Modul - Unsupervised and Reinforcement Learning (URL)

Bachelor Programme AAI

05 - Principal Component Analysis (PCA)

Prof. Dr. Marcel Tilly

Faculty of Computer Science, Cloud Computing

Vers. 21.11.2022

Agenda

Code can be found in PCA.ipynb on GitHub or hosted on myBinder

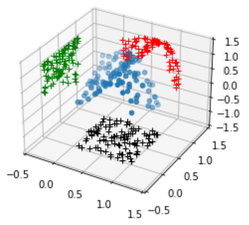
- PCA algorithm
- Details on PCA
- Apply principal component analysis (PCA) when solving problems using scikit-learn
- Calculate Eigenvalues and Eigenvectors

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is an exploratory approach to reduce the data set's dimensionality to 2D or 3D, used in exploratory data analysis for making predictive models.

Principal Component Analysis is a linear transformation of data set that defines a new coordinate rule such that:

- The highest variance by any projection of the data set maps to the first axis.
- The second biggest variance on the second axis, and so on.



PCA Motivation

Let's start with a simple data set:

- We do have samples (columns) and variables (rows), so-called *feature*
- Examples:
  - Flowers , blood test, ...
  - Students and scores in math and programming
  - Businesses with market capitalization and number of employees

Sample

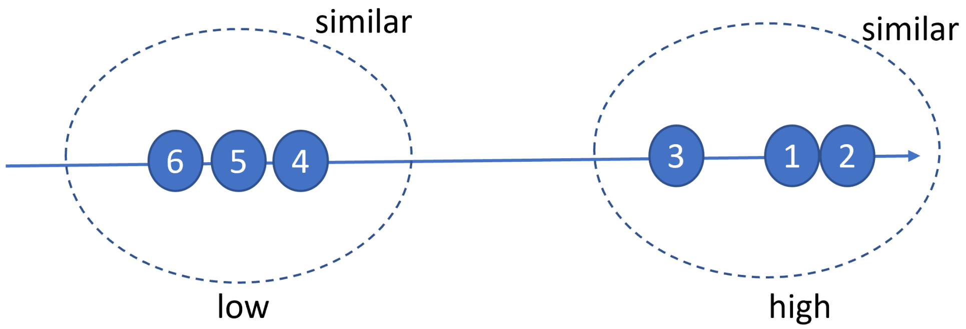
	Data1	Data2	Data3	Data4	Data5	Data6
Variable 1	10	11	8	3	2	1
Variable 2	6	4	5	3	3	0.5
...						

Feature

1-dim Clustering

In case we consider only 1-dimension, we can see how data clusters into *high* and *low* values.

	Data1	Data2	Data3	Data4	Data5	Data6
Feature 1	10	11	8	3	2	1

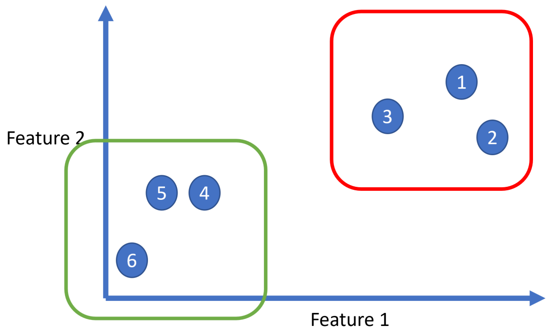


2-dim Clustering

In case we consider 2-dimensions, we can still see how data clusters.

- Feature 1 becomes the x-axis
- Feature 2 becomes the y-axis

	Data1	Data2	Data3	Data4	Data5	Data6
Feature 1	10	11	8	3	2	1
Feature 2	6	4	5	3	3	0.5



3-dim would work the same way. BUT: What about 4 and more features?

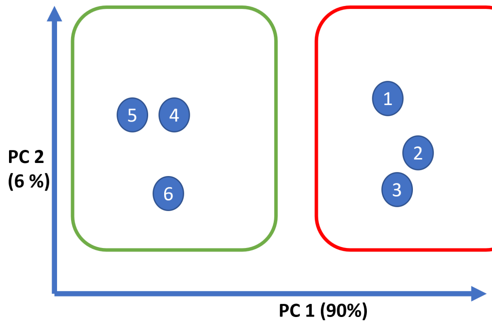
PCA

We are going to talk about how PCA can take 4 or more features and extract the main *Principal Components* out of them.

It will tell us…

1. … how does data cluster together?
2. … which feature is the prominent feature?
3. … how accurate the PCA result is?

	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
F 1	10	11	8	3	2	1
F 2	6	4	5	3	3	0.5
F 3	12	9	10	3.1	2.5	1.7
F 4	5	6	7	2	3	2
…						



Methodology (simplified!)

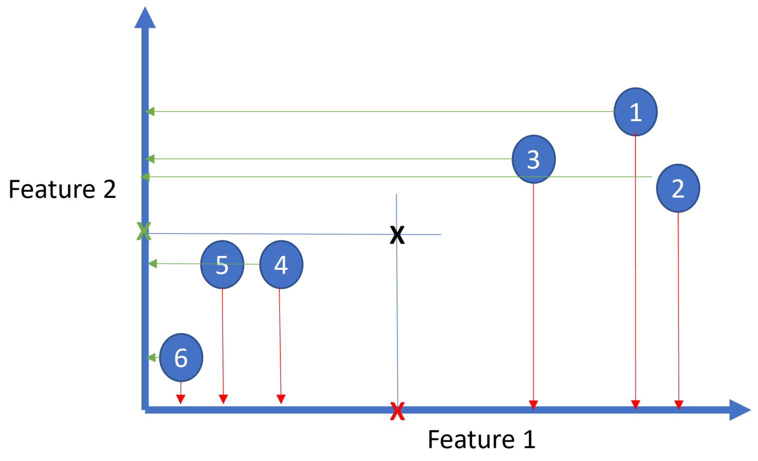
- Start from m x n data matrix X
  - Standardize and **Recenter**
- Compute covariance matrix
- Find eigenvectors and eigenvalues
- Principal components
  - PC1: eigenvector with highest eigenvalues
  - PC2: eigenvector with highest eigenvalues

Step-by-step: 1

Let´s consider 2-dim data set (m x n data matrix): data samples in columns and features in a row!

1. Calculate the mean of F1 and F2 to identify new *center*

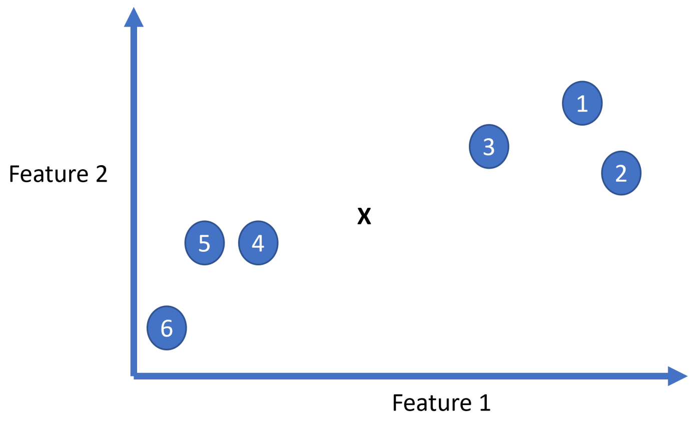
	Data1	Data2	Data3	Data4	Data5	Data6
Feature 1	10	11	8	3	2	1
Feature 2	6	4	5	3	3	0.5

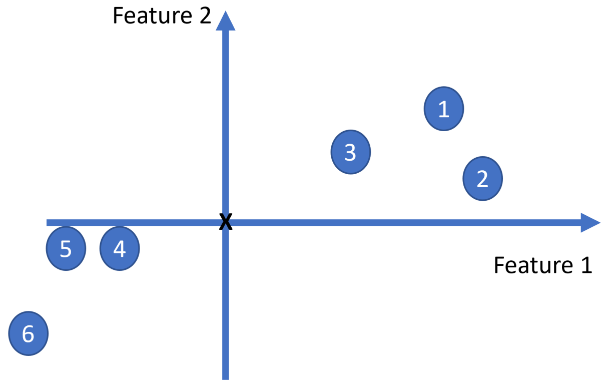


Step-by-step: 2

1. Now recenter the data.

> This does not change the dataset's *nature*. The relative position of data points remains.

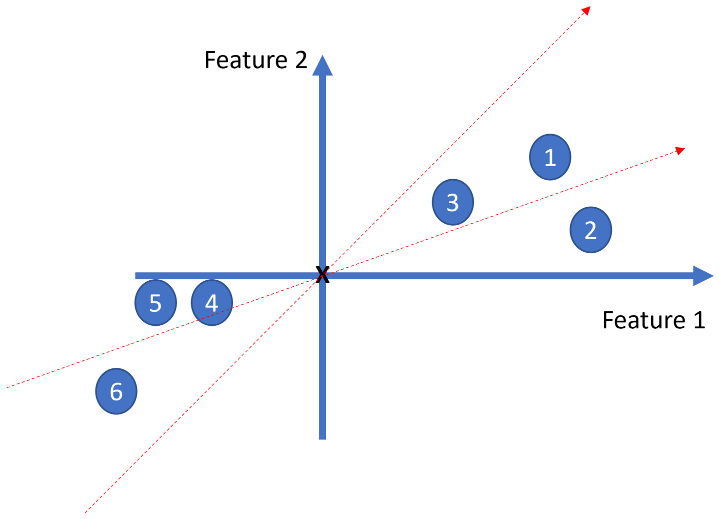


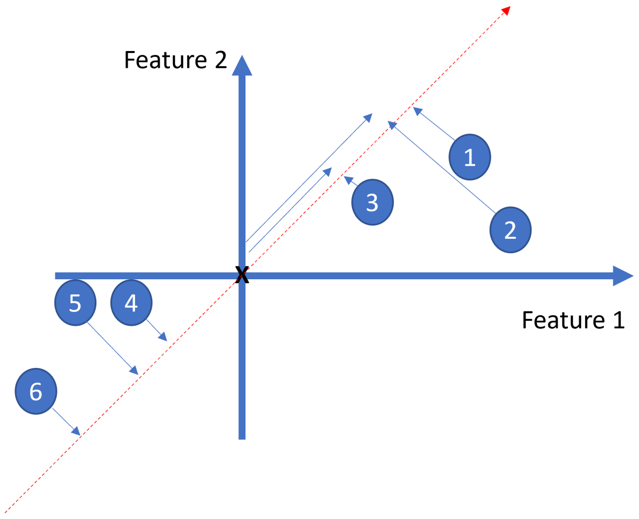


Step-by-step: 3

1. Fit a line through the data; this line goes through the origin

- It can be found by either
  - **minimize** the distance of each point to the line
  - or **maximizing** the distances from the projected points to the origin





Step-by-step: 4

PCA finds the best fitting line by **maximizing** the *sum of the squared distances* from the projected points to the origin!

- We can express the relation between a,b, and c via the Pythagorean theorem:

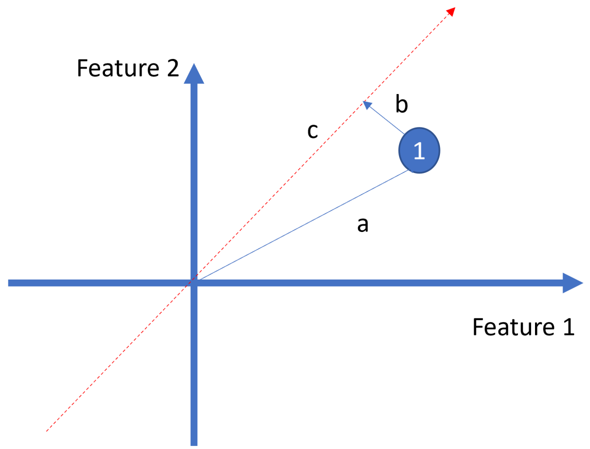
$$a^2 = b^2 + c^2$$

- a is constant since the point does not move

=> if *b* gets bigger *c* gets smaller

(and vice-versa)

=> we can either **minimize** b or **maximize** c

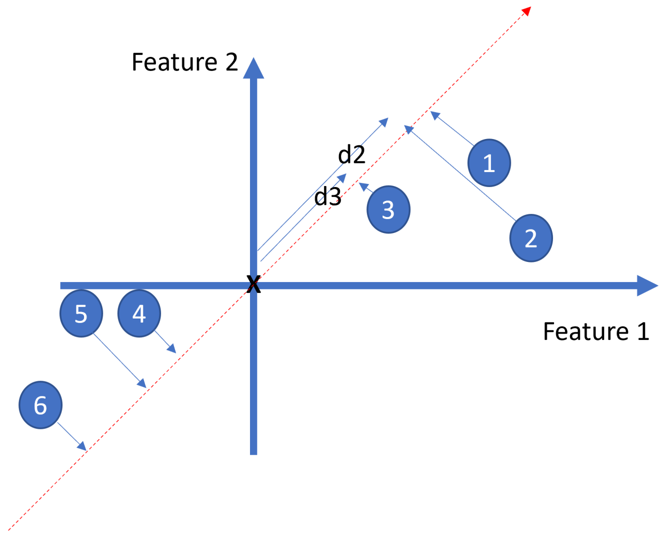


Step-by-step: 5

- PCA projects the data onto the line and then measures the distance from this projected point to the origin (called d_i)
- Use squares to avoid canceling out negative values
- Sum them up
- Find the line with the largest sum of squares!

$$ss_j = \sum_i d_i^2$$

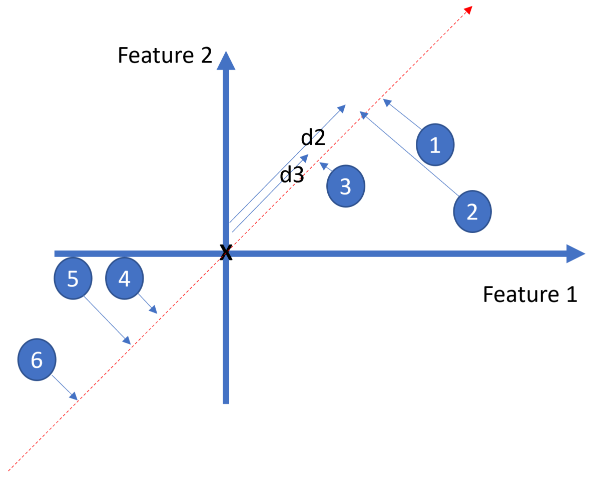
with j = feature and i=sample



Step-by-step: 6

> The vector PC1 is called the **Eigenvector** The sum of the squares of the distances are called the **Eigenvalues** of PC1

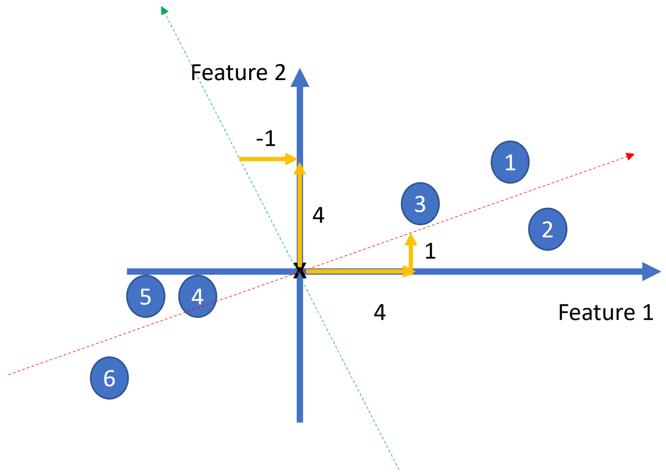
- The line with the larges *sum of squares* is called the *PC1*
- Here:
    - PC1 is a *linear combination* of F1 and F2 => $PC1 = [4, 1]$ (4 units of feature1 and 1 unit of feature2)
    - This means that data is mostly spread out via F1 axis.
    - We can normalize PC1 by dividing through the length => $PC1 = [0.97, 0.242]$



Step-by-step: 7

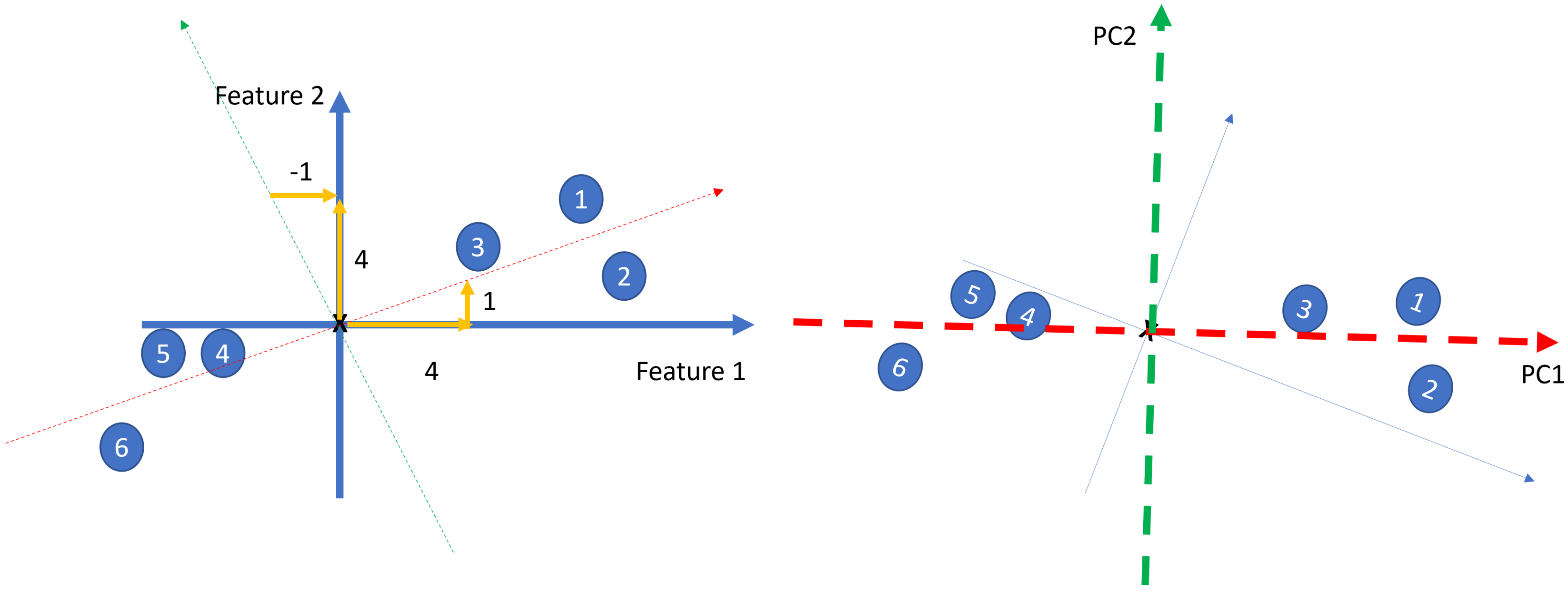
> This is the **Eigenvector** of PC2 The sum of the squares of the distances are the **Eigenvalues** of PC2

- PC2 is the line which is *orthogonal* to PC1!
- Here:
  - PC2 is a *linear combination* of F1 and F2 => $PC1 = [-1, 4]$
  - This means that data is mostly spread out via F2 axis.
  - We can normalize PC1 by dividing through the length => $PC2 = [-0.242, 0.97]$



Final Plot

Just rotate PC1 and PC2 and convert points into PC1/PC2 coordinates!



Variance

Eigenvalues:

- We have used *squared sum* , e.g. ss_1 and ss_2, of the distances of the projected points to the origin
- This can be converted to *variation* by dividing by the sample size minus 1 (n-1)

> $var_1 = ss_2/n - 1$ and $var_2 = ss_2/n - 1$

Example:

- PC1 = 15 and PC2 = 3 => total variation is PC1+PC2 = 18
- PC1 accounts fr 15/18=0.83= 83% of the total variation
- PC2 accounts for 3/18=0.17= 17%

Variance and Covariance

The variance along x is defined as:

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

This can be extended to x and y as

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

If we have k features those values can be summarized in a matrix, called *covariance matrix* C:

$$C_{n,m} = \sigma(x_n, x_m)$$

Covariance Matrix

$$C(x,y,z) = \begin{bmatrix} var_x & covar_{x,y} & covar_{x,z} \\ covar_{y,x} & var_y & covar_{y,z} \\ covar_{z,x} & covar_{z,y} & var_z \end{bmatrix}$$

If x is positively correlated with y, y is also positively correlated with x.

=> In other words, we can state that $\sigma(x,y) = \sigma(y,x)$

=> Therefore, the covariance matrix is always a symmetric matrix with the variances on its diagonal and the covariances off-diagonal.

> Symmetric Matrices have real Eigenvalues
>
> Symmetric Matrices are always *diagonalizable*

$$C = VDV^{-1}$$

Covariance Properties 1/2

- Because the covariance matrix is symmetric, the eigenvectors are orthogonal to each other.
- The eigenvalues and eigenvectors come in pairs. Such a pair is known as an eigenpair. The number of eigenpairs is equal to the number of rows or columns in the covariance matrix. For example, in a 4 x 4 covariance matrix, there are 4 eigenvalues and 4 eigenvectors.
- The eigenvectors of the covariance matrix represent the principal components (directions of maximum variance).
- The eigenvalues of the covariance matrix define the magnitude of eigenvectors. To get the top eigenvectors, we have to sort the eigenvalues in descending order.

Covariance Properties 1/2

- The amount of variance explained by an eigenvector can be obtained by getting the ratio (fraction) between the corresponding eigenvalue and the sum of all eigenvalues.
- The covariance matrix of standardized data is almost equal to the correlation matrix of the non-standardized, original data.
- The diagonal in the covariance matrix of standardized data always contains 1s.
- The sum of all eigenvalues of the covariance matrix is equal to the sum of the diagonal elements in the covariance matrix.

Eigenvectors

- The eigenvectors V represent the directions of the largest variance of the data
- The eigenvalues λ represent the magnitude of this variance in those directions.

$$CV = \lambda V$$

> Find eigenvectors and values of C

Summary

- Today we talked about the need for dimensionality reduction
  - various approaches
  - PCA and the math behind
- Applications

> Code can be found in PCA.ipynb on GitHub or hosted on myBinder

Exercise

- On GitLab: https://inf-git.fh-rosenheim.de/aai-url/05_exercise
- A bit of math
- Work on the correlation and covariance
- Try PCA