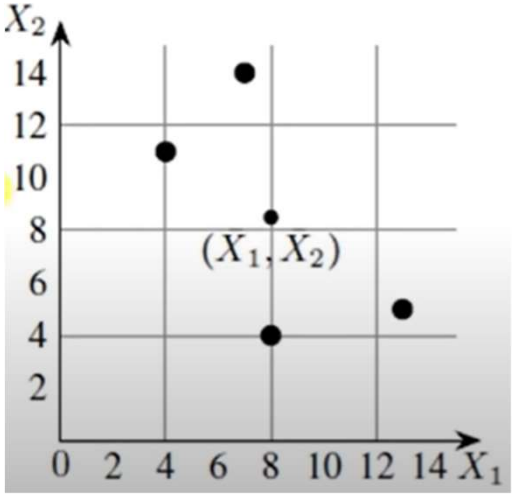
Data Reduction

PCA

Numerical Example Cntd...



$$e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

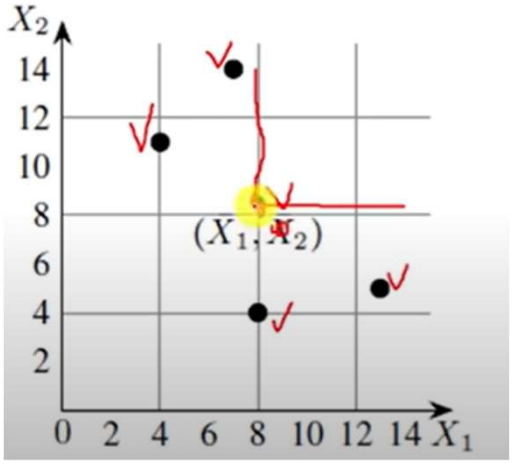
$$\overline{X_1} = 8$$

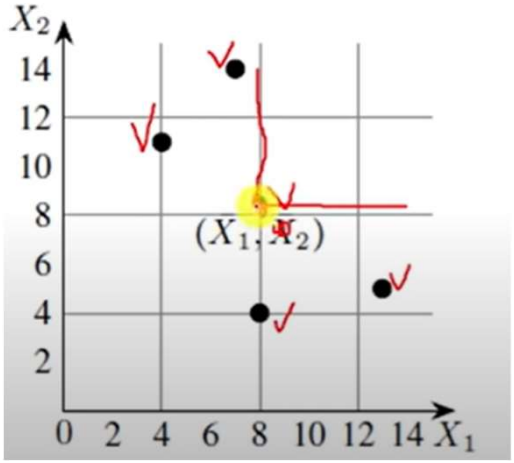
$$\overline{X_2} = 8.5$$

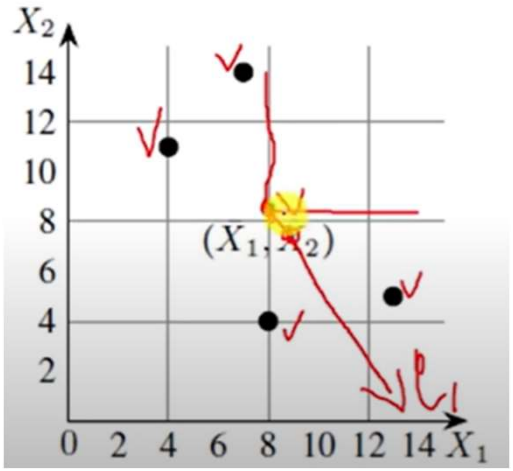
$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix} \quad S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

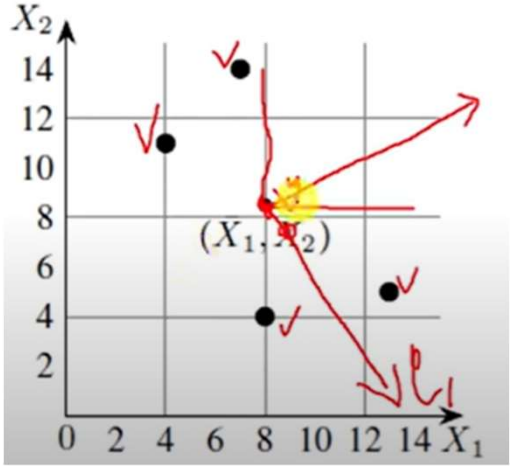
$$\lambda_1 = 30.3849$$

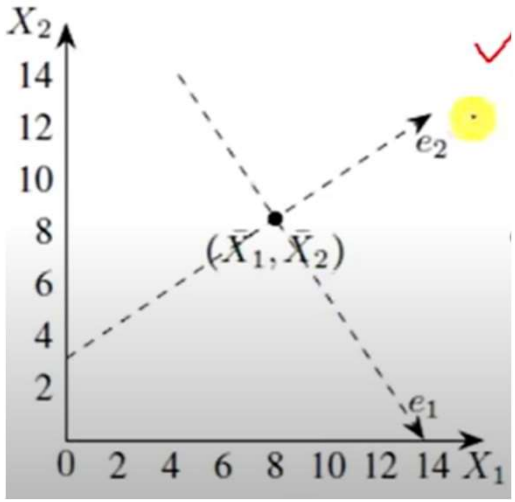
$$\lambda_2 = 6.6151$$

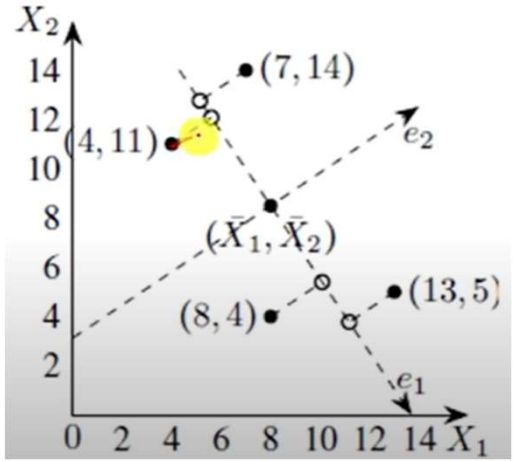



X_2
14
12
10
8
6
4
2
0 2 4 6 8 10 12 14 X_1
(X_1, X_2)


X_2
14
12
10
8
6
4
2
0 2 4 6 8 10 12 14 X_1
(X_1, X_2)


X_2
14
12
10
8
6
4
2
0 2 4 6 8 10 12 14 X_1
(X_1, X_2)

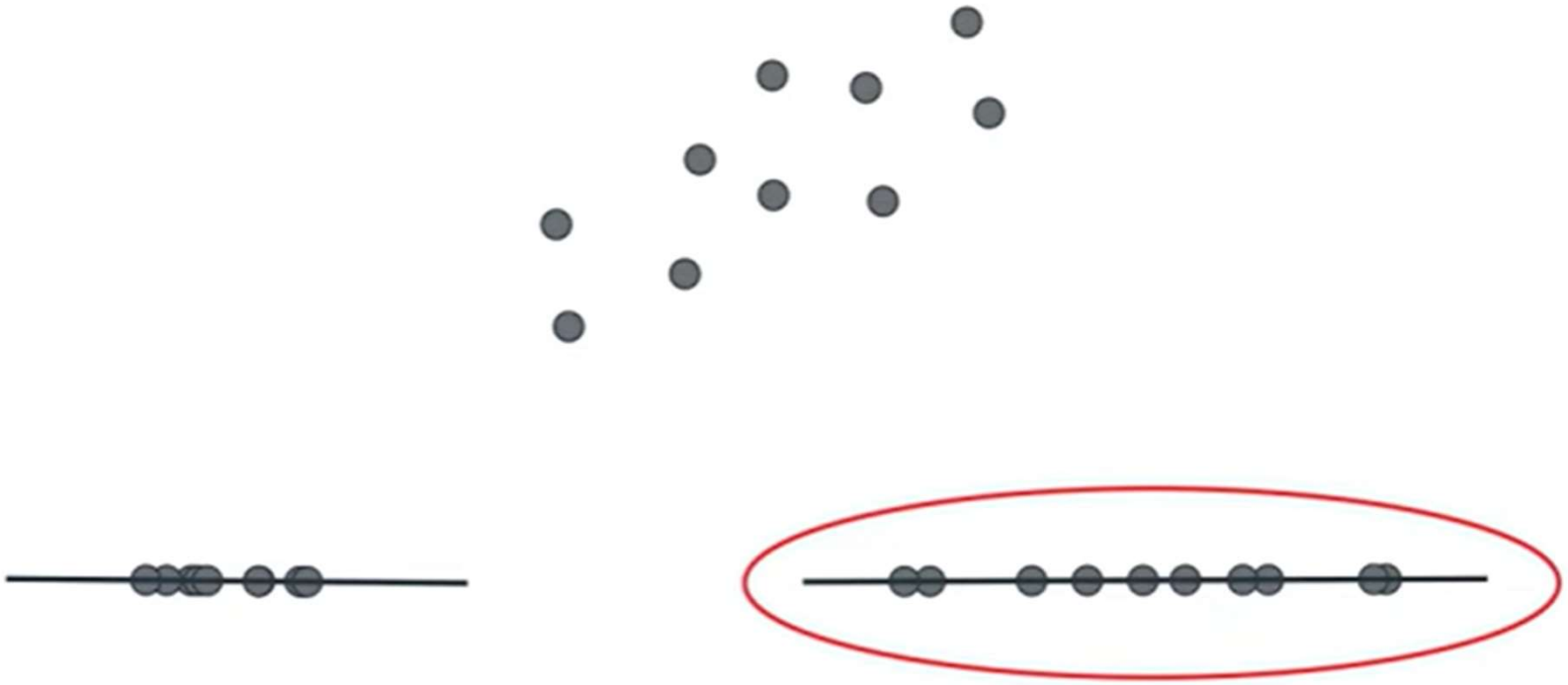

X_2
14
12
10
8
6
4
2
0 2 4 6 8 10 12 14 X_1
e_2
$(\bar{X}_1, \bar{X}_2)$
e_1


X_2
14
12
10
8
6
4
2
0 2 4 6 8 10 12 14 X_1
(7, 14)
(4, 11)
e_2
$(\bar{X}_1, \bar{X}_2)$
(13, 5)
(8, 4)
e_1

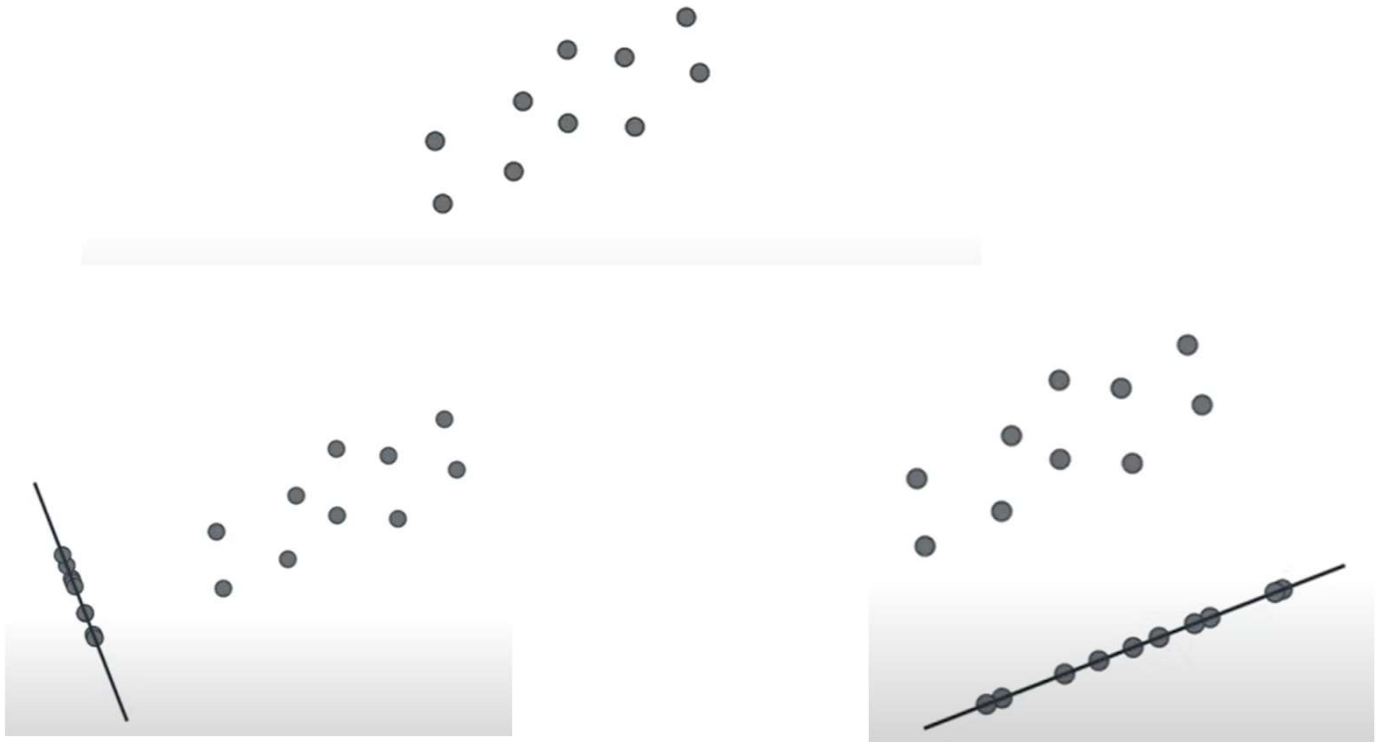
Taking a picture

Dimensionality Reduction

Which projection is good?



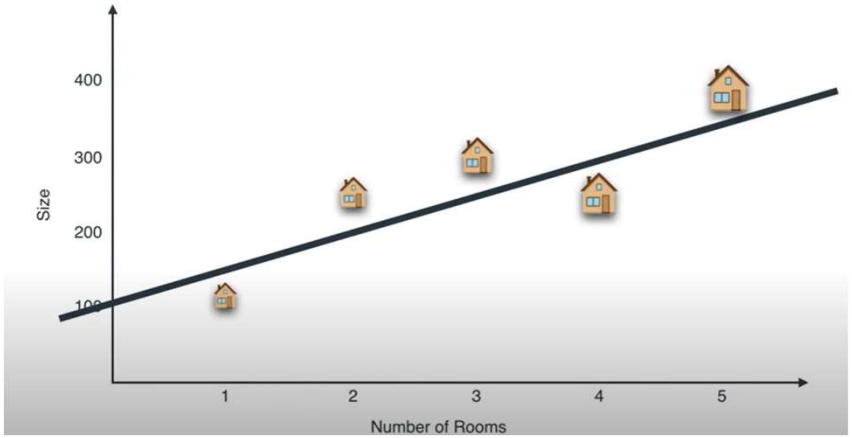
Dimensionality Reduction

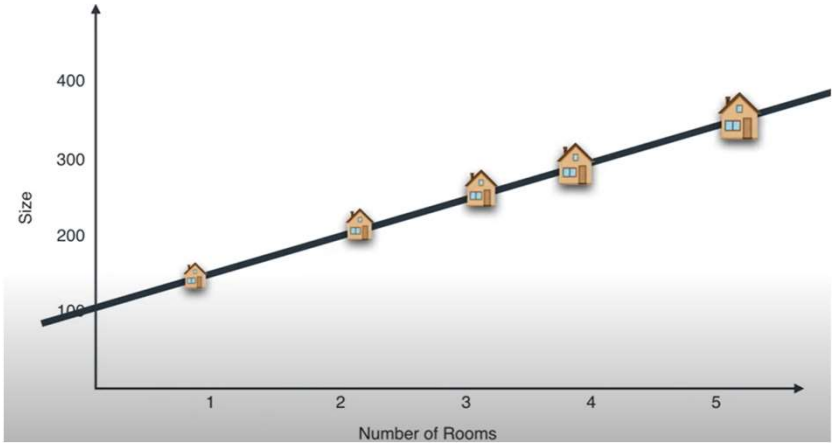


Housing Data

Size
Number of rooms
Number of bathrooms

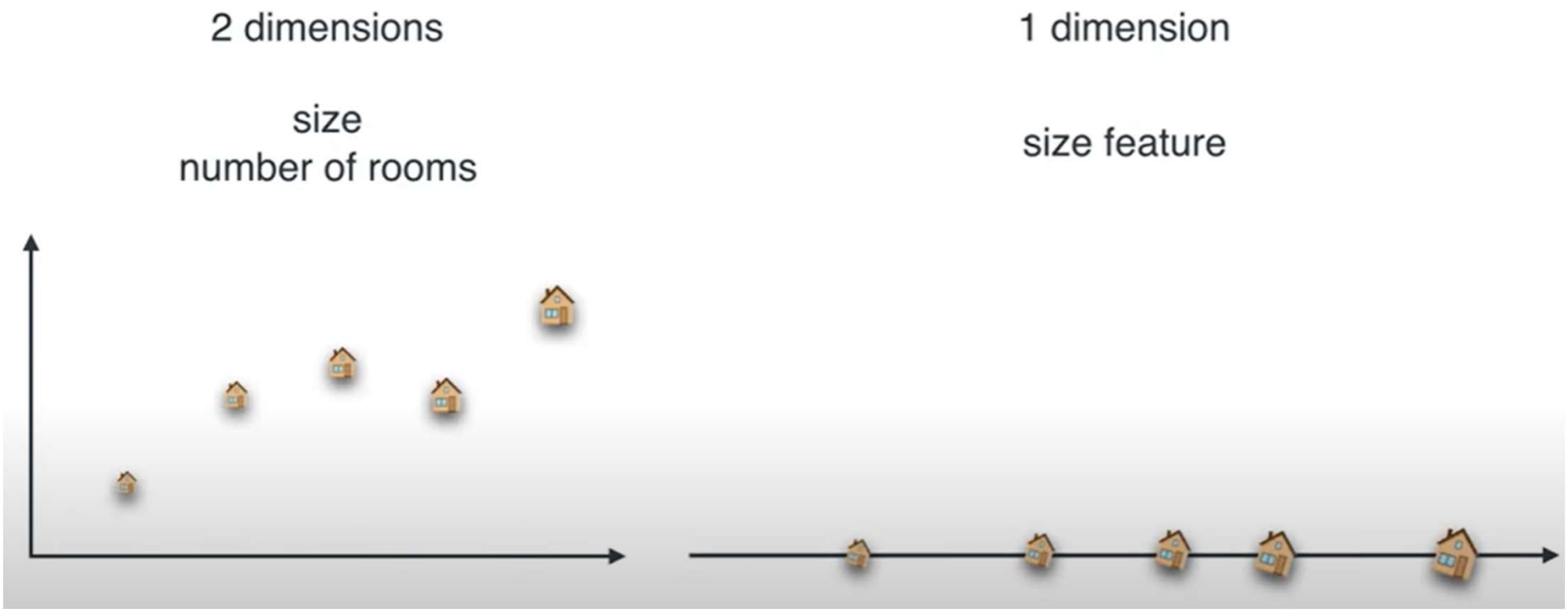
Schools around
Crime rate


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

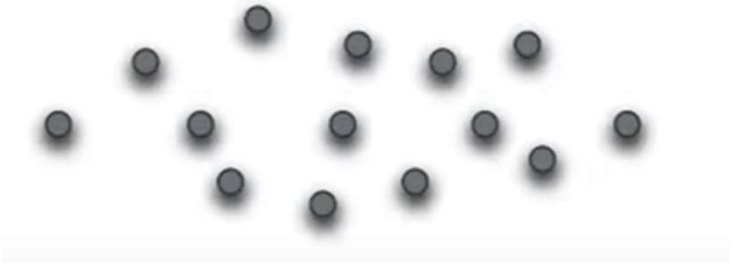

400
300
200
100
Size
1
2
3
4
5
Number of Rooms

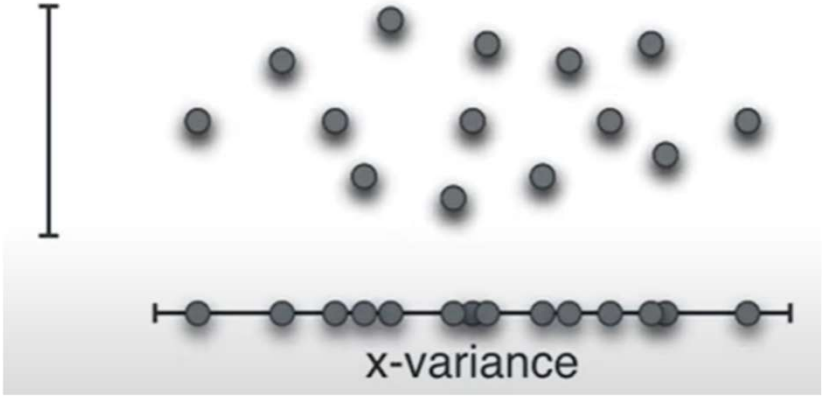

Size feature

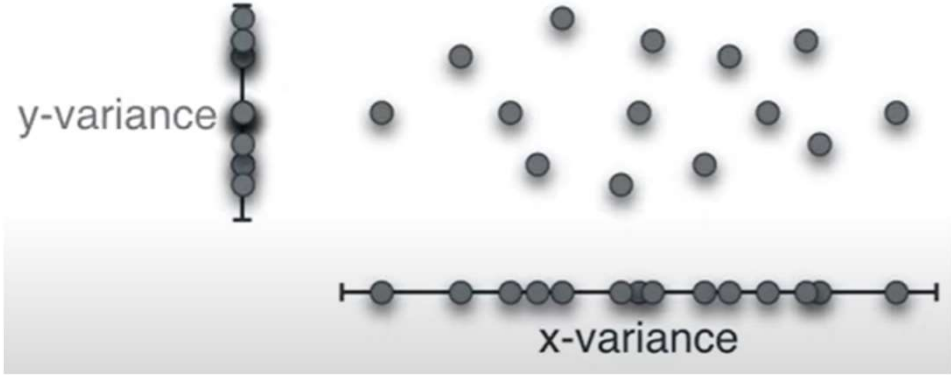
2D to 1D



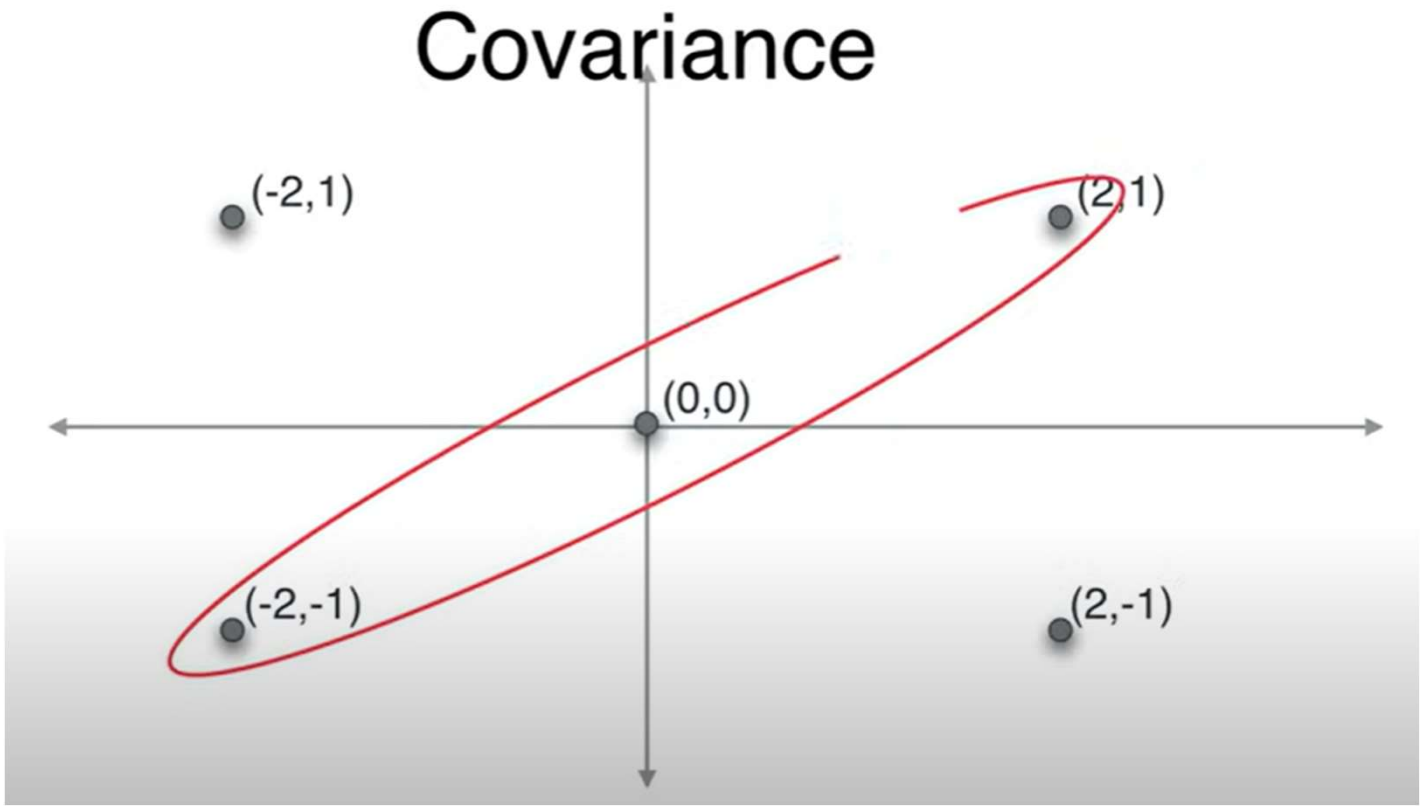
Variance



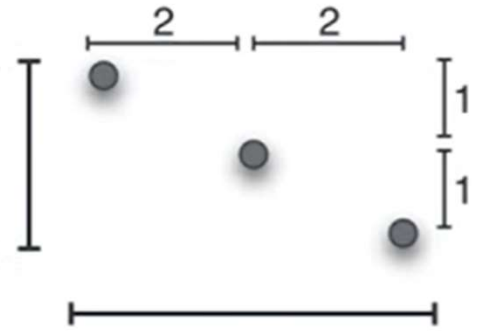


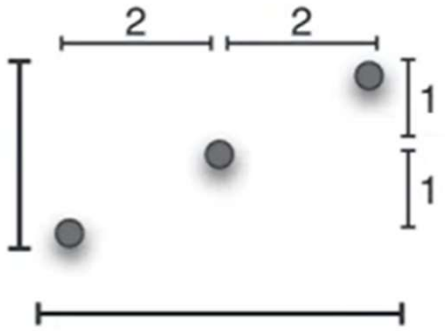


Covariance



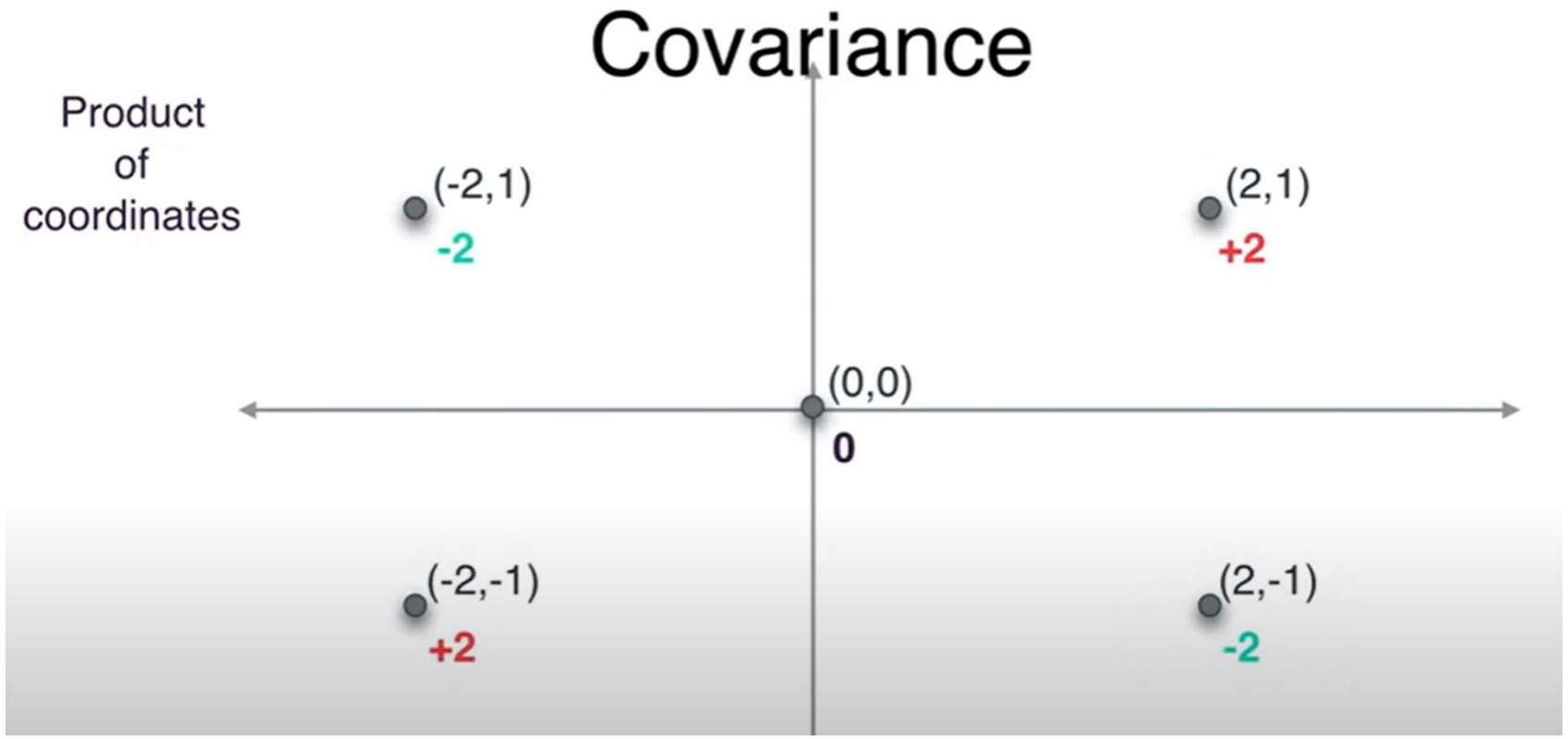
Variance



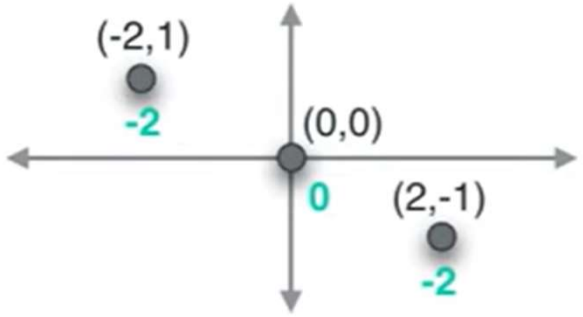


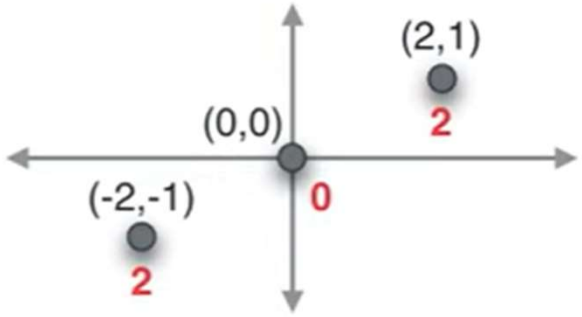
$$\text{x-variance} = \frac{2^2+0^2+2^2}{3} = 8/3$$

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


Covariance
Product
of
coordinates
(-2,1)
-2
(2,1)
+2
(0,0)
0
(-2,-1)
+2
(2,-1)
-2

Covariance





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

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

Covariance

(-2,1) **-2** (0,1) **0** (2,1) **2**

(-2,0) **0** (0,0) **0** (2,0) **0**

(-2,-1) **2** (0,-1) **0** (2,-1) **-2**

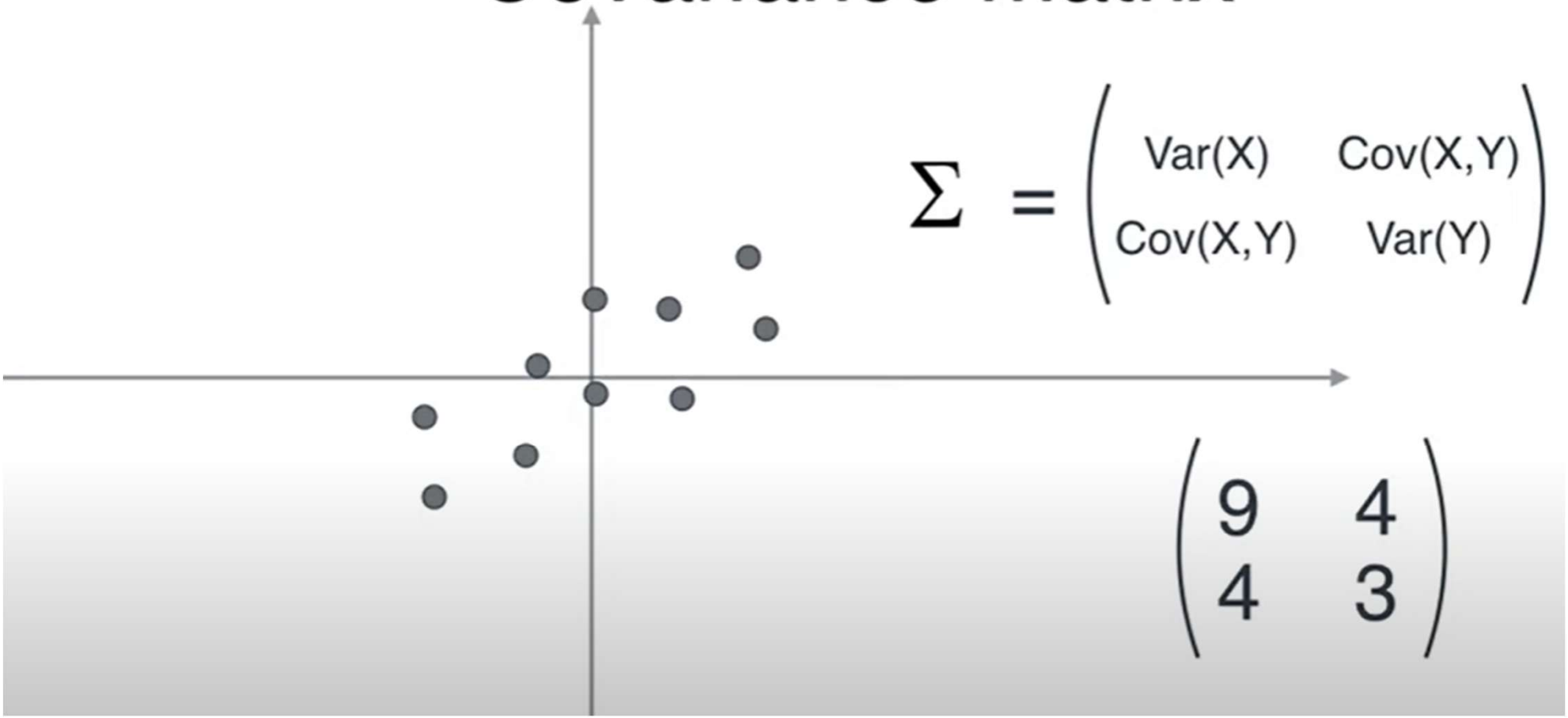
$$\text{covariance} = \frac{-2 + 0 + 2 + 0 + 0 + 0 + 2 + 0 + -2}{9}$$

negative covariance

covariance zero (or very small)

positive covariance

Covariance matrix

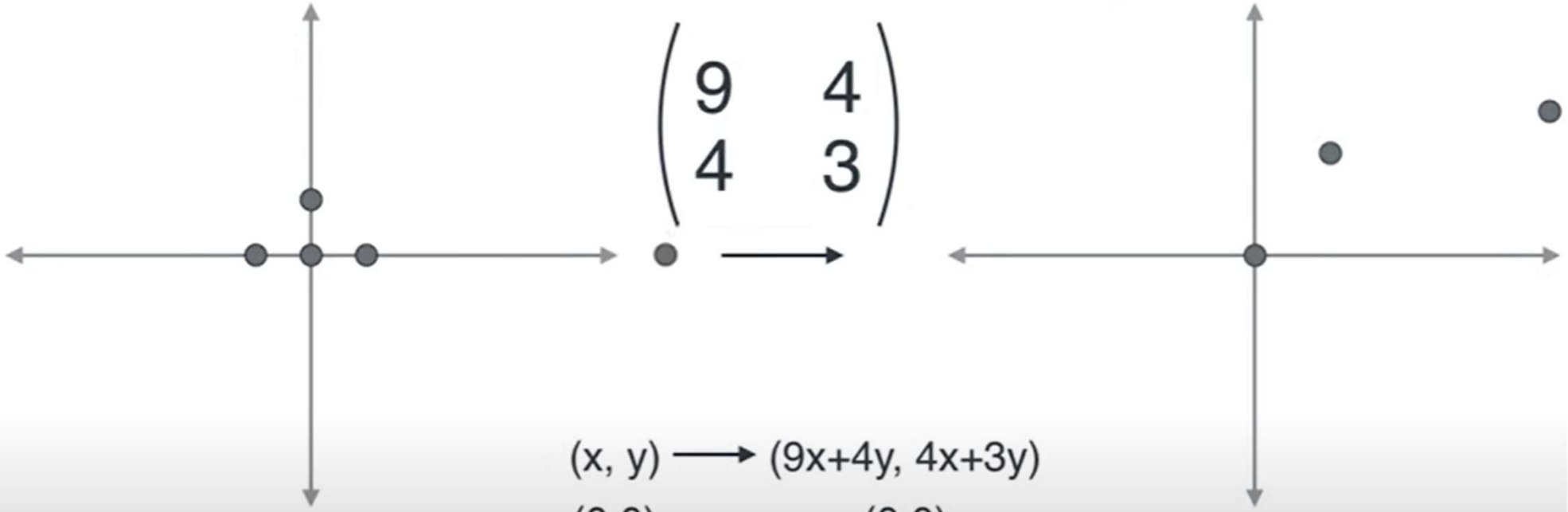


$$\Sigma = \begin{pmatrix} \text{Var}(X) & \text{Cov}(X,Y) \\ \text{Cov}(X,Y) & \text{Var}(Y) \end{pmatrix}$$

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

Linear Transformations

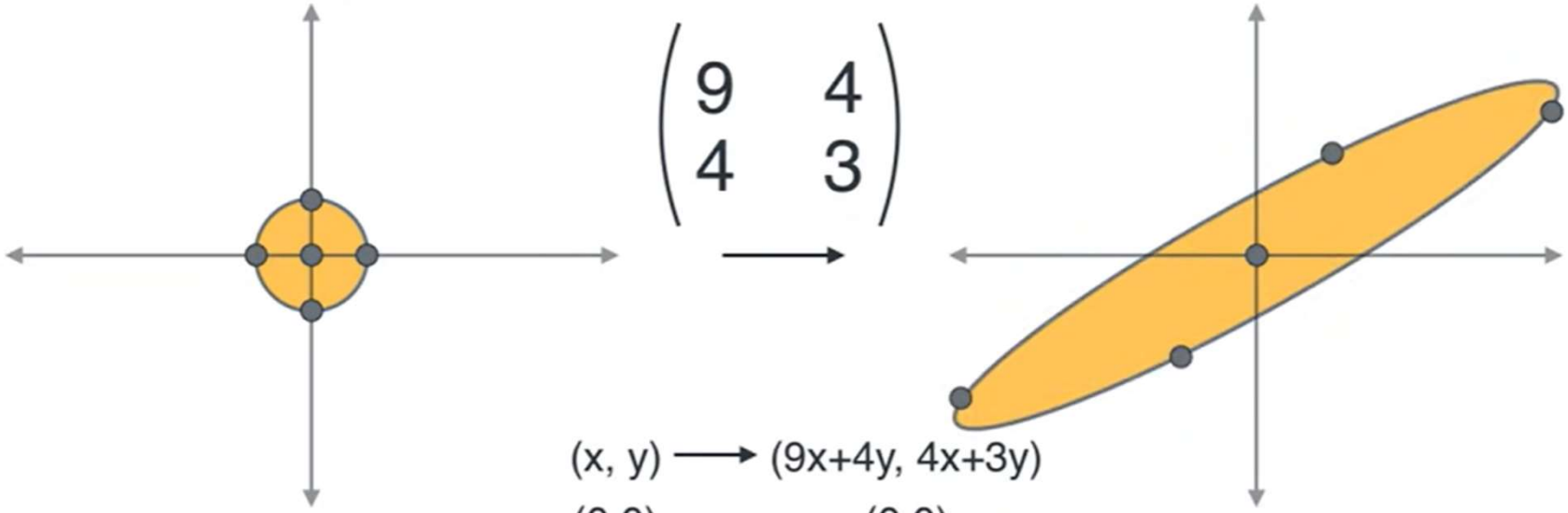
$$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$



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

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

Linear Transformations

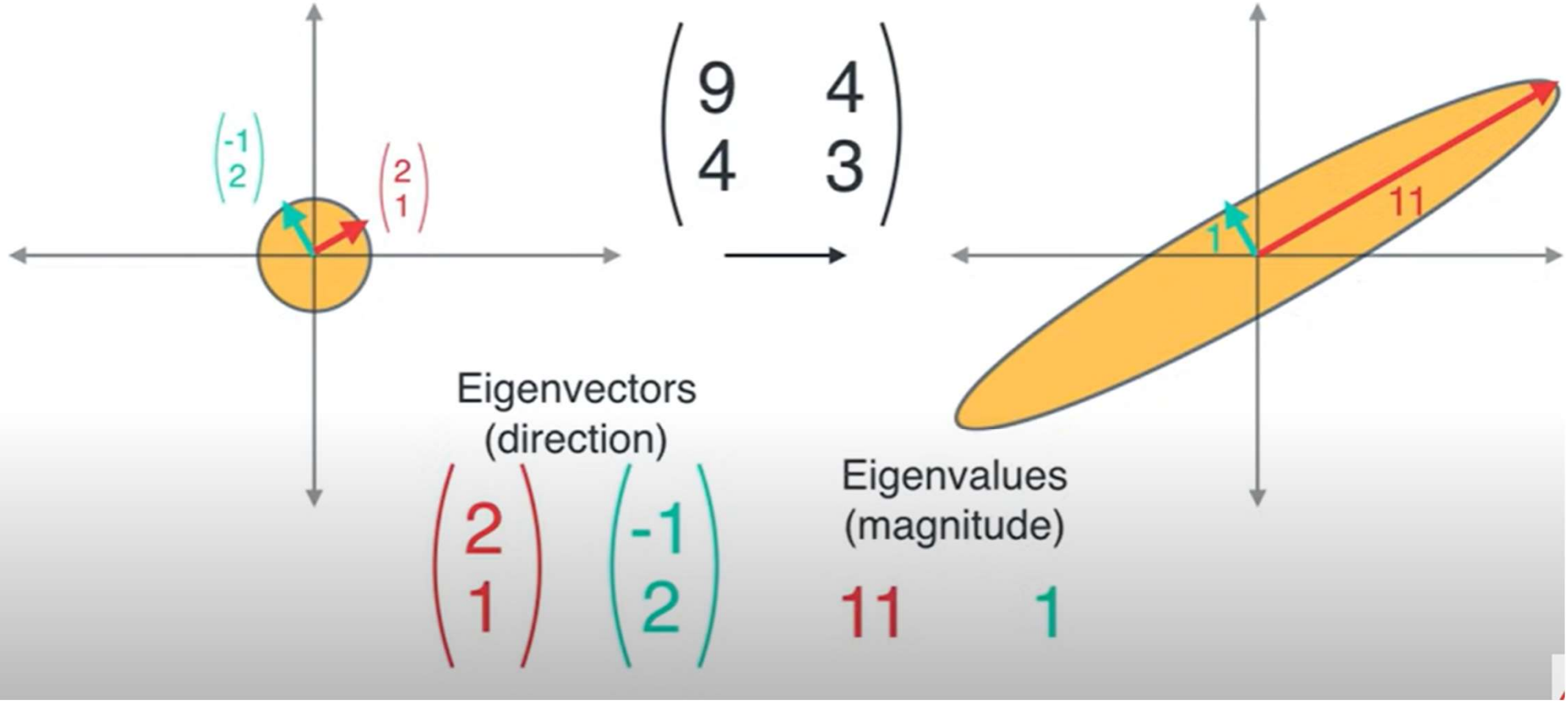


$$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

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

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

Linear Transformations



Eigen vectors

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

not an integer multiple of original vector

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

an integer multiple of original vector

$$2 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

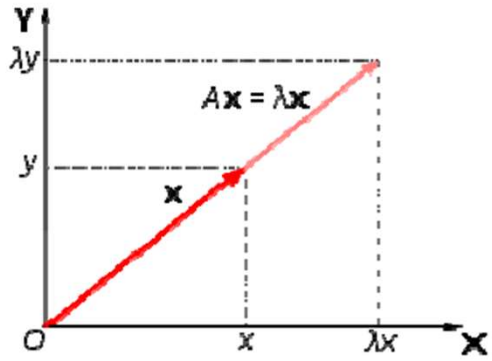
$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \times \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

Eigen vector features

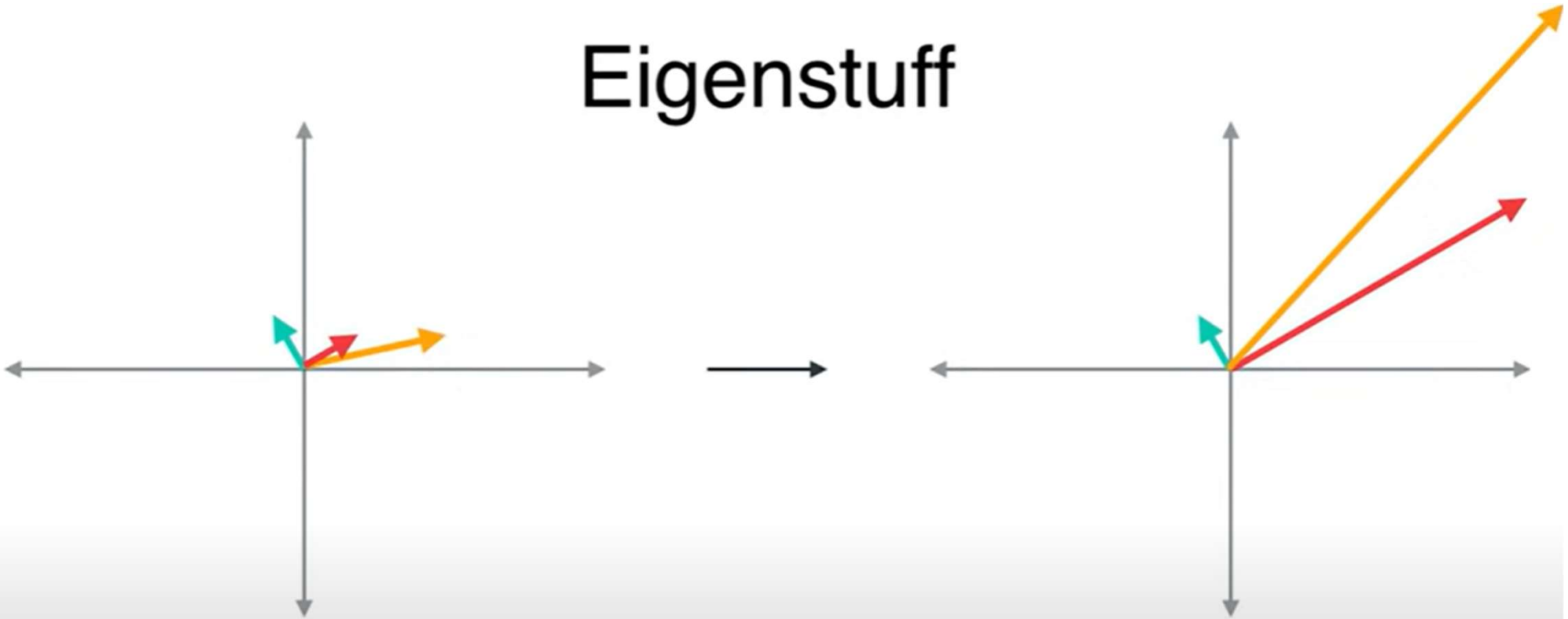
- Eigen vectors can only be found for square matrices.
- given an n X n matrix it has n eigenvectors
- Even if scale the vector by some amount before multiply it, still get the same multiple of it as result
- All the eigen vectors of a matrix are perpendicular (orthogonal)

Physical Significance……..

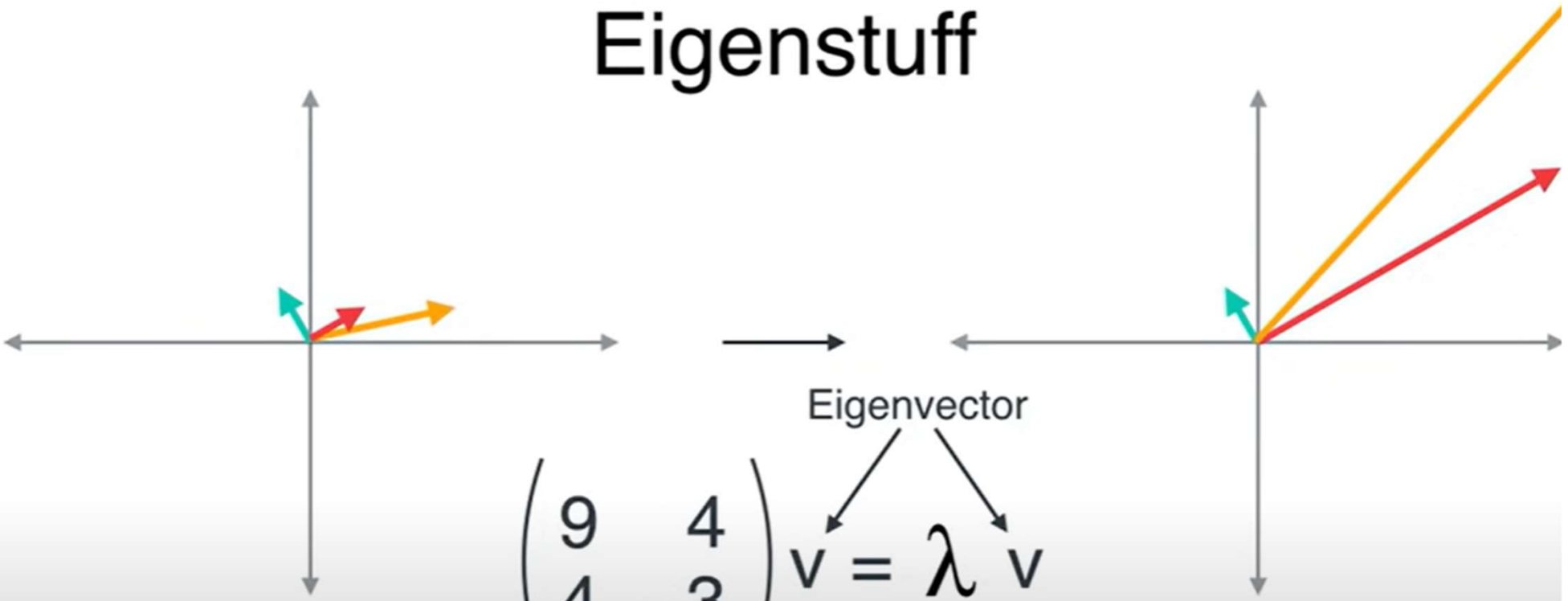
- Transformation matrix acts on certain vectors by changing only their magnitude and leaving the direction unchanged. These vectors are eigen vectors
- A matrix acts on an eigen vector by multiplying its magnitude by a factor. This values is eigen value associated with that eigen vector



Eigenstuff



Eigenstuff



Eigenvector

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

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

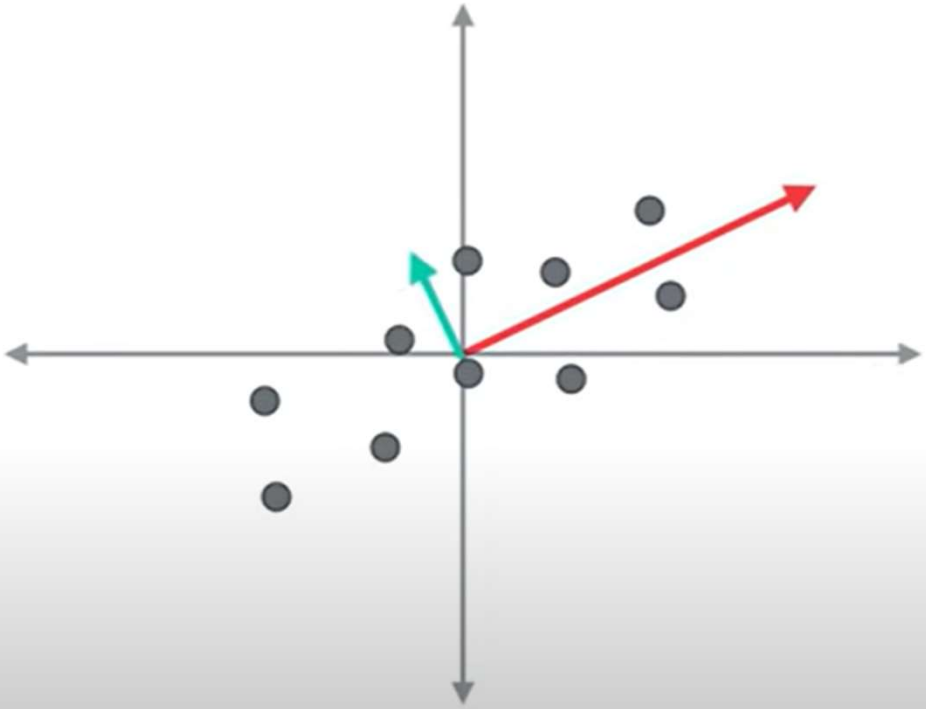
$$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = 11\begin{pmatrix} u \\ v \end{pmatrix}$$

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

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

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

Principal Component Analysis (PCA)



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

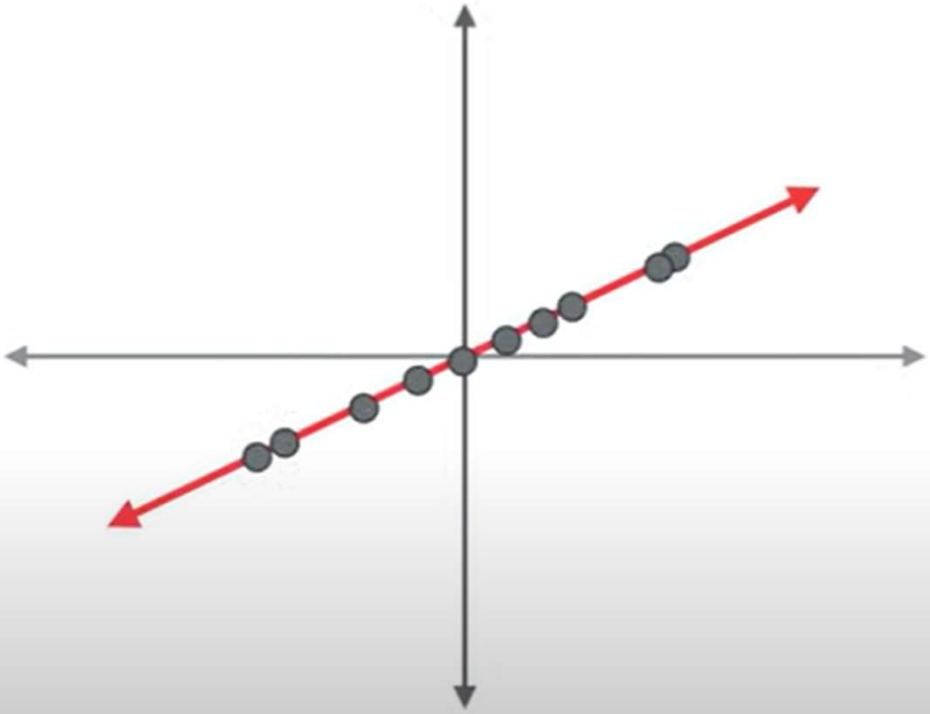
$$\begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

Eigenvectors (direction)

11 1

Eigenvalues (magnitude)

Principal Component Analysis (PCA)



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

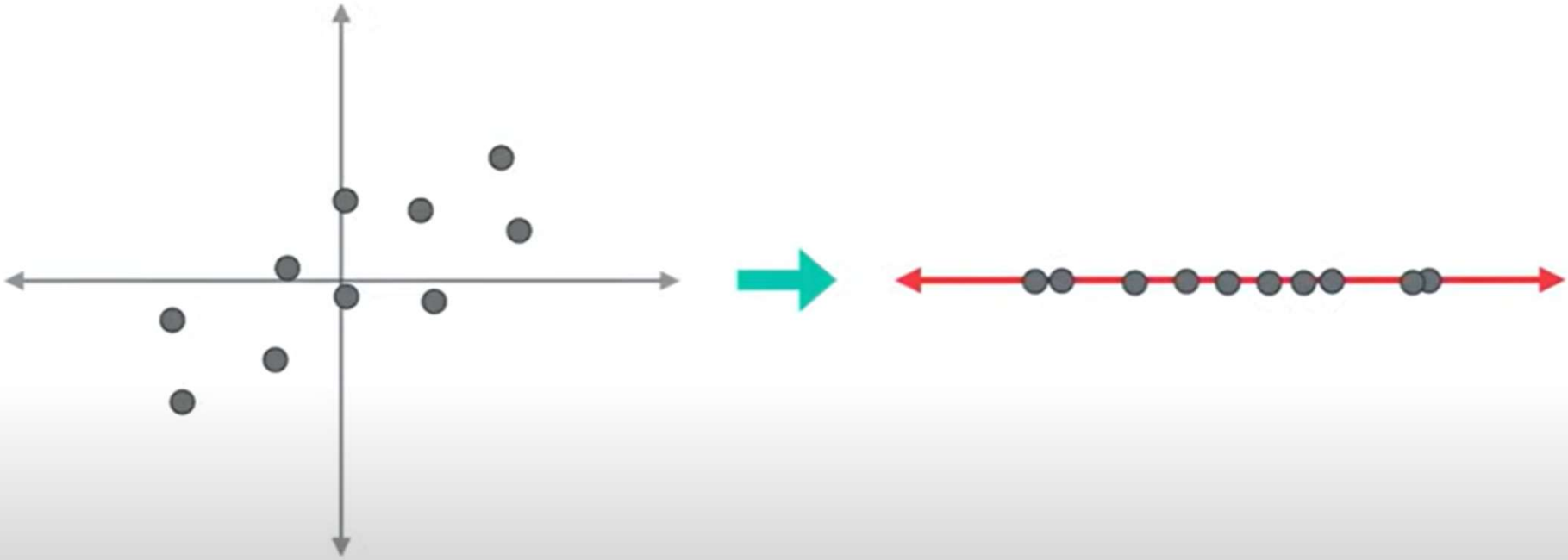
$$\begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

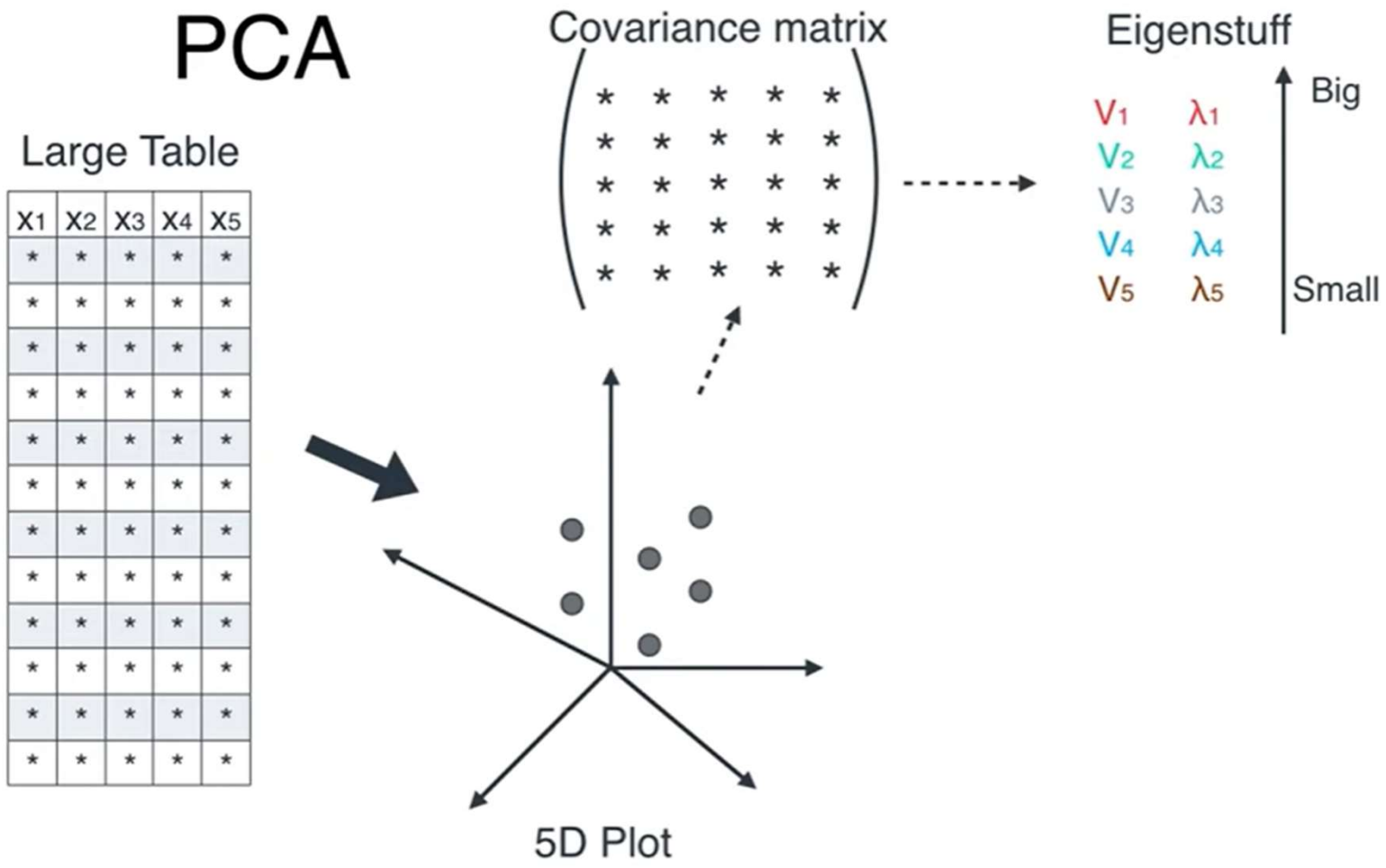
Eigenvectors
(direction)

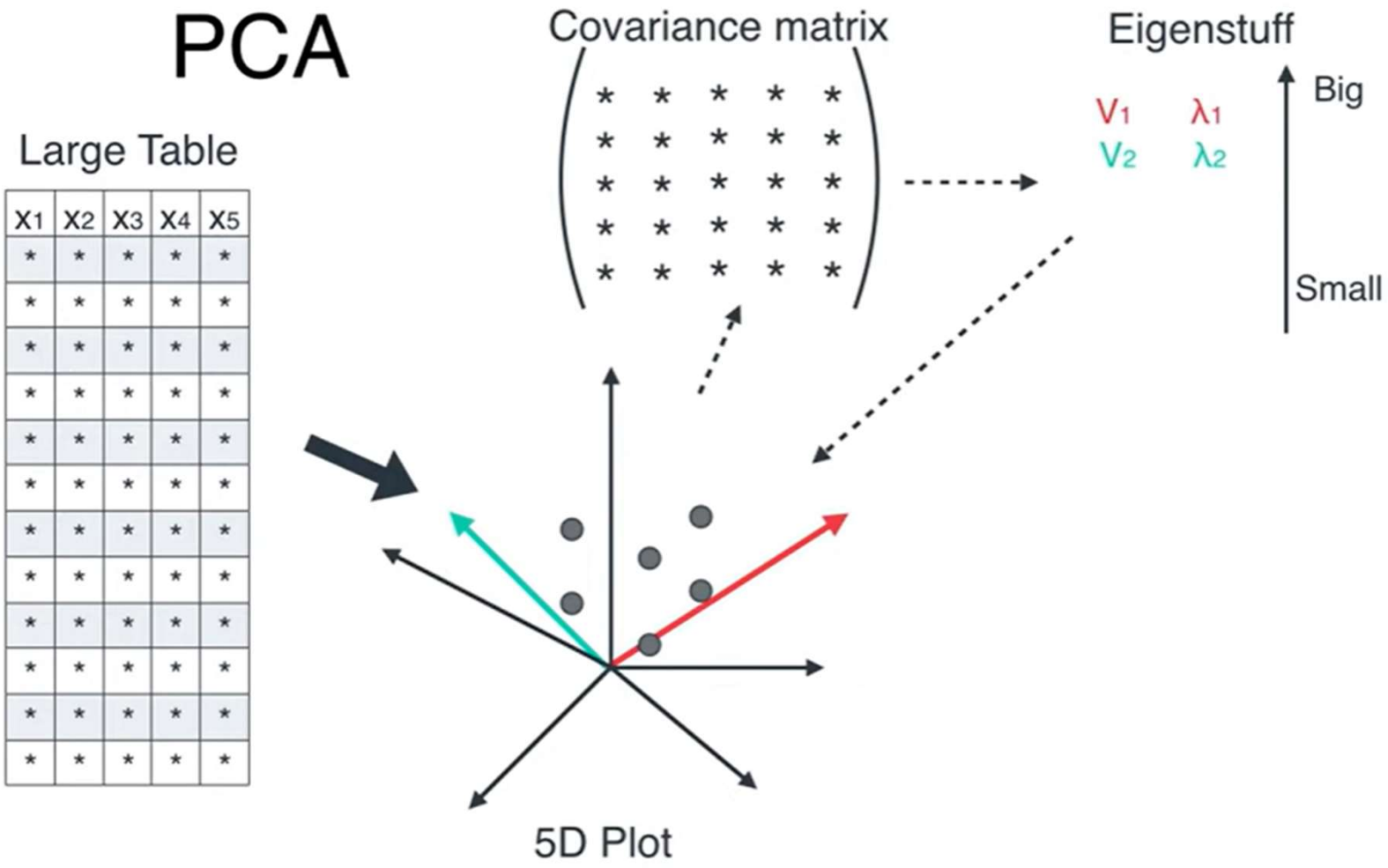
11

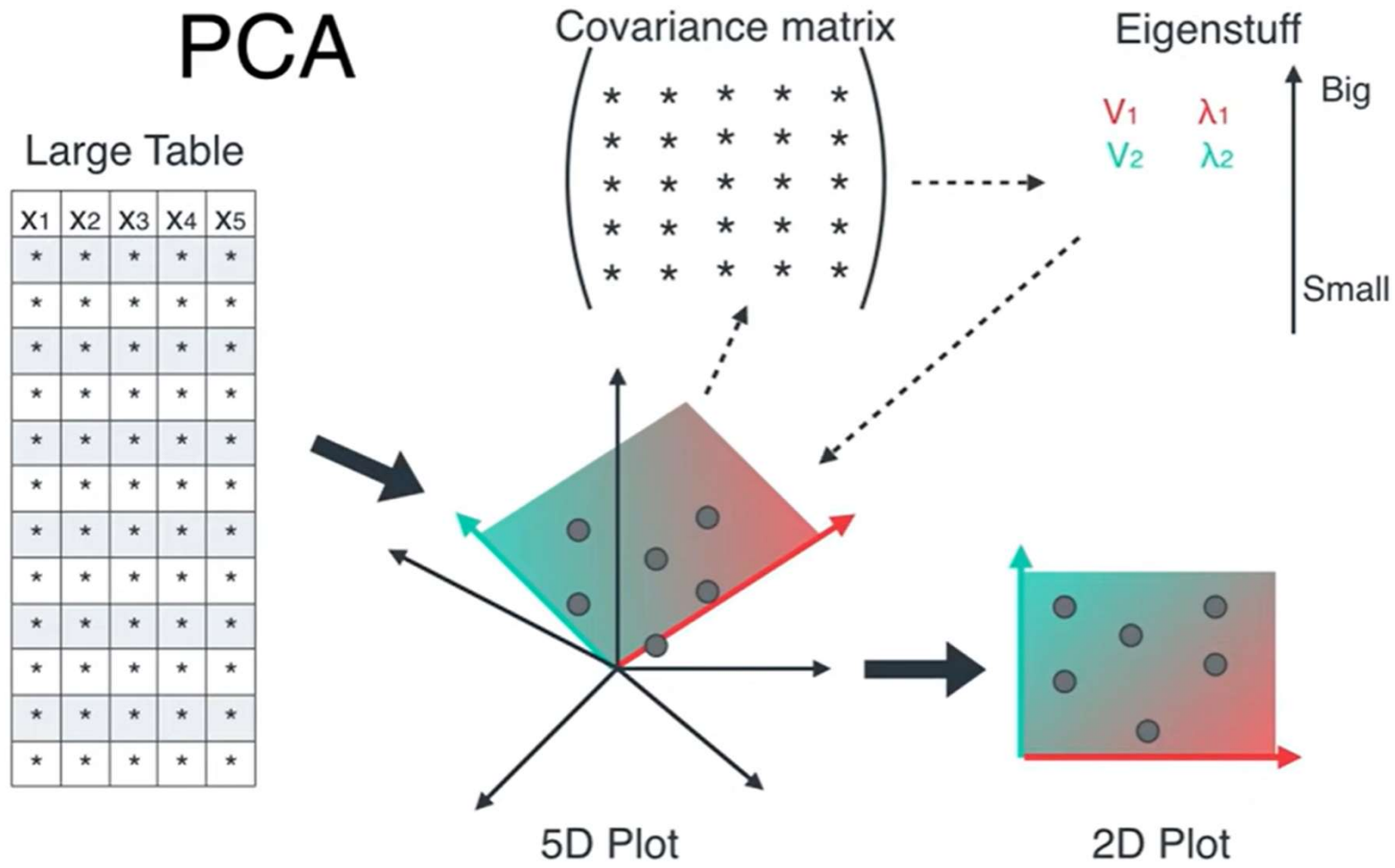
Eigenvalues
(magnitude)

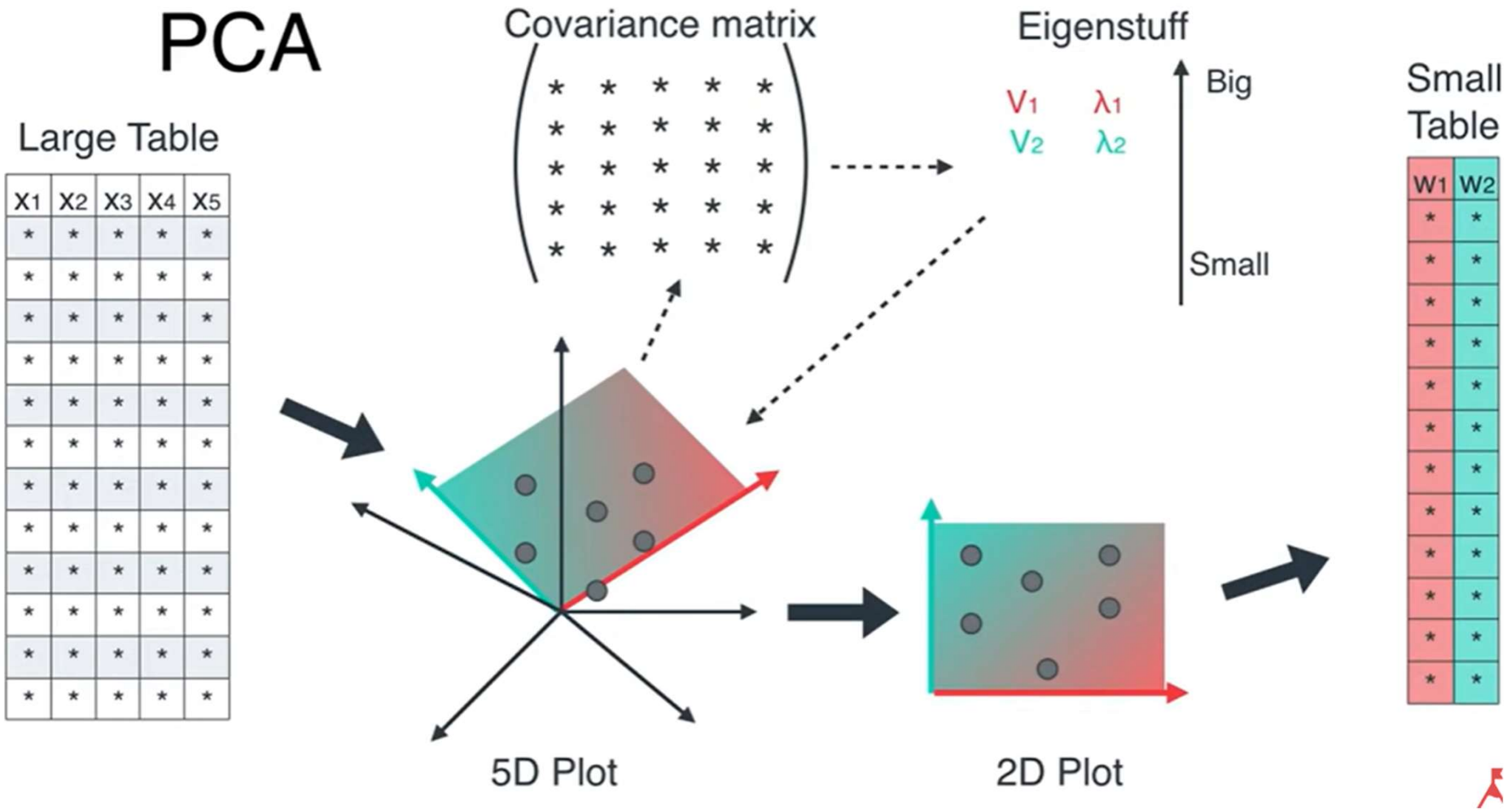
Principal Component Analysis (PCA)




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
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

PCA Steps

- Step 1 Get some data
- Step2 Subtract the mean – produces a data set whose mean is zero

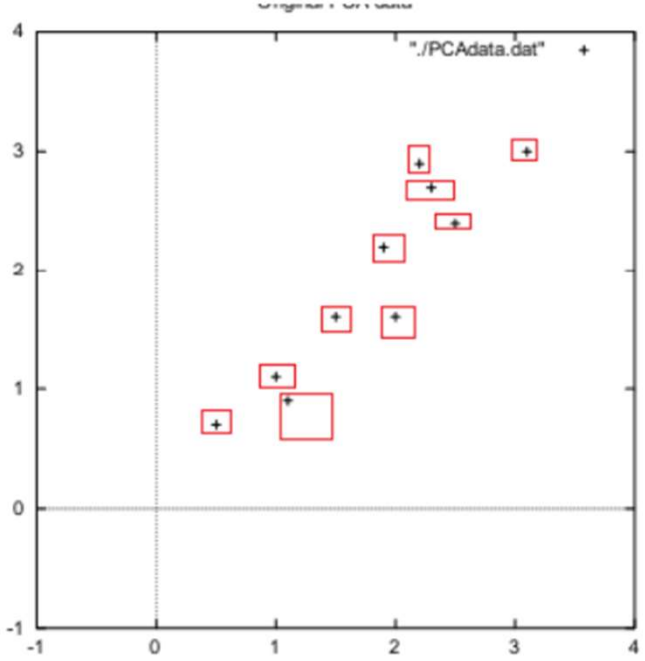
Data =

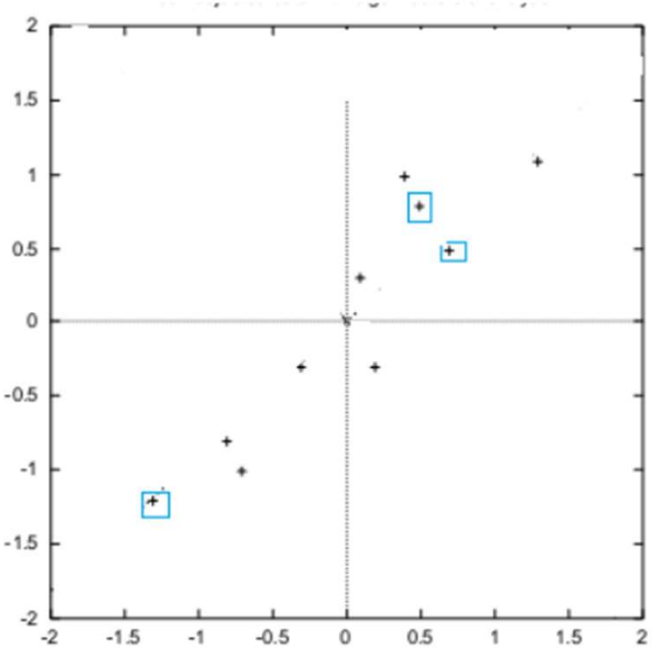
x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

PCA Steps





Mean adjusted data

PCA Steps

- Step3: Calculate the covariance matrix
- non-diagonal elements in this covariance matrix are both the variable increase together

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

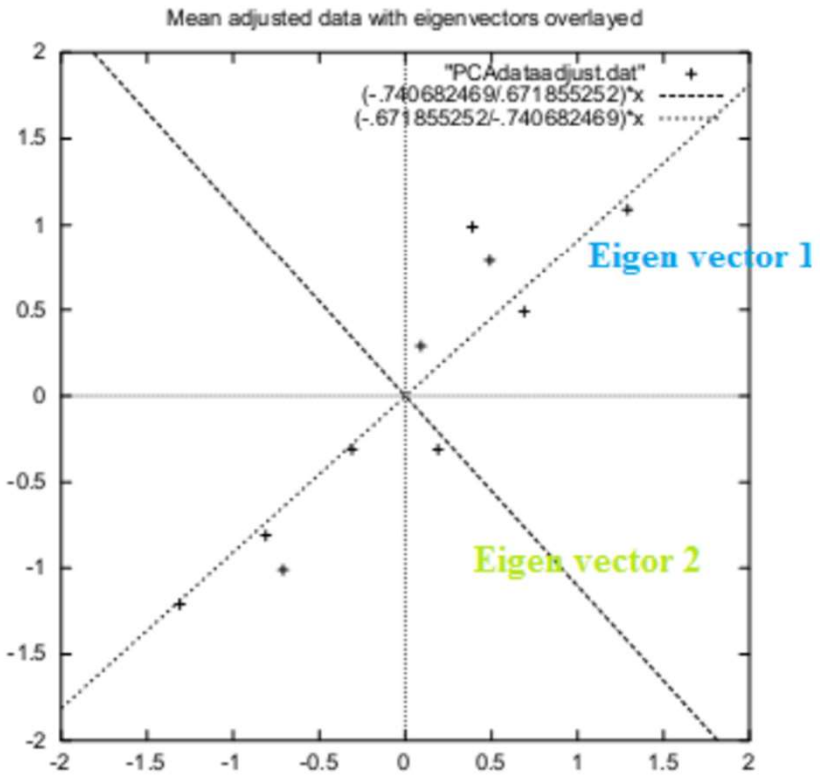
PCA Steps

- Step4:Calculate the eigen vectors and eigen values of the covariance matrix

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

PCA Steps

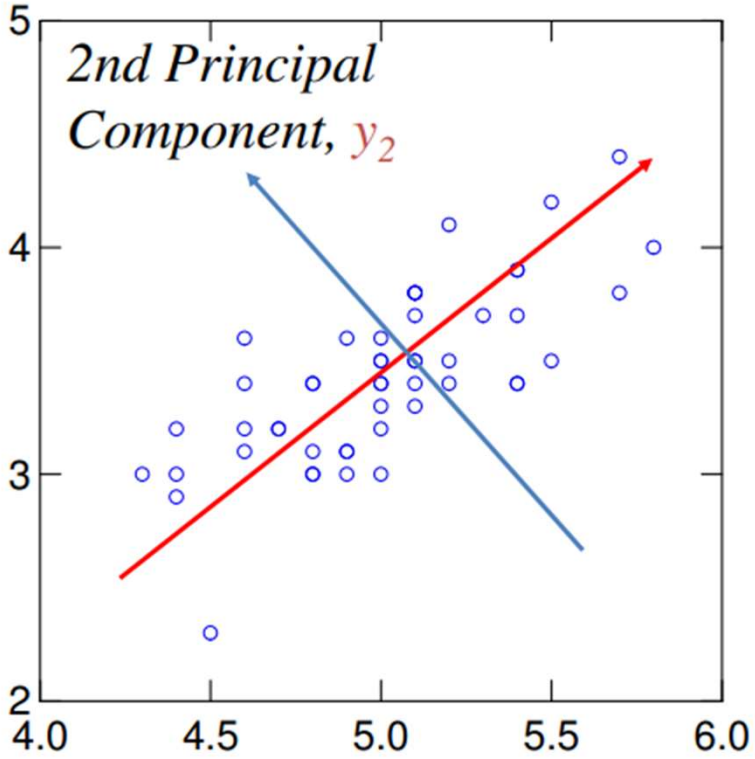


line of best fit

PCA Steps

- Step5:Choosing components and forming a feature vector
- Eigen vector with the highest eigen value is the principle compone
- Order by eigen value, highest to lowest gives the components in order of significance

$$FeatureVector = (eig_1\ eig_2\ eig_3\ ....\ eig_n)$$


2nd Principal
Component, y_2
1st Principal
Component, y_1
5
4
3
2
4.0
4.5
5.0
5.5
6.0

PCA Steps

- Step6:Deriving the new dataset

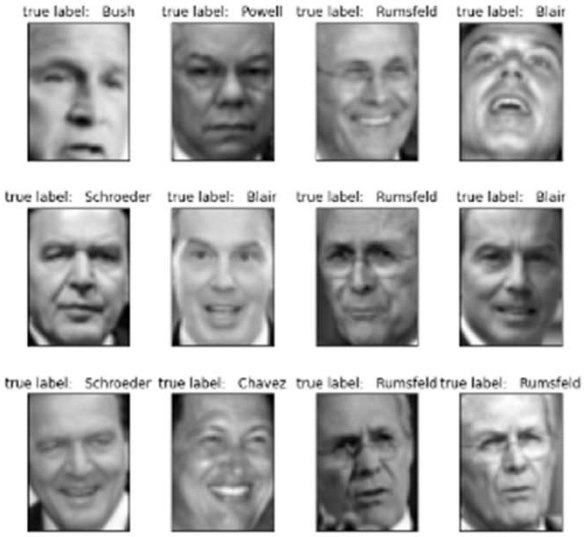
$$FinalData = RowFeatureVector \times RowDataAdjust,$$

Transformed Data=

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

Face Recognition Using PCA Algorithm

Let's Consider a set of *m* images of dimension *N*N* (training images).



Training Image with True Label (LFW people's dataset)

Reconstruction of original Data

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

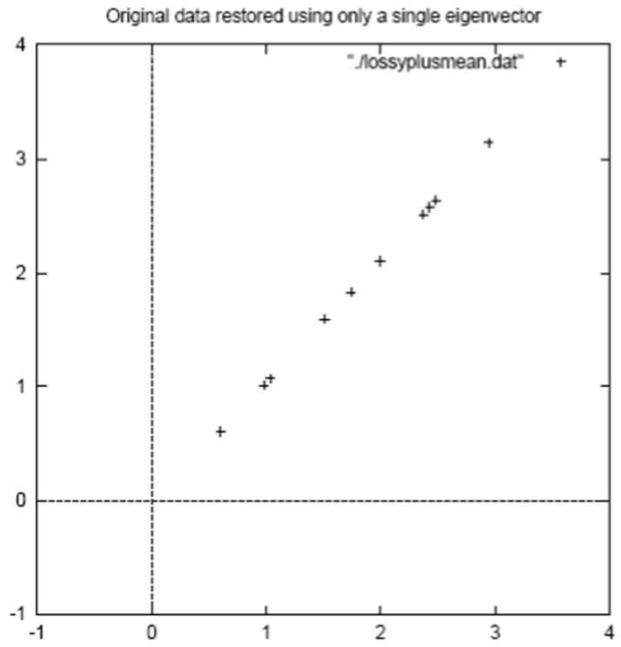
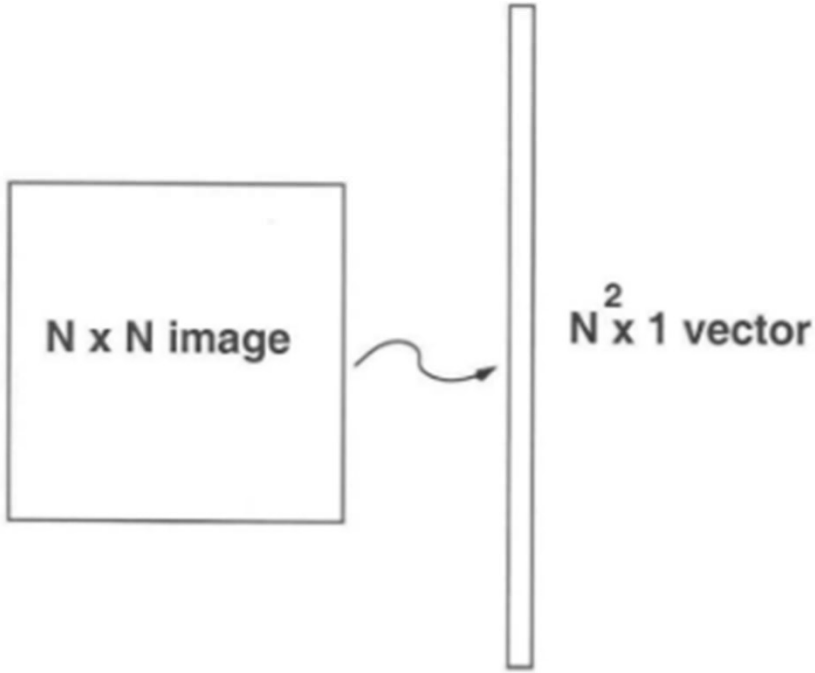


Figure 3.5: The reconstruction from the data that was derived using only a single eigenvector

We first convert these images into vectors of size N^2 such that:

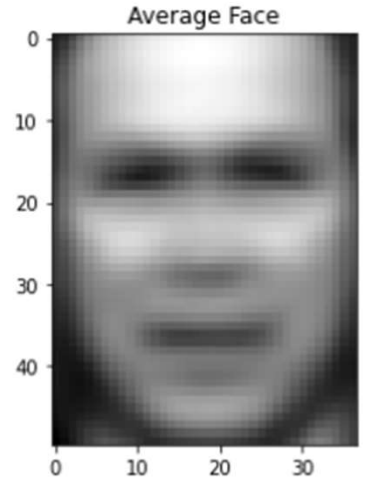


$$x_1, x_2, x_3 ... x_m$$

Now we calculate the average of all these face vectors and subtract it from each vector

$$\psi = \frac{1}{m} \sum_{i=1}^{m} x_i$$

$$\mathfrak{a}_i = x_i - \psi$$



average_face

$$A = [a_1 \quad a_2 \quad a_3 \quad .... \quad a_m]$$

- Now, we find Covariance matrix by multiplying *A* with A^T. A has dimensions $N^2 * M$, thus A^T has dimensions $M * N^2$. When we multiplied this gives us matrix of $N^2 * N^2$, which gives us N^2 eigenvectors of N^2 size which is not computationally efficient to calculate. So we calculate our covariance matrix by multiplying A^T and *A*. This gives us *M * M* matrix which has *M (assuming* $M << N^2$*)* eigenvectors of size *M*.

$$Cov = A^T A$$

- In this step we calculate eigen values and eigenvectors of above covariance matrix using the formula below.

$$A^T A \nu_i = \lambda_i \nu_i$$

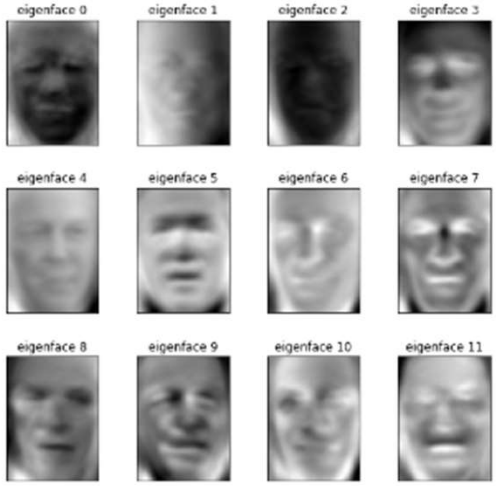
$$A A^T A \nu_i = \lambda_i A \nu_i$$

$$C' u_i = \lambda_i u_i$$

In this step we used the eigenvectors that we got in previous step. We take the normalized training faces (face – average face)
x_i
and represent each face vectors in the linear of combination of the best **K** eigenvectors (as shown in the diagram below).

$$\mathrm{x}_i - \psi = \sum_{j=1}^{K} w_j u_j$$

These
u_j
are called **EigenFaces**.



EigenFaces

In this step, we take the coefficient of eigenfaces and represent the training faces in the form of a vector of those coefficients.

$$x_i = \begin{bmatrix} w_1^i \\ w_2^i \\ w_3^i \\ . \\ . \\ w_k^i \end{bmatrix}$$

Linear Combination of EigenFaces

Testing/Detection Algorithm :

Test Images With true labels

- Given an unknown face *y*, we need to first preprocess the face to make it centered in the image and have the same dimensions as the training face
- Now, we subtract the face from the average face
  ψ
  .

$$\phi = y - \psi$$

Now, we project the normalized vector into eigenspace to obtain the linear combination of eigenfaces.

$$\phi = \sum_{i=1}^{k} w_i u_i$$

From the above projection, we generate the vector of the coefficient such that

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \\ w_k \end{bmatrix}$$

We take the vector generated in the above step and subtract it from the training image to get the minimum distance between the training vectors and testing vectors

$$e_r = min_l \, \|\Omega - \Omega_l\|$$

If this
e_r
is below tolerance level T_r, then it is recognised with l face from training image else the face is not matched from any faces in training set.

Test images With prediction

•L. Sirovich and M. Kirby (1987). "Low-dimensional procedure for the characterization of human faces". *Journal of the Optical Society of America A*. 4(3): 519–524.
•M. Turk and A. Pentland (1991). "Eigenfaces for recognition". *Journal of Cognitive Neuroscience*. 3(1): 71–86.

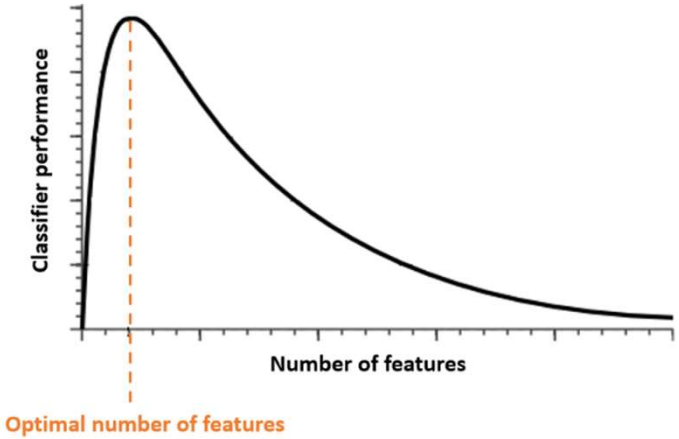
Real World Applications of PCA	
Reducing the size of images.	PCA can be used to reduce the size of an image without significantly impacting the quality. Beyond just reducing the size, this is useful for image classification algorithms.
Visualizing multidimensional data.	PCA allows us to represent the information contained in multidimensional data in reduced dimensions which are more compatible with visualization.
Finding patterns in high-dimensional datasets.	Examining the relationships between principal components and original features can help uncover patterns in the data that are harder to identify in our full dataset.
Stock price prediction in finance.	Many models of stock price prediction rely on estimating covariance matrices. However, this can be difficult with high-dimensional data. PCA can be used for data reduction to help remedy this issue.
Dataset reduction in healthcare models.	Healthcare models use high-dimensional datasets because there are many factors that influence healthcare outcomes. PCA provides a method to reduce the dimensionality while still capturing the relevant variance.

The Curse of Dimensionality

The phrase, attributed to Richard Bellman, was coined to express the difficulty of using brute force (a.k.a. grid search) to optimize a function with too many input variables.

3. Hughes Phenomenon

The CoD is closely related to the Hughes Phenomenon. It states that with a fixed number of training instances, the performance of a classifier first increases as the number of features increases until we reach the optimal number of features. Adding more features beyond this value will deteriorate the classifier's performance:



When the dimensionality increases, the volume of the space increases, and the data become more sparse