TD 4-5

September 30, 2024

Exercise 1 : Canonical correlation analysis

Load Wine and exercise/physiological datasets :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_wine, load_linnerud

#Loading data in a DataFrame:
data_wine = load_wine(as_frame=True)
data_physio = load_linnerud(as_frame=True)

#Loading wine features in a matrix:
X_wine = data_wine.data
#Loading wine classes in a vector:
y = data_wine.target

#Loading exercise/physio. data
X_ex = data_physio.data  # Exercise variables
X_physio = data_physio.target  # Physiological variables
```

1 Plot the correlation matrix of ex./physio. dataset using *corr()* method on the DataFrame and *heatmap* function of seaborn library.

We want to study relations between variables that determines wine TASTE, and variables that determines wine ASPECT.

- TASTE variables :

  ```
  ['alcohol' 'od280/od315_of_diluted_wines' 'proline']
  ```

- ASPECT variables :

  ```
  ['color_intensity' 'hue']
  ```

2 Create a new DataFrame selecting only these features and plot the corresponding correlation matrix.

3 Code a function that computes empirical covariance matrices Σ_{XX}, Σ_{YY} and Σ_{XY}.

4 Use it to code your own CCA function that takes both matrices X and Y as input and returns Canonical variables for group X, Canonical variables for group Y and canonical correlations.

5 Apply it on both datasets and plot new correlation matrices on canonical variables.

6 Plot both dataset samples on first canonical variables of each group.

7 Add a straight line of λ_1 coefficient, the maximum eigenvalue of the CCA. What do you observe ?

8 Based on this observation, design a linear estimator between first canonical variable of group X and first canonical variable of group Y. Compute the mean squared error for this estimator for both datasets.

9 Fill the following function with correlations between original variables and canonical variables to draw correlation circles. Plot them on both datasets and comment.

```
def plot_correlation_circles( X, Y,X_c,Y_c, X_names= None, Y_names = None, X_can = None, Y_can= None):

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8))

    # Correlations for X and Y original variables with X canonical variables

    ### ...TO FILL...
    # x_loadings_x = ...
    # y_loadings_x = ...
    ###

    # Correlations for X and Y original variables with Y canonical variables

    ### ...TO FILL...
    # x_loadings_y = ...
    # y_loadings_y = ...
    ###



    # Plot relative to X canonical variables
    ax1.set_title('Correlation Circle (X Canonical Variables)')
    for i in range(X.shape[1]):
        ax1.arrow(0, 0, x_loadings_x[i, 0], x_loadings_x[i, 1], head_width=0.05, head_length=0.05, fc='
        if X_names is None:
            ax1.text(x_loadings_x[i, 0]*1.15, x_loadings_x[i, 1]*1.15, f'X{i+1}', color='blue')
        else:
            ax1.text(x_loadings_x[i, 0]*1.15, x_loadings_x[i, 1]*1.15, X_names[i], color='blue')
    for i in range(Y.shape[1]):
        ax1.arrow(0, 0, y_loadings_x[i, 0], y_loadings_x[i, 1], head_width=0.05, head_length=0.05, fc='
        if Y_names is None:
```

```
        ax1.text(y_loadings_x[i, 0]*1.15, y_loadings_x[i, 1]*1.15, f'Y{i+1}', color='red')
    else:
        ax1.text(y_loadings_x[i, 0]*1.15, y_loadings_x[i, 1]*1.15, Y_names[i], color='red')

ax1.add_artist(plt.Circle((0,0), 1, color='gray', fill=False))
ax1.axhline(y=0, color='k', linestyle='--')
ax1.axvline(x=0, color='k', linestyle='--')
ax1.set_xlim(-1.2, 1.2)
ax1.set_ylim(-1.2, 1.2)
ax1.set_aspect('equal')
if X_can is not None:
    ax1.set_xlabel('X ('+X_can+') CC1')
    ax1.set_ylabel('X ('+X_can+') CC2')
else:
    ax1.set_xlabel('X CC1')
    ax1.set_ylabel('X CC2')
# Plot relative to Y canonical variables
ax2.set_title('Correlation Circle (Y Canonical Variables)')
for i in range(X.shape[1]):
    ax2.arrow(0, 0, x_loadings_y[i, 0], x_loadings_y[i, 1], head_width=0.05, head_length=0.05, fc='
    if X_names is None:
        ax2.text(x_loadings_y[i, 0]*1.15, x_loadings_y[i, 1]*1.15, f'X{i+1}', color='blue')
    else:
        ax2.text(x_loadings_y[i, 0]*1.15, x_loadings_y[i, 1]*1.15, X_names[i], color='blue')
for i in range(Y.shape[1]):
    ax2.arrow(0, 0, y_loadings_y[i, 0], y_loadings_y[i, 1], head_width=0.05, head_length=0.05, fc='
    if X_names is None:
        ax2.text(y_loadings_y[i, 0]*1.15, y_loadings_y[i, 1]*1.15, f'Y{i+1}', color='red')
    else:
        ax2.text(y_loadings_y[i, 0]*1.15, y_loadings_y[i, 1]*1.15, Y_names[i], color='red')

ax2.add_artist(plt.Circle((0,0), 1, color='gray', fill=False))
ax2.axhline(y=0, color='k', linestyle='--')
ax2.axvline(x=0, color='k', linestyle='--')
ax2.set_xlim(-1.2, 1.2)
ax2.set_ylim(-1.2, 1.2)
ax2.set_aspect('equal')
if Y_can is not None:
    ax2.set_xlabel('Y ('+Y_can+') CC1')
    ax2.set_ylabel('Y ('+Y_can+') CC2')
else:
    ax2.set_xlabel('Y CC1')
    ax2.set_ylabel('Y CC2')

plt.tight_layout()
plt.show()
```

Exercise 2 : Factorial correspondence analysis

1 Write contingency tables seen in course into DataFrames.

2 Code a function to compute conditional probabilities estimates.

3 Code a function to compute standardized residuals and plot them.

4 Code a function that compute χ^2 distance and apply it.

5 Comment by choosing a quantile threshold α and comparing it with observed χ^2 distances.

6 Write a function that performs FCA on two variables, taking in input the contingency table and returning eigenvectors, singular values and projected row/column profiles.

7 Plot row/column profiles projection on the two first factors.

8 Plot the cumulative χ^2 distance ratio.

To illustrate the Guttman effect we consider a new dataset containing popularity of baby names by years from 1952 to 2023 in Australia.

9 Apply FCA on these data and plot row/column profiles projection on the two first factors.

10 Plot row/column profiles projection now on first and third factors.

11 What do you observe and how can you explain that ?