Machine learning for dimension reduction

M. Mougeot

ENSIIE, November 2020

Contents

Introduction

Goal of the practical sessions

- To understand classification machine learning methods, from a methodological and practical point of view.
- To apply models and to tune the appropriate parameters on several data sets using the 'Python' language.
- To interprete 'Python' outputs.

Warnings and Advices

- The goal of this practical session is not "just to program with Python"" but more specifically to understand the framework of Modeling, to learn how to developp appropriate models for answering to a given operationnal question on a given data set. The MAL course belongs to the **Data Science courses** . For each MAL practical session, you should **first understand** the mathematical and statistical backgrounds, **then write your own program with 'Python'** to practically answer to the questions.

Instructions

- The practical work must be carried on with a 'group of two students'.
- The MAL project aims to developp a Jupyter notebook to solve a classification problem (the subject will be given soon). Your names have to be written in the first lines of the program file with comments. Please note that without this information, no grade will be attributed to the missing name project.

Python libraries

For this practical session, the following librairies need bo be uploaded in the python environment.

```
import numpy as np
import csv
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

Principal component Analysis (PCA) is a tool to investigate multi dimensional data. This method leads to project the initial raw data on orthogonal factorial plans, which maximize the variance of the projected data. This method is frequently used to represent the data in low dimension, to investigate the distribution of multi dimensional data, to discover features or clusters in the data.

Principal component Analysis (PCA)

The data

The data corresponds to a set of signals recorded from an industrial process and are quite "typical" signals. Each signal represents the evolution of a temperature recorded from a sensor for a motor on a bench test or for chemical batches.

The file `sigref.csv` contains a set of $n = 100$ signals sampled over $p = 120$ points.

The main objective of this practical session is first to design a monitoring and diagnosis approach using the reference signals then to test the methods "in real life" with other signals.

Load the set of signals of the `sigref.csv` file using the following instructions: The $n = 100$ signals are then stored in a matrix of size $(n = 100, p = 120)$.

```
tab = pd.read_csv("sigref.csv",header=None);
tab=np.array(tab);
n=len(tab);
```

-Visualize the set of temporal signals using a different color for each signal.

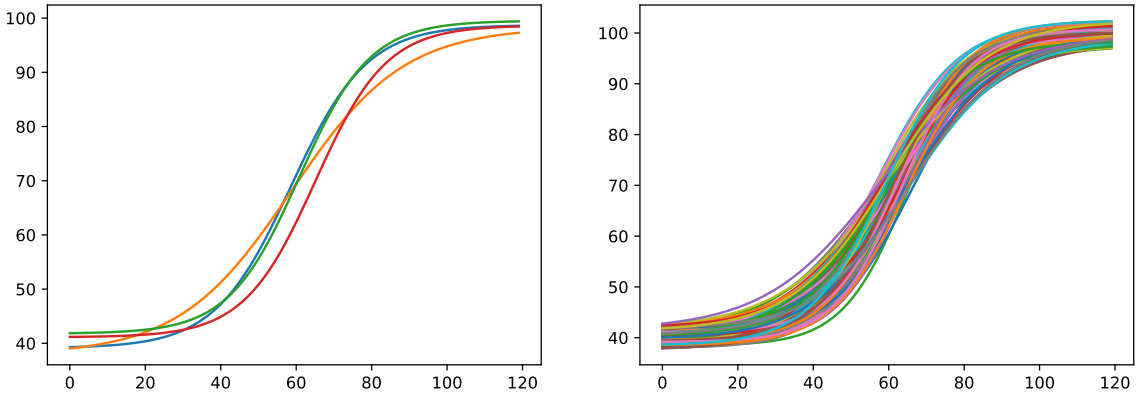

Figure 1: Signal evolution (left: 5 arbitrary signals; right: all signals.

PCA on functional data

Notations: v^k denotes the eigenvector corresponding to the eigenvalue λ_k, with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ of the variance covariance matrix of the centered signals $s_1 \ldots, s_n$.

y_j^k denotes the projection of the centered signal s_j on the k^{th} principal axis: $y_j^k =< s_j, v^k >$.

-Using the Python `PCA()` function, compute a PCA of the initial *centered* signals.

-Compute the percentage of variance explained by each principal axis.

-How many factorial axes should you keep to explain 99% of the total variance ?

-Conclusion.

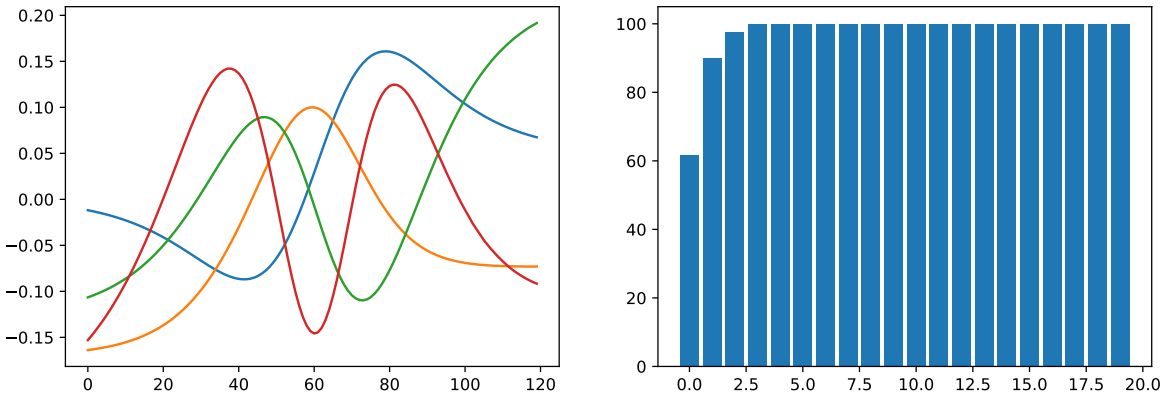

Figure 2: First 4 Eigenvectors (left) and pourcentage of the cumulative variance of the PCA axes (right)

Signal compression

-Plot the first four eigenvectors. Based on an interpretation of this graph, characterize the principal components (coefficients of the projected centered signals).

-For a given centered signal, for example s_j , $1 \leq j \leq n = 100$, compute the succesive reconstruction of the signal called $\hat{s}_j^k$ of the signal using only 1, 2, 3, 4 and $k = 5$ coefficients.

$\hat{s}_j^k = \bar{s} + \sum_{k=1}^{J} y_j^k v^k$ for $J = 1 \dots 4$.

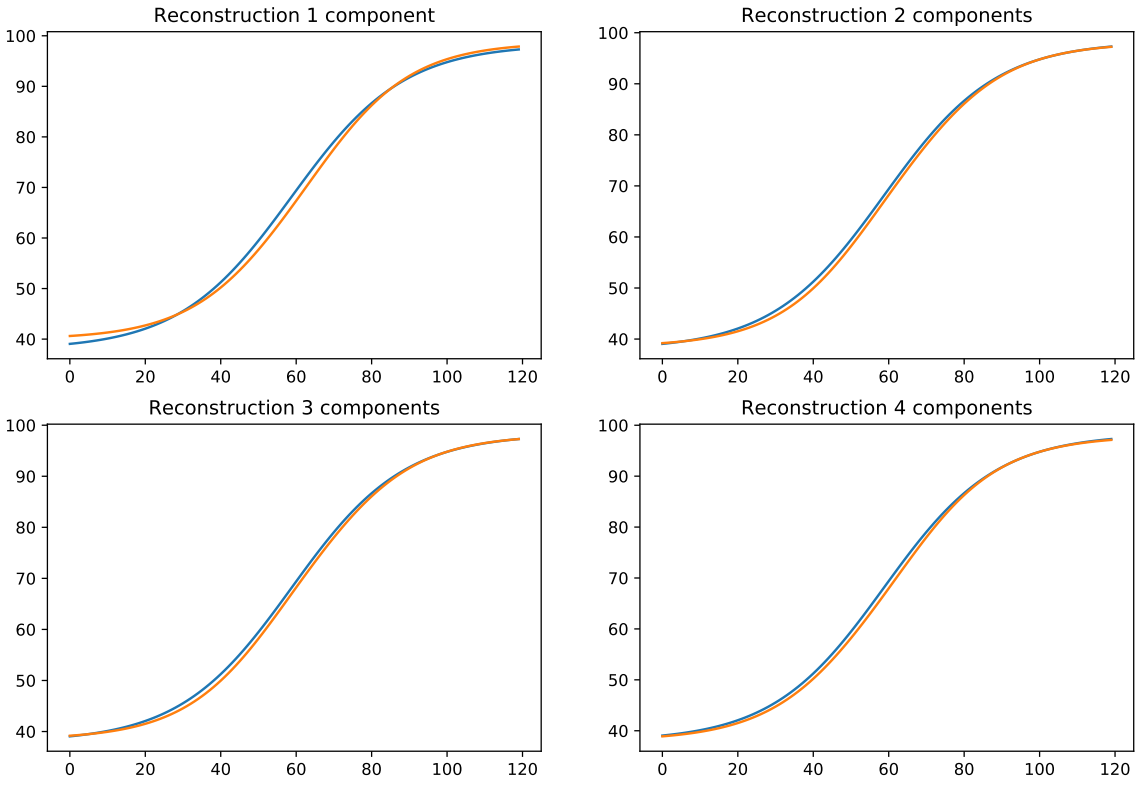

Figure 3: Reconstruction of signal 5 respectively with 1,2,3,4 components

-Compute the quadratic errors between each initial signal s_j and its reconstruction

$\hat{s}_j^k$ for $1 \leq j \leq n$.

-Conclusion. What are the benefits of this approach ?

Parametric modeling of the principal components

-Visualize on a graph the projections, y_j^k of the $n = 100$ initial signals on the first factorial plan.

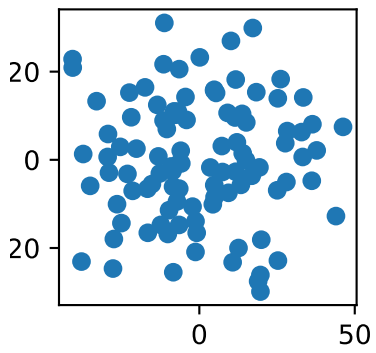

Figure 4: Projections of the two first principal components

-Analyze the statistical distribution on the first $K = 4$ axes with the help of an empirical histogram and a kernel estimation of the density.

-Show that the empirical densities are gaussian using the Henry line tool (instructions `qqplot()`, `qqline()`).

```
import numpy as np
import pylab
import scipy.stats as stats
data=x_pca[:,0];
stats.probplot(data, dist="norm", plot=pylab)
pylab.show()
```

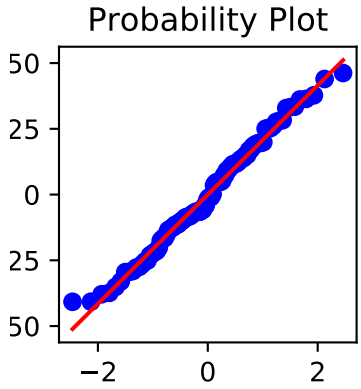

Figure 5: QQplot

-Using the Shapiro Test, test if the empirical distributions of the 4 principal components are gaussian where 'x_pca' is the array containing the principal components values

```
from scipy.stats import shapiro
data=x_pca[:,0];
stat, p = shapiro(data)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print("Data follows Normal Distribution")
else:
    print("Data does not follow Normal Distribution")
```

Diagnosis using the projected coefficients

-Z is the variable defined by $Z = \sum_{j=1}^{J} (Y_j)^2$ with $Y_j \sim \mathcal{N}(0,1)$. If the Y_j are Gaussian variables, , Z is a Chi-square variable with J ddl..

-Compute for each signal k, the $Proba(Z \leq z_k)$ using the instuction . With a risk $\alpha = 1\%$ (or with a confidence interval $1 - \alpha$, provide a test on each signal. Conclusion.

Diagnostis of a new family of signals

-Use the `sigdiag.txt` files to diagnose new signals based on this approach.