‘Data Analysis’
Computer Lab Session
Principal Component Analysis

Master 1 MLDM / CPS^2 / COSI / 3DMT
Saint-Étienne, France

Ievgen Redko

1 Outline

1. Introduction to Python (1/2)
2. Introduction to Python (2/2)
3. Probability, Random Variables and Probability Distributions
4. Linear Algebra (1/2)
5. Linear Algebra (2/2)
6. **Principal Component Analysis**
7. Linear Regression (1/2)
8. Linear Regression (2/2)
9. Clustering (1/2)
10. Clustering (2/2)

Outcome

The objective of this lab is to become familiar with Python functions for working with Principal Component Analysis (PCA).

2 Introduction: « Mon Violon Tombe Mais Je Sauve Une Note »

How can we represent high-dimensional data?

Three important features of the planets in our solar system are (cf. Marc Sebban’s lesson):

a) the distance to the Sun

b) the equatorial diameter

c) the density.

	Distance	Diameter	Density
Mercury	0.387	4878	5.42
Venus	0.723	12104	5.25
Earth	1.000	12756	5.52
Mars	1.524	6787	3.94
Jupiter	5.203	142800	1.31
Saturn	9.539	120660	0.69
Uranus	19.180	51118	1.29
Neptune	30.060	49528	1.64
Pluto	39.530	2300	2.03

We will load the planet data stored in an .csv file using the following piece of code:

```
import csv

names = []
data = []
with open('planets.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        names.append(row[0])
        data.append(row[1:])

feat_names = data[0]
data.pop(0)
names.pop(0)
```

Pairwise Representation

The dataset has now 3 numerical variables. We can represent the data 2 by 2 by using the pairwise representation with the Python function pairplot() seen in the previous lab sessions. Note that this function applies only to a special data structure called DataFrame defined in pandas toolbox.

It is difficult to make a relevant conclusion just by seeing this plot. With the diameter and distance variables, some points are concentrated on an unique set (a cluster).

This is due to the astronomical scale. We can fix this problem by taking into account the log transform of each variable using functions. This can be done using function apply() that can be called for a given DataFrame.

This new representation is better than the previous one, but we have many plots to study (if we have d variables, we can have $d \times (d-1)$ or at least $\frac{d \times (d-1)}{2}$ plots to see), and we don't have the information of the name of the planets.

2D Representation and Plot Size

In this context, we have one dimension representing the diameter of the planets (the second variable). We can use this information for changing the size of the markers and represent the data with the two other variables for the X- and Y-axes.

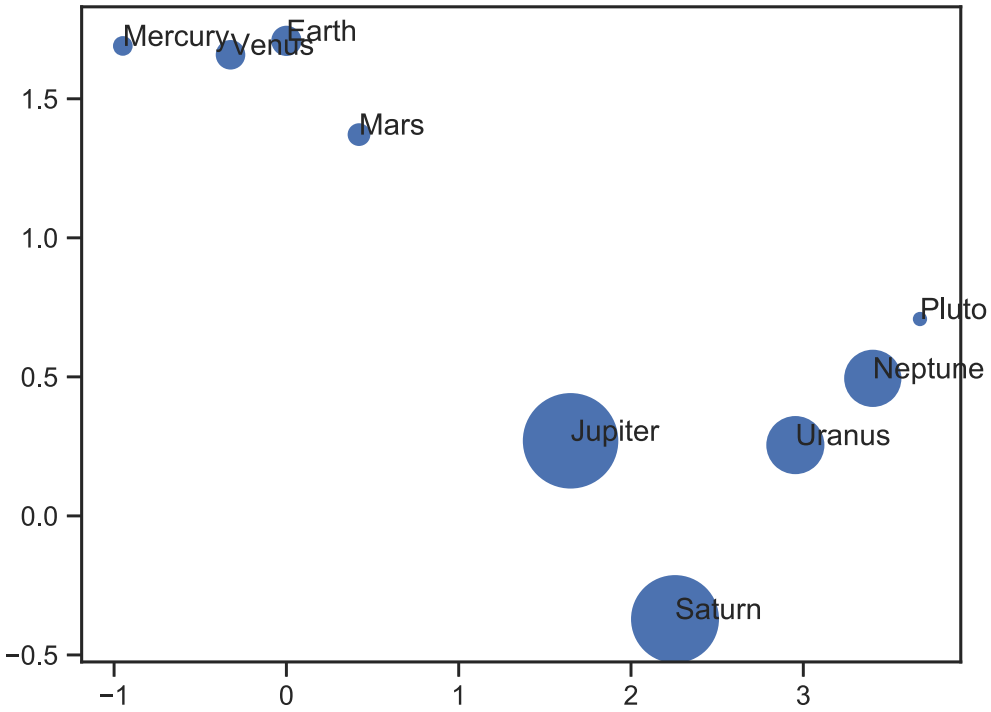

FIG. 1 – *2D Representation of the Planets.*

This plot (Figure 1) is a little misleading: we can see two groups of objects: one on the top-left and the other on the bottom-right. The diameters of the planets, represented by the size of the circles, is not presented in the same way of the two others.

3D Representation

In Python, we can easily represent the data in 3-dimensions with the mpl_toolkits.mplot3d package.

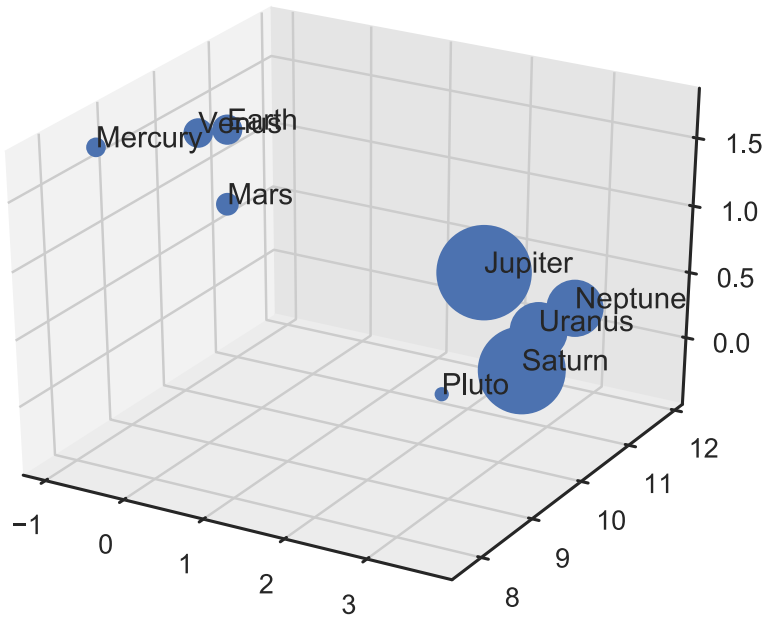

FIG. 2 – *3D Representation of the Planets.*

```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(...)
```

The plot obtained is an interactive one: we can rotate the cube for having another view of the scatterplot.

On the Figure 2, we can see that they are two clusters (one with Mercury, Venus, Earth and Mars, and another cluster with Jupiter, Saturn, Uranus and Neptune) and an unique point.

In statistics, this kind of point, far from the other ones in the representation space, is denoted as “outlier”.

This outlier corresponds to Pluto. Note that in 2006 the International Astronomical Union (IAU) excluded Pluto to the list of the planets and reclassified it as a member of the “dwarf planet” category.

Representation with Dimensionality Reduction

In conclusion, even if the 3D representation works well for this dataset, we must consider the case where there are more variables than only 3. In this context, we must proceed to a dimensionality reduction: finding a low-dimensional representation (a model) for high-dimensional data.

With the linear transformation, we can obtain a more compact representation: it is the so called “Principal Component Analysis” (PCA). This analysis is widely used for dimensionality reduction (projecting data points of a d-dimensional space onto a M-dimensional subspace) and for feature extraction.

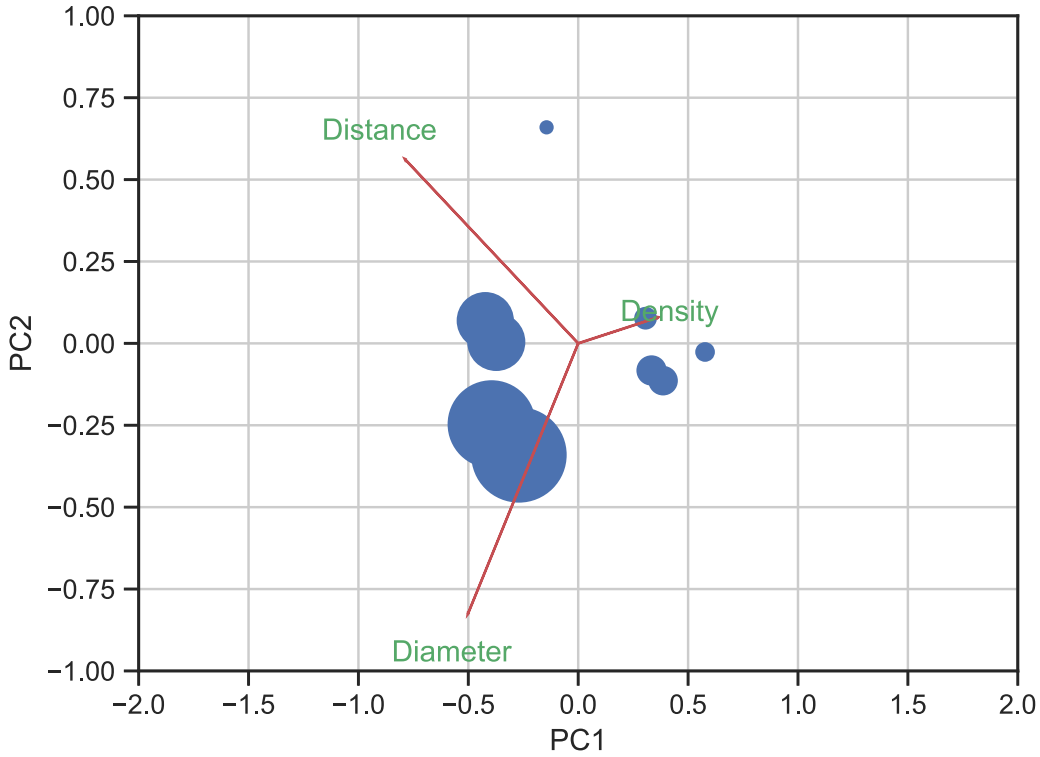

FIG. 3 – *Representation of the Planets with the 2 first components.*

In Python, we can easily compute a PCA with the sklearn.decomposition.PCA function. It fits a principal components analysis on to a given data matrix and returns an object carrying the information about many attributes related to PCA.

Use the provided myplot function to visualize the directions of the principal components and the data projection.

```
pcamodel = PCA()
pca = pcamodel.fit_transform(np.log(data))
```

We can easily see the two clusters (corresponding to the terrestrial planets on the right and the gas giants on the left) and the outlier (Pluto, on the top).

3 PCA on a 2-Dimensional Dataset, Step by Step

The calculation of PCA is done using numpy.linalg.eig() (for computing eigenvalues and eigenvectors of matrices) on the correlation or covariance matrix numpy.cov(), as determined by numpy.corrcoef() (for computing the correlation).

Compute the PCA for $X = \{(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)\}$ by implementing the following steps.

1. Plot the data in the original 2D-space.

Note that when plotting one can use matplotlib.axes.Axes.set_aspect to set the aspect ratio y/x. If it is a finite positive value then the window is set up so that one data unit in the x direction is equal in length to asp $\times$ one data unit in the y direction. The special case set_aspect$==' equal'$ produces plots where distances between points are represented accurately on screen.

```
fig = plt.figure()
ax = fig.add_subplot(111)
# plot something
ax.set_aspect('equal')
```

2. Compute the covariance matrix Σ from the zero mean values $(x - \overline{x})$.
   Reminder: $cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{n-1}$ or $cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{n}$ without correction.
   Or simply by using the Python function numpy.cov().
   The numpy.cov() function computes and prints
   $$C = \begin{pmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{pmatrix}$$
3. Solve the characteristic equation $det(\Sigma - \lambda I) = 0$ to get the eigenvalues and deduce the eigenvectors, then compute the 2 first components. In Python, we can simply use the numpy.linalg.eig() function.
4. Plot the scatterplot of the dataset, then the 2 first components. They will intersect perpendicularly (if this is not the case, apply the necessary aspect option).
5. Plot the biplot of the PCA obtained with the function PCA.

4 Exercise 2: PCA on an High-Dimensional Dataset

In a previous lab session, we have used a dataset carrying information about the students:

- Size (height in cm)
- Year of birth
- Number of days from the birthday to the present day)
- LatBP_NS (latitude decimal coordinate of the birth place = North-South axis)
- Long_BP_EW (longitude decimal coordinate of the birth place = East-West axis)

1. Load the Excel file (or CSV file) with this dataset in Python.
2. Print the information about the dataset by converting its ndarray version to a DataFrame and using the function describe().
3. Use pairplot() function to study the relation between the variables two by two. Are some variables more linked than others? Print the correlation coefficients for answering.
4. Compute the PCA of the dataset and check the explained variance of each component. How can you interpret the obtained results?
5. Plot the variances associated with the axes (i.e., the eigenvalues) by deducing them from the singular values. Print the table of the eigenvalues for each component with a confidence interval $\alpha = 95\%$.
6. Plot the biplot of the first two PCA components and discuss the obtained results.