

Principal Component Analysis

CSC 461: Machine Learning

Fall 2022

Prof. Marco Alvarez
University of Rhode Island

Dimensionality reduction

- Basically ...
 - ✓ mapping **high-dimensional** data into **low dimensional** data
 - ✓ can be as simple as dropping some features or using a combination of all features
- Given $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^d$, find a representation $\mathcal{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}$ with $\mathbf{z}_i \in \mathbb{R}^{d'}$ and $d' \ll d$
 - ✓ properties should be preserved (e.g., variance, distances, neighborhood)

$$\begin{bmatrix} \text{---} & \mathbf{x}_1^T & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{x}_n^T & \text{---} \end{bmatrix}_{n \times d} \Rightarrow \begin{bmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_n^T \end{bmatrix}_{n \times d'}$$

Why dimensionality reduction?

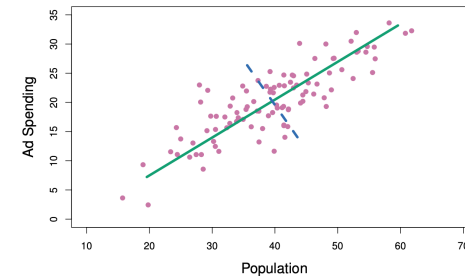
- Data visualization (2-d or 3-d)
- Preprocess data before machine learning
 - ✓ algorithms can focus on important features/patterns
 - ✓ training can be more efficient
- Removing noise and redundant information
- Data compression

Principal component analysis

eigendecomposition of the covariance matrix

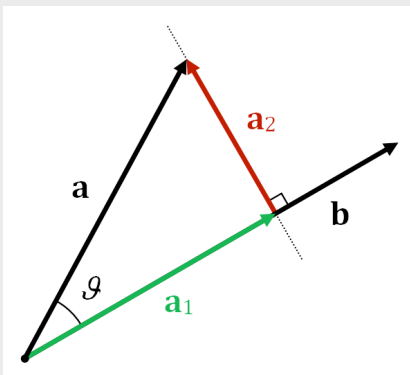
Principal component analysis

- Find projections of the data onto directions that maximize variance
- directions are **orthogonal** to each other



<https://www.dataschool.io/15-hours-of-expert-machine-learning-videos/>

Preliminaries



The **vector projection** of vector **a** onto a nonzero vector **b** is the orthogonal projection of **a** onto a straight line parallel to **b**:

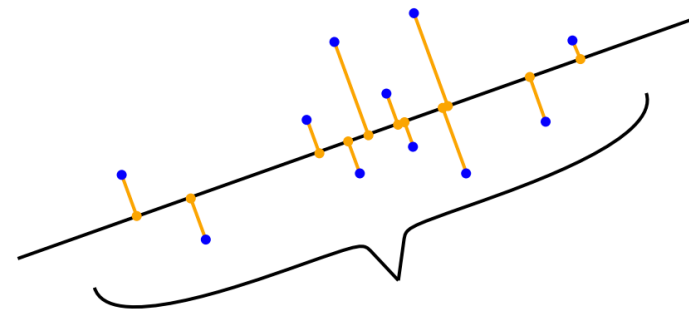
$$\mathbf{a}_1 = a_1 \hat{\mathbf{b}}$$

where a_1 is a scalar, called the **scalar projection** of **a** onto **b**, and $\hat{\mathbf{b}}$ is the unit vector in the direction of **b**. The scalar projection is defined as:

$$a_1 = \mathbf{a} \cdot \hat{\mathbf{b}}$$

https://en.wikipedia.org/wiki/Vector_projection

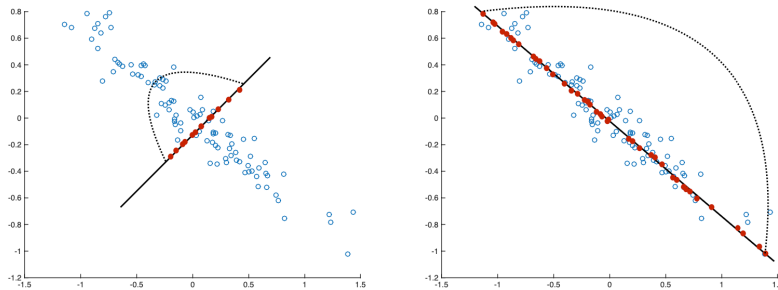
Data projection on a line



PCA idea is finding a line that maintains as much variance (spread) as possible when the data is projected.

Mathematics for Machine Learning

What is the best line? — maximize variance



Idea: minimize sum of square distances to the line (or maximize the sum of squares of the scalar projections)

Machine Learning for Data Science (CS 4786), Cornell

Disclaimer

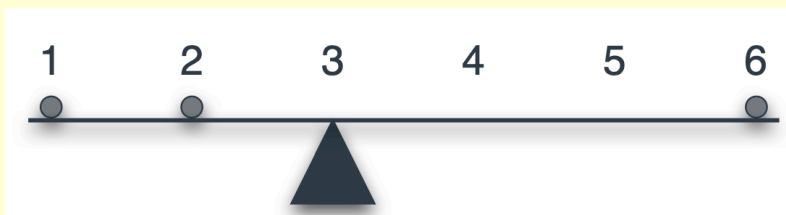
► The following slides contain figures adapted from:

✓ Unsupervised Learning **SERRANO.ACADEMY**
the art of understanding
✓ <https://serrano.academy/unsupervised-learning/>

► Video:

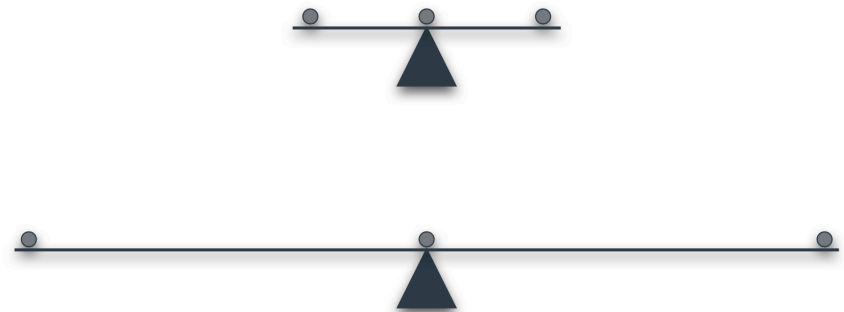
✓ <https://www.youtube.com/watch?v=g-Hb26agBFg>

The mean




$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$


Doesn't always help



The variance



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

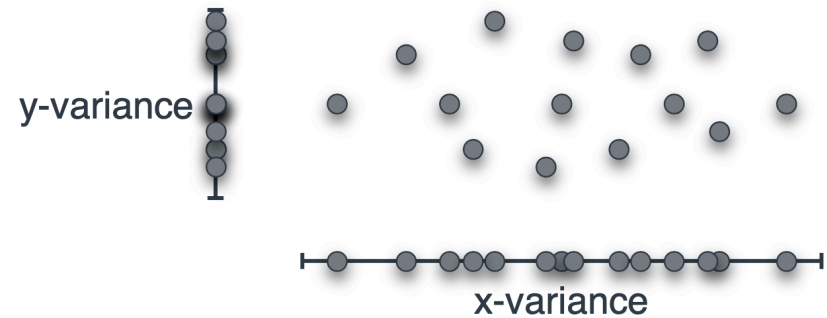


$$\text{Variance} = \frac{5^2 + 0^2 + 5^2}{3} = 10/3$$

“biased” estimator
default in numpy

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

Variance and data



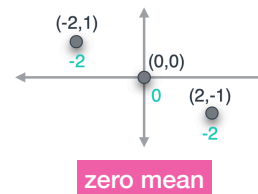
Doesn't always help



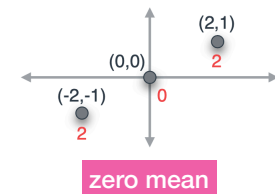
$$\text{x-variance} = \frac{2^2 + 0^2 + 2^2}{3} = 8/3$$

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

Covariance



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



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

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

Covariance



negative
covariance

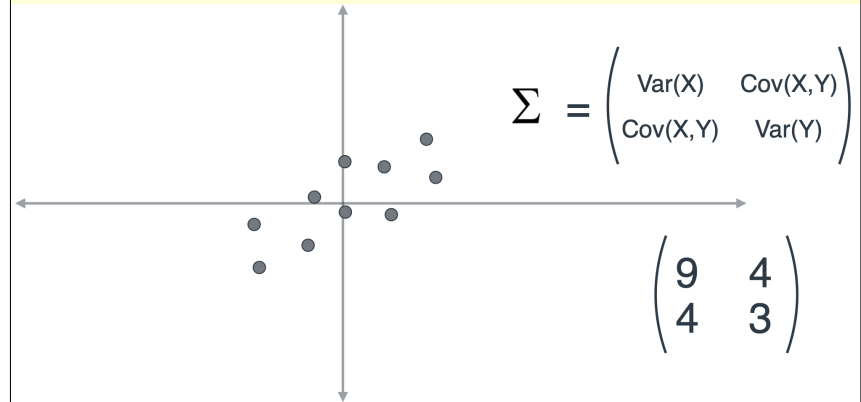


covariance zero
(or very small)



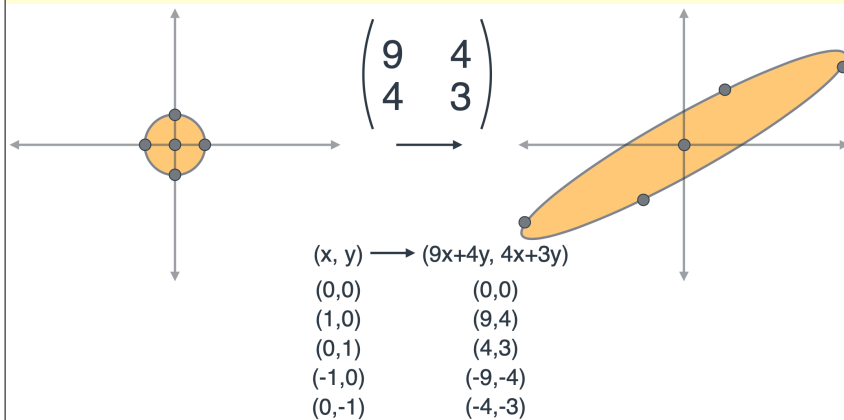
positive
covariance

Covariance matrix

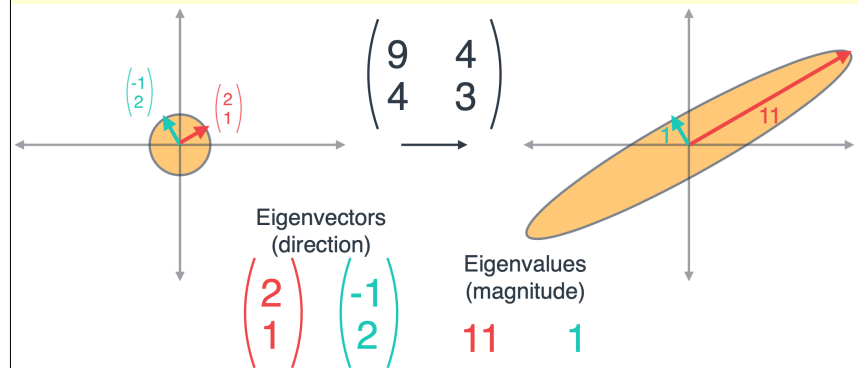


Every element Σ_{ij} of the covariance matrix is the covariance between variable i and variable j

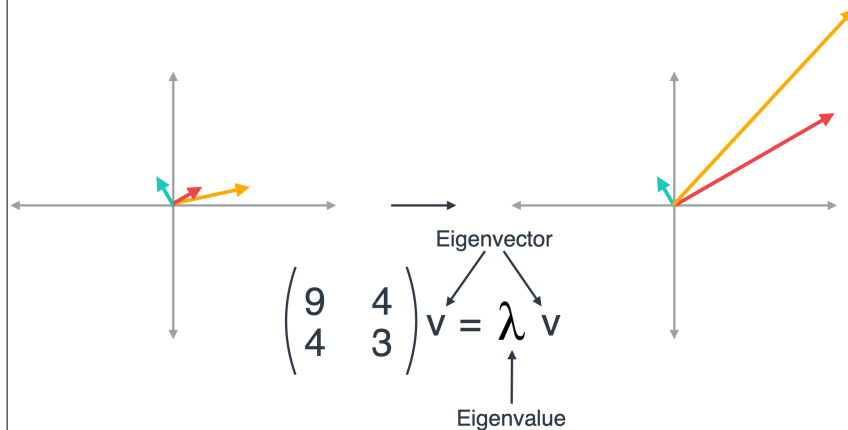
Linear transformations



Eigenvectors and eigenvalues



Eigenvectors and eigenvalues



Eigenvectors and eigenvalues

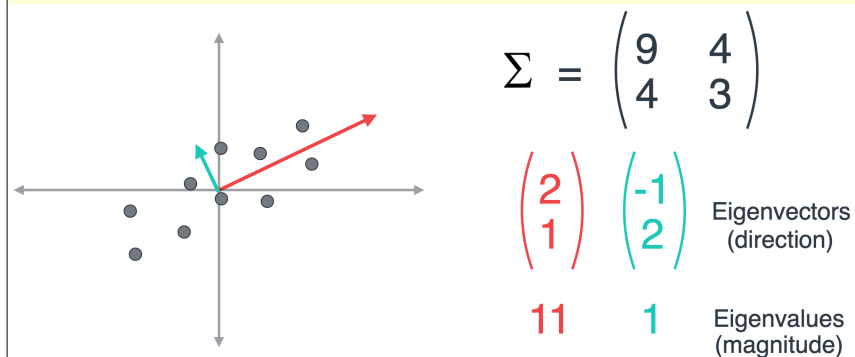
- The decomposition of a **square matrix A** into **eigenvalues** and **eigenvectors** is known as **eigen decomposition**

✓ for **real symmetric matrices** eigenvectors can be chosen real and orthonormal

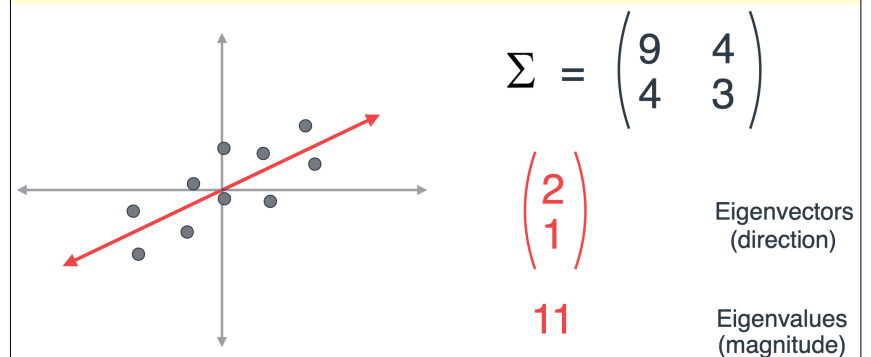
$$A = V\Lambda V^T \quad A\mathbf{v} = \lambda\mathbf{v}$$

columns of V are the eigenvectors of A and Λ is a diagonal matrix whose entries are the eigenvalues of A

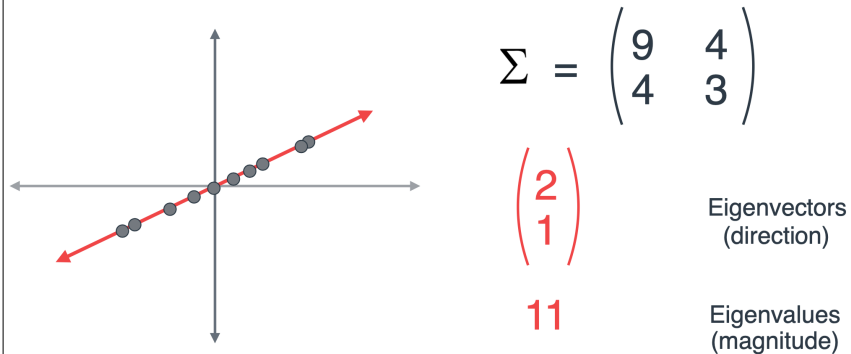
Principal component analysis (PCA)



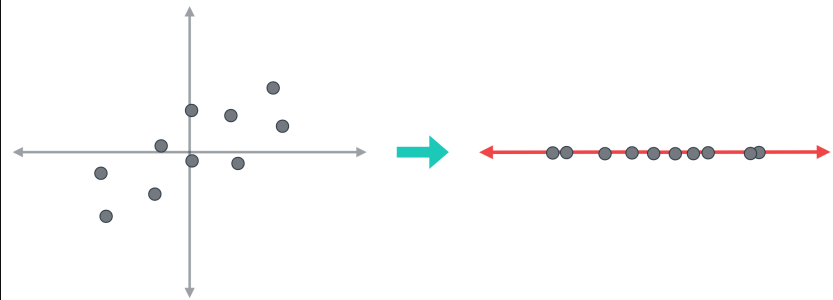
Principal component analysis (PCA)



Principal component analysis (PCA)



Principal component analysis (PCA)



More formally

- Given $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^d$
 ✓ find $\mathcal{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}$ with $\mathbf{z}_i \in \mathbb{R}^{d'}$ and $d' \ll d$

- Example from 3-d to 2-d

$$\mathbf{v}_1 \in \mathbb{R}^3, \mathbf{v}_2 \in \mathbb{R}^3 \quad \text{blue arrow}$$

$$\mathbf{z}_i = [\mathbf{v}_1^T \mathbf{x}_i, \mathbf{v}_2^T \mathbf{x}_i]^T, \mathbf{z}_i \in \mathbb{R}^2$$

PCA

- Input

✓ data must be **centered**

$$X = \begin{bmatrix} | & & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_n \\ | & & | \end{bmatrix}_{d \times n}$$

$$\frac{1}{n} \sum_{i=1}^n c_i = 0, \quad \text{for all rows } \mathbf{c} \text{ in } X$$

- Output

✓ orthonormal eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_k$ (principal components)

First principal component

$$\|\mathbf{v}_1\|_2 = 1 \quad \arg \max_{\mathbf{v}_1} \sum_{i=1}^n (\mathbf{x}_i^T \mathbf{v}_1)^2$$

$$\arg \max_{\mathbf{v}_1} \mathbf{v}_1^T \left[\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \right] \mathbf{v}_1$$
$$XX^T = \Sigma$$

solution \mathbf{v}_1 is the first eigenvector of XX^T

Second principal component

$$\|\mathbf{v}_2\|_2 = 1$$
$$\mathbf{v}_2^T \mathbf{v}_1 = 0 \quad \arg \max_{\mathbf{v}_2} \sum_{i=1}^n (\mathbf{x}_i^T \mathbf{v}_2)^2$$

$$\arg \max_{\mathbf{v}_2} \mathbf{v}_2^T (XX^T) \mathbf{v}_2$$

solution \mathbf{v}_2 is the second eigenvector of XX^T

Other principal components will follow the same idea

PCA algorithm

- Assuming X and K as inputs
 - ✓ Center the data in X
 - ✓ Compute eigendecomposition ($V\Lambda V^T$) of XX^T
 - ✓ Return the top K eigenvectors from V
- Eigenvectors can then be used for projecting the data into lower dimensions

Google colab