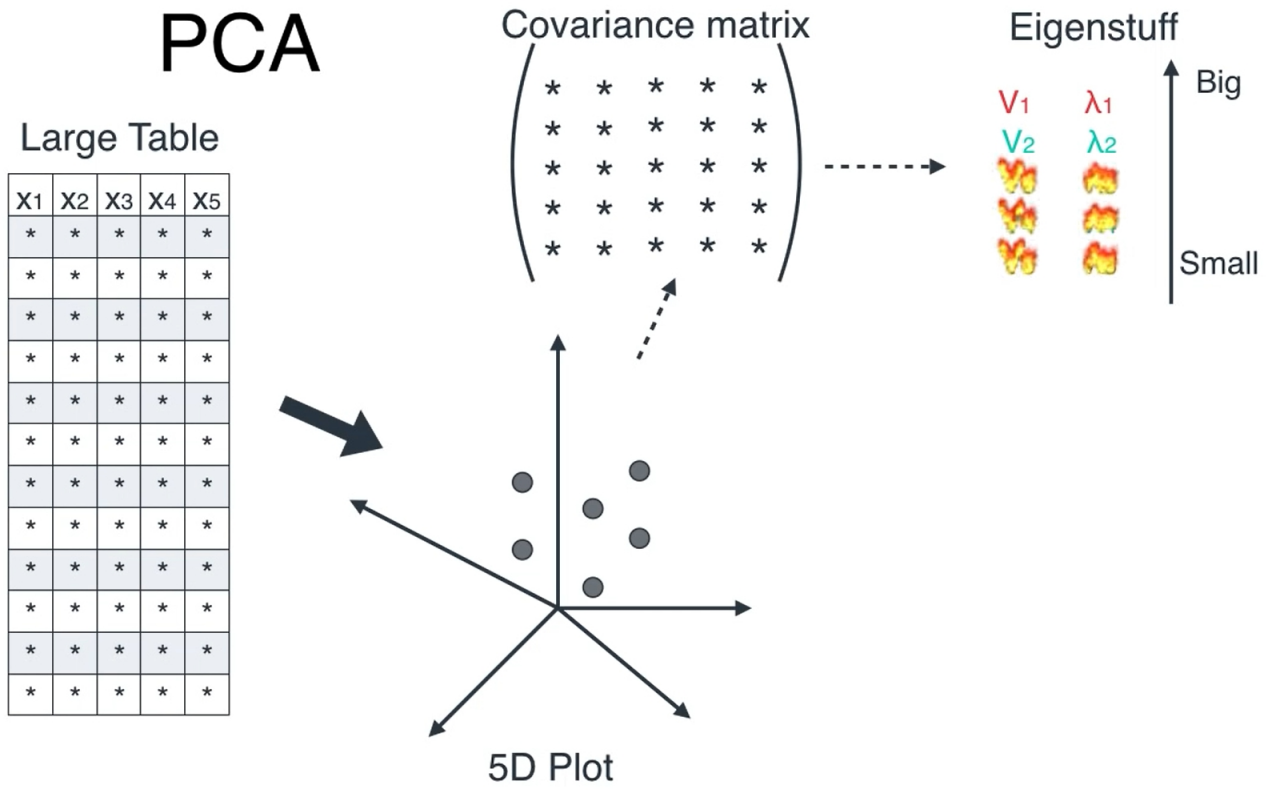
11

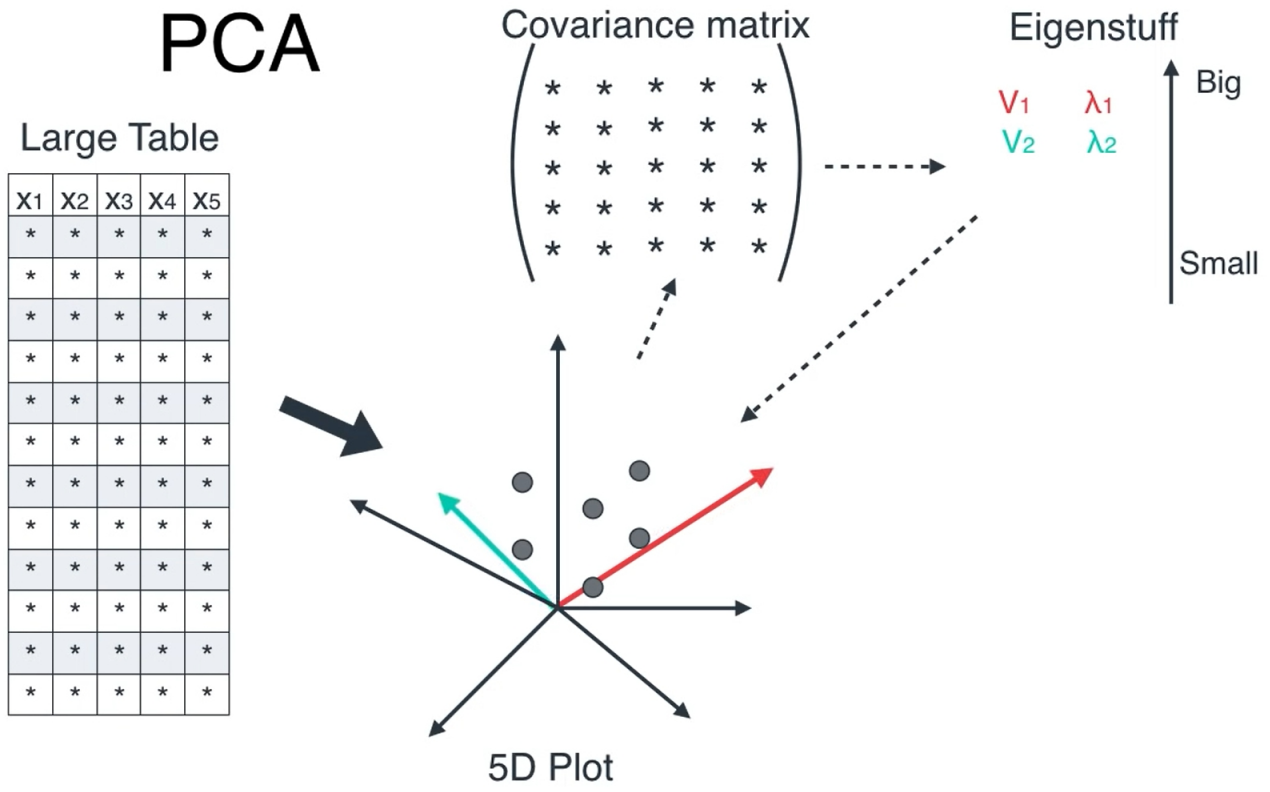
Eigenvalues (magnitude)

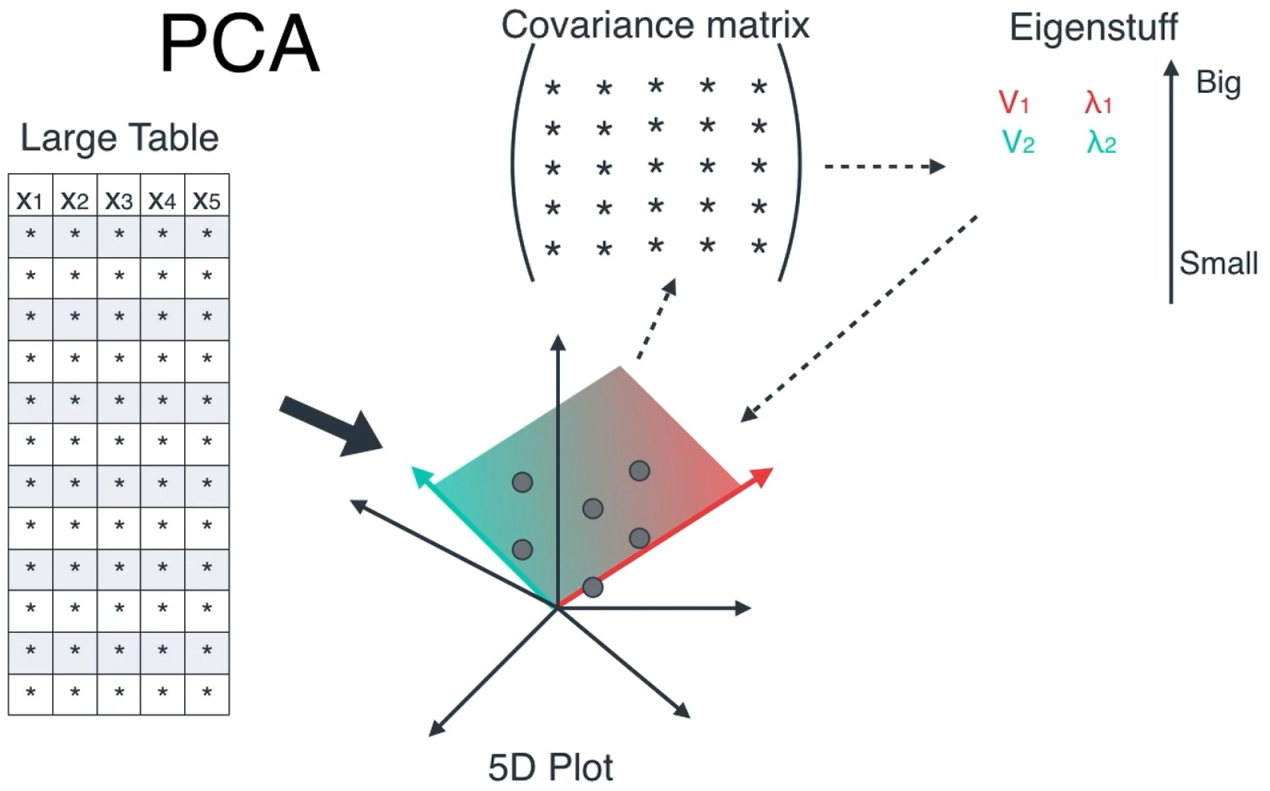
Principal Component Analysis (PCA)

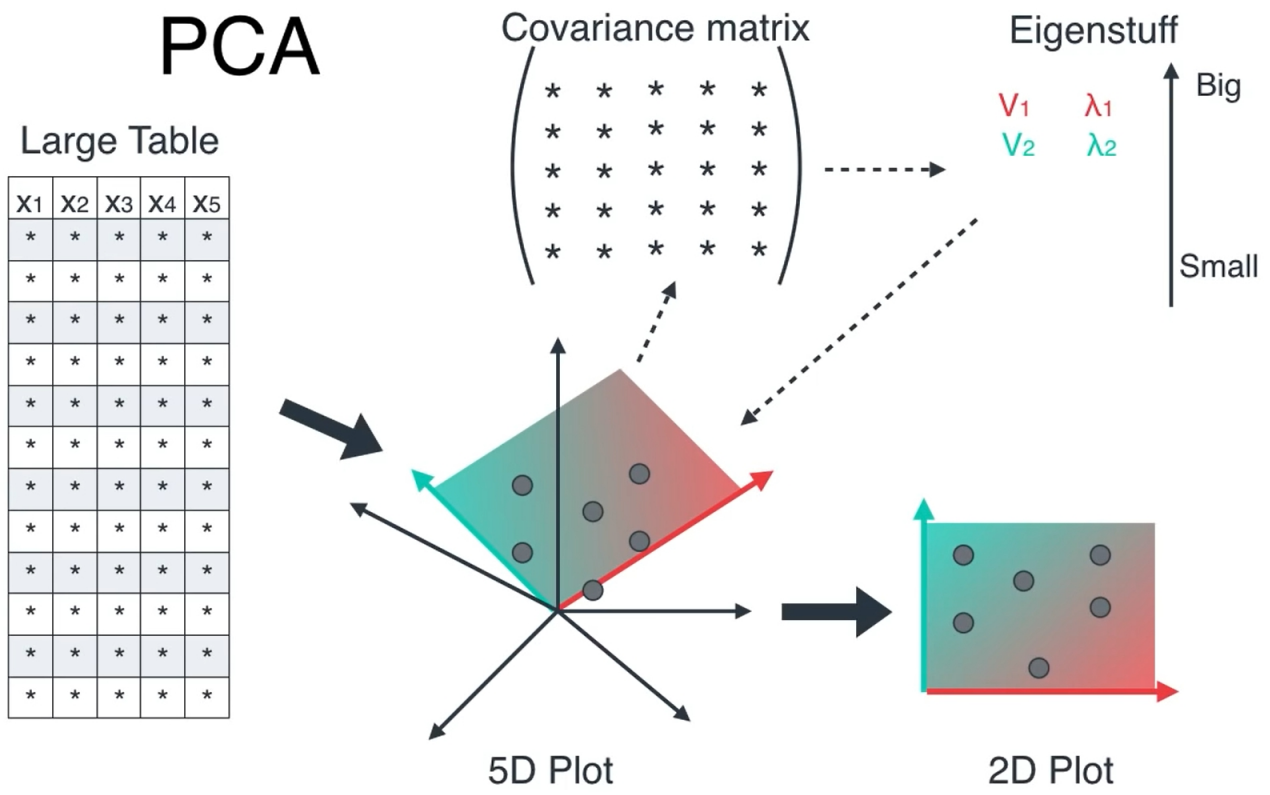


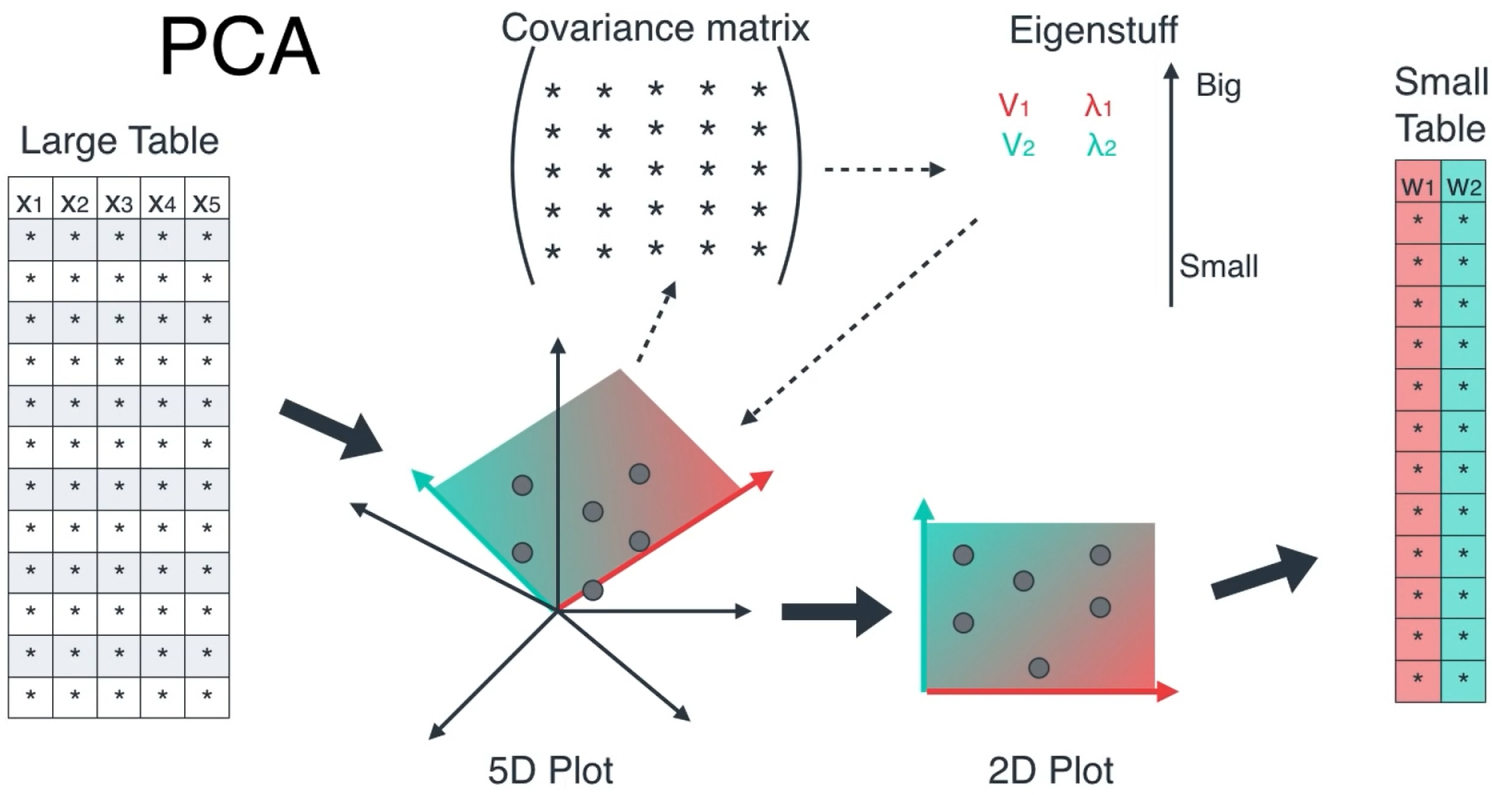

PCA
Large Table
X1 X2 X3 X4 X5
Covariance matrix
Eigenstuff
V1 λ1
V2 λ2
V3 λ3
V4 λ4
V5 λ5
Big
Small
5D Plot


PCA
Large Table
X1 X2 X3 X4 X5
Covariance matrix
Eigenstuff
V1 λ1
V2 λ2
Big
Small
5D Plot


PCA
Large Table
X1 X2 X3 X4 X5
Covariance matrix
Eigenstuff
V1 λ1
V2 λ2
Big
Small
5D Plot


PCA
Large Table
X1 X2 X3 X4 X5
Covariance matrix
Eigenstuff
V1 λ1
V2 λ2
Big
Small
5D Plot


PCA
Large Table
X1 X2 X3 X4 X5
Covariance matrix
Eigenstuff
V1 λ1
V2 λ2
Big
Small
5D Plot
2D Plot


PCA
Large Table
X1 X2 X3 X4 X5
Covariance matrix
Eigenstuff
V1 λ1
V2 λ2
Big
Small
Small Table
W1 W2
5D Plot
2D Plot

Data set Description: Movies rating data set that contains ratings from 700+ users for approximately 9000 movies (features).

movieId	title	genres	userId	movieId	rating	timestamp
1	Toy Story	Adventure|Animation|Childre	1	1	5	8.47E+08
2	Jumanji (1	Adventure|Children|Fantasy	1	2	3	8.48E+08
3	Grumpier	Comedy|Romance	1	10	3	8.48E+08
4	Waiting tc	Comedy|Drama|Romance	1	32	4	8.48E+08
5	Father of	Comedy	1	34	4	8.48E+08
6	Heat (199	Action|Crime|Thriller	1	47	3	8.48E+08
7	Sabrina (1	Comedy|Romance	1	50	4	8.48E+08
8	Tom and H	Adventure|Children	1	62	4	8.48E+08
9	Sudden D	Action	1	150	4	8.47E+08
10	GoldenEy	Action|Adventure|Thriller	1	153	3	8.47E+08
11	American	Comedy|Drama|Romance	1	160	3	8.48E+08
12	Dracula: D	Comedy|Horror	1	161	4	8.48E+08
13	Balto (199	Adventure|Animation|Childre	1	165	4	8.47E+08
14	Nixon (19	Drama	1	185	3	8.48E+08
15	Cutthroat	Action|Adventure|Romance	1	208	3	8.48E+08
16	Casino (19	Crime|Drama	1	253	3	8.48E+08
17	Sense and	Drama|Romance	1	265	5	8.48E+08
18	Four Roon	Comedy				

Principal Component Analysis (PCA) In Python

Problem Statement: To perform step by step Principal Component Analysis on movie review data set in order to reduce the dimensions of the data set.

File Edit View Navigate Code Refactor Run Tools VCS Window Help

C: Users NeelTemp Desktop pca.py

pca

scratch_9.py pca.py

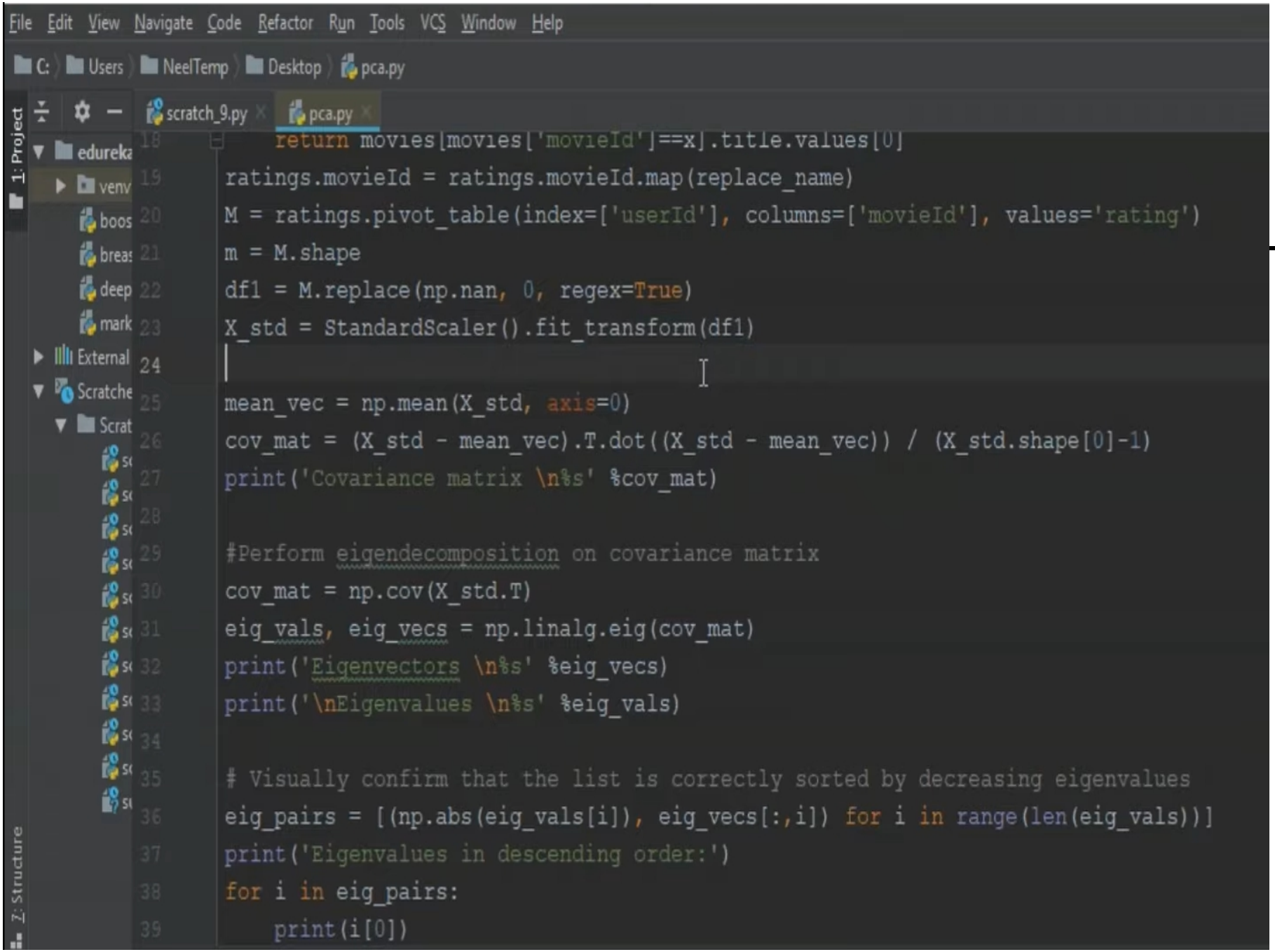
1: Project

edureka
venv
boos
breas
deep
mark
External
Scratche
Scrat

```
#Load dependencies
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from matplotlib import*
import matplotlib.pyplot as plt
from matplotlib.cm import register_cmap
from scipy import stats
from sklearn.decomposition import PCA as sklearnPCA
import seaborn

#Load movie names and movie ratings
movies = pd.read_csv('C:\\Users\\NeelTemp\\Desktop\\PCA DATA\\movies.csv')
ratings = pd.read_csv('C:\\Users\\NeelTemp\\Desktop\\PCA DATA\\ratings.csv')
ratings.drop(['timestamp'], axis=1, inplace=True)

def replace_name(x):
    return movies[movies['movieId']==x].title.values[0]
ratings.movieId = ratings.movieId.map(replace_name)
M = ratings.pivot_table(index=['userId'], columns=['movieId'], values='rating')
m = M.shape
df1 = M.replace(np.nan, 0, regex=True)
X_std = StandardScaler().fit_transform(df1)
```

7: Structure

vorites


File Edit View Navigate Code Refactor Run Tools VCS Window Help
C: Users NeelTemp Desktop pca.py
scratch_9.py
pca.py
1: Project
7: Structure
edureka
venv
External
Scratche
return movies[movies['movieId']==x].title.values[0]
ratings.movieId = ratings.movieId.map(replace_name)
M = ratings.pivot_table(index=['userId'], columns=['movieId'], values='rating')
m = M.shape
df1 = M.replace(np.nan, 0, regex=True)
X_std = StandardScaler().fit_transform(df1)
mean_vec = np.mean(X_std, axis=0)
cov_mat = (X_std - mean_vec).T.dot((X_std - mean_vec)) / (X_std.shape[0]-1)
print('Covariance matrix \n%s' %cov_mat)
#Perform eigendecomposition on covariance matrix
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
# Visually confirm that the list is correctly sorted by decreasing eigenvalues
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]
print('Eigenvalues in descending order:')
for i in eig_pairs:
print(i[0])

File Edit View Navigate Code Refactor Run Tools VCS Window Help

C: Users NeelTemp Desktop pca.py

scratch_9.py pca.py

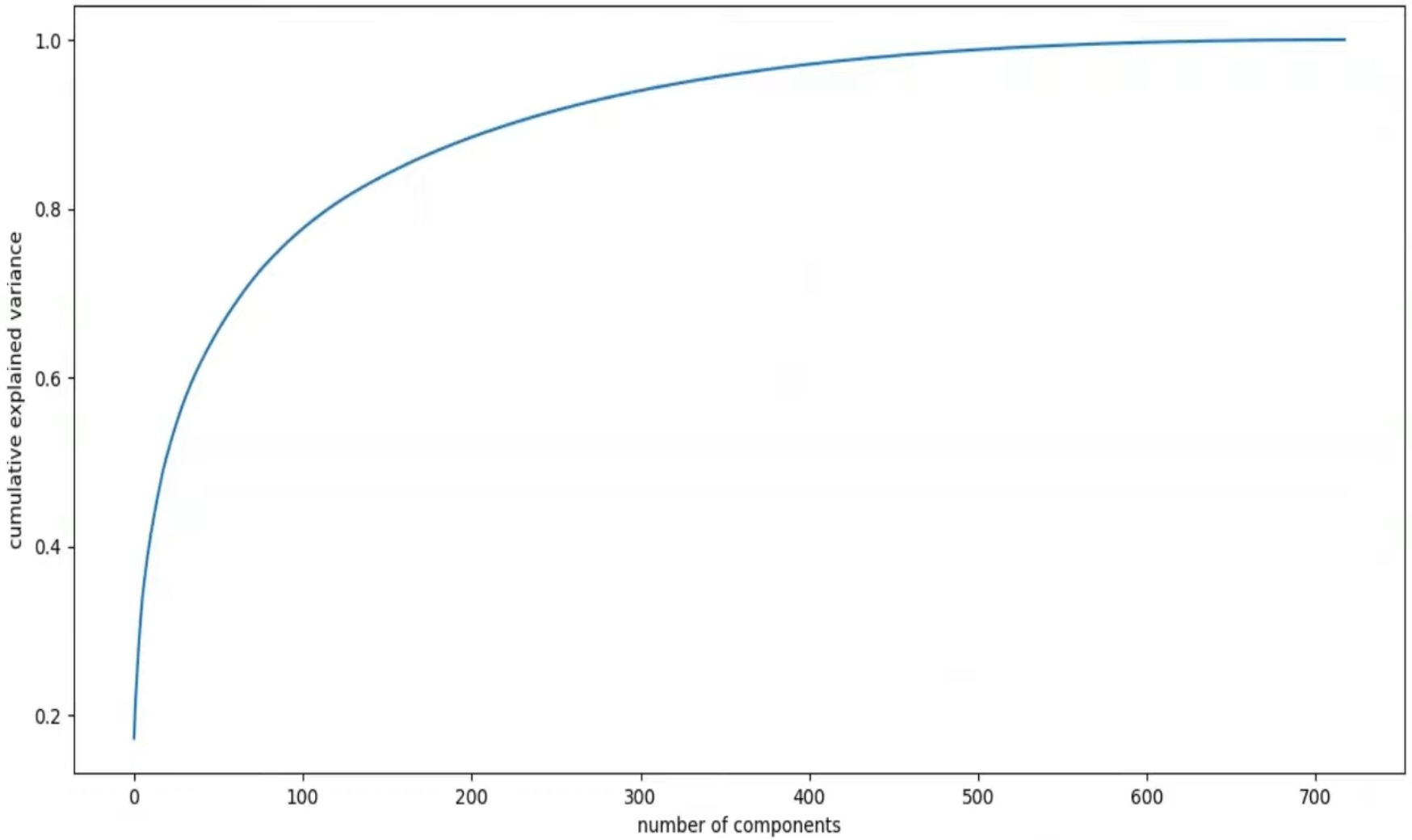
```
print(i[0])

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca.fit_transform(df1)
print(pca.explained_variance_ratio_)

#Explained variance
pca = PCA().fit(X_std)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```

Run: pca

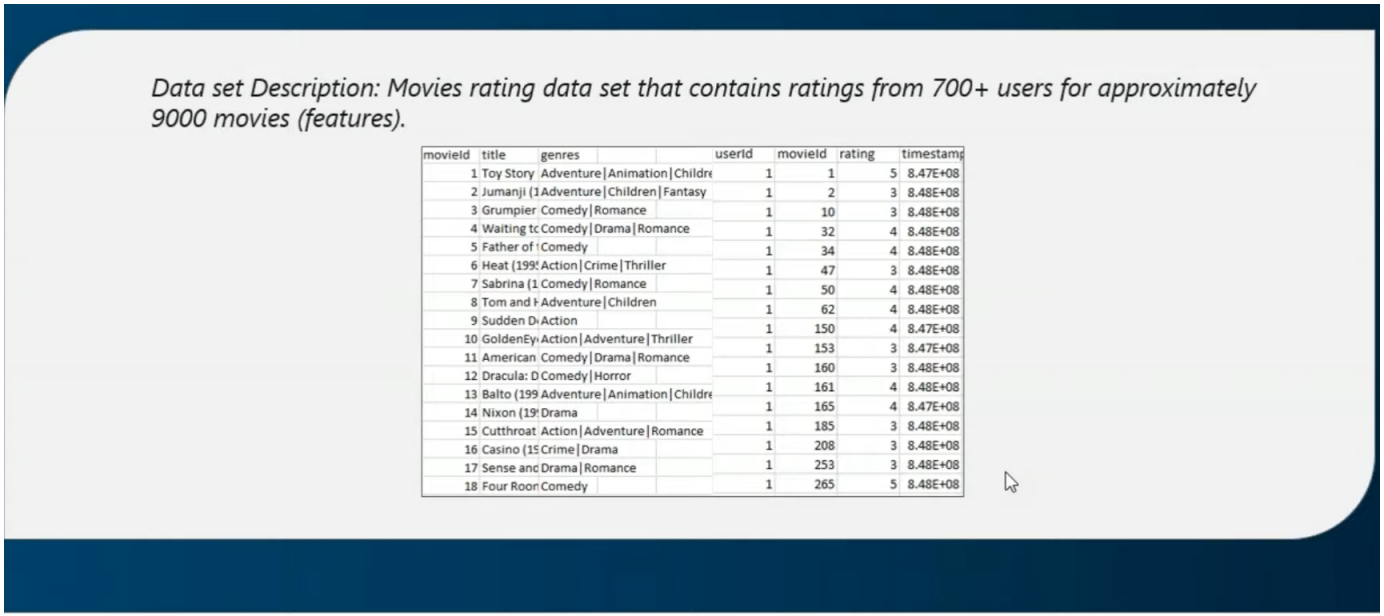
```
[-0.00321221 -0.0032393  -0.0022929  ...  0.02888777  1.0013947
  0.01676203]
[-0.01055463 -0.01064364 -0.00753398 ...  0.14005644  0.01676203
  1.0013947 ]]
```

4: Run 6: TODO Terminal Python Console

IDE and Plugin Updates: PyCharm is ready to update. (11 minutes ago)


1.0
0.8
0.6
0.4
0.2
cumulative explained variance
0
100
200
300
400
500
600
700
number of components

Dimension reduced from 9000 to 500



Data set Description: Movies rating data set that contains ratings from 700+ users for approximately 9000 movies (features).

movieId	title	genres	userId	movieId	rating	timestamp
1	Toy Story	Adventure|Animation|Childre	1	1	5	8.47E+08
2	Jumanji (1	Adventure|Children|Fantasy	1	2	3	8.48E+08
3	Grumpier	Comedy|Romance	1	10	3	8.48E+08
4	Waiting to	Comedy|Drama|Romance	1	32	4	8.48E+08
5	Father of	Comedy	1	34	4	8.48E+08
6	Heat (199	Action|Crime|Thriller	1	47	3	8.48E+08
7	Sabrina (1	Comedy|Romance	1	50	4	8.48E+08
8	Tom and H	Adventure|Children	1	62	4	8.48E+08
9	Sudden De	Action	1	150	4	8.47E+08
10	GoldenEye	Action|Adventure|Thriller	1	153	3	8.47E+08
11	American	Comedy|Drama|Romance	1	160	3	8.48E+08
12	Dracula: D	Comedy|Horror	1	161	4	8.48E+08
13	Balto (199	Adventure|Animation|Childre	1	165	4	8.47E+08
14	Nixon (19	Drama	1	185	3	8.48E+08
15	Cutthroat	Action|Adventure|Romance	1	208	3	8.48E+08
16	Casino (19	Crime|Drama	1	253	3	8.48E+08
17	Sense and	Drama|Romance	1	265	5	8.48E+08
18	Four Roon	Comedy				

Principal Component Analysis (PCA) In Python

Problem Statement: To perform step by step Principal Component Analysis on movie review data set in order to reduce the dimensions of the data set.

localhost:8888/notebooks/Py-DS-ML-Bootcamp-master/Refactored_Py_DS_ML_Bootcamp-master/18-Principal-Component-Analysis/01-Principal%20...

Singam - Kadhal Van | www.upsc.gov.in/exa | Contact Us - How To | Traffic Domination — | CP-006 Video - 1475 | Search - Visual Studi | How to Customize Pe | Top 10 Algorithms a

jupyter 01-Principal Component Analysis Last Checkpoint: Last Wednesday at 9:28 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

In [21]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
```

The Data

Let's work with the cancer data set again since it had so many features.

In [22]:
```
from sklearn.datasets import load_breast_cancer
```

In [23]:
```
cancer = load_breast_cancer()
```

In [24]:
```
cancer.keys()
```

Out[24]:
```
dict_keys(['DESCR', 'data', 'feature_names', 'target_names', 'target'])
```

In [25]:
```
print(cancer['DESCR'])
```

```
Breast Cancer Wisconsin (Diagnostic) Database

Notes
-----
Data Set Characteristics:
```

localhost:8888/notebooks/Py-DS-ML-Bootcamp-master/Refactored_Py_DS_ML_Bootcamp-master/18-Principal-Component-Analysis/01-Principal%20...

jupyter 01-Principal Component Analysis Last Checkpoint: Last Wednesday at 9:28 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

In [33]:
```
scaled_data = scaler.transform(df)
```

PCA with Scikit Learn uses a very similar process to other preprocessing functions that come with SciKit Learn. We instantiate a PCA object, find the principal components using the fit method, then apply the rotation and dimensionality reduction by calling transform().

We can also specify how many components we want to keep when creating the PCA object.

In [34]:
```
from sklearn.decomposition import PCA
```

In [35]:
```
pca = PCA(n_components=2)
```

30 features into 2 features

In [36]:
```
pca.fit(scaled_data)
```

Out[36]:
```
PCA(copy=True, n_components=2, whiten=False)
```

3 features

Now we can transform this data to its first 2 principal components.

In [37]:
```
x_pca = pca.transform(scaled_data)
```

In [38]:
```
scaled_data.shape
```

Out[38]:
```
(569, 30)
```

In [36]:
```
pca.fit(scaled_data)
```

Out[36]:
```
PCA(copy=True, n_components=2, whiten=False)
```

Now we can transform this data to its first 2 principal components.

In [37]:
```
x_pca = pca.transform(scaled_data)
```

In [38]:
```
scaled_data.shape
```

Out[38]:
```
(569, 30)
```

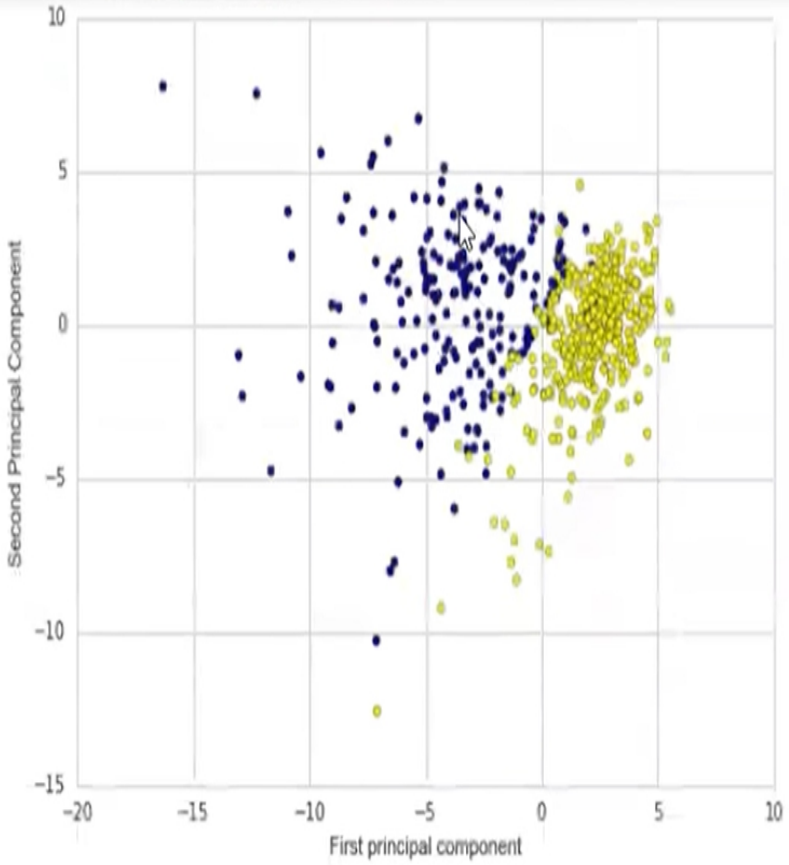
In [39]:
```
x_pca.shape
```

Out[39]:
```
(569, 2)
```

Great! We've reduced 30 dimensions to just 2! Let's plot these two dimensions out!

In [52]:
```
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=cancer['target'],cmap='plasma')
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
```

Out[52]:
```
<matplotlib.text.Text at 0x11eb56908>
```



Clearly by using these two components we can easily separate these two classes.

Refrences:

Refrences:

[Steven M. Holland, Univ. of Georgia]: Principal Components Analysis

[skymind.ai]: Eigenvectors, Eigenvalues, PCA, Covariance and Entropy

[Lindsay I. Smith] : A tutorial on Principal Component Analysis