Data Analysis - Lab 3

M. Sébastien MASCHA & M. Sauvage Pierre

ISEP Paris – September 24th, 2019

Import of libraries

This document has been done using python on Jupyter Notebook with the librairies:

- maths for sqrt, pi, exp
- Numpy to manipulate arrays
- pandas to import csv
- matplotlib to plot graphics
- seaborn to make your charts prettier (built on top of Matplotlib)
- sklearn : tools for data mining and data analysis

In [1]:

```
# coding: utf-8

import data

from math import sqrt,pi,exp
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns; sns.set()

import sklearn
# Normalize data
from sklearn.preprocessing import StandardScaler
# Dimension reduction
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
# Useful
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

import mlxtend
from mlxtend.plotting import plot_pca_correlation_graph
```

Exercice B.1 - MDS, LLE and Isomap

We are now going to study the "golub" data. These data contains the level of expression of 7129 genes based on 72 sample. Each sample is linked to a variant of Leukemia: AML(25) and ALL(47). Our goal is to visualize the 72 samples on a 2D plan.

Question 1 - Load the file Golub_data

We use the semicolon separator because we saw in the text editor that the data was separated by semicolon.

In [8]:

```
df = pd.read_csv("data/golub_data.csv", sep=',')
df.head()
```

Out[8]:

	Ex 1	Ex 2	Ex 3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8	Ex 9	Ex 10	...	Ex 63	Ex 64	Ex 65	Ex 66	E 6
G 1	-214	-139	-76	-135	-106	-138	-72	-413	5	-88	...	-90	-137	-157	-172	-4
G 2	-153	-73	-49	-114	-125	-85	-144	-260	-127	-105	...	-87	-51	-370	-122	-44
G 3	-58	-1	-307	265	-76	215	238	7	106	42	...	102	-82	-77	38	-2
G 4	88	283	309	12	168	71	55	-2	268	219	...	319	178	340	31	39
G 5	-295	-264	-376	-419	-230	-272	-399	-541	-210	-178	...	-283	-135	-438	-201	-35

5 rows × 72 columns

Question 2 - Open the label in the file Golub_class2.

Once again, be very careful with the opening parameters.

In [7]:

```
df2 = pd.read_csv("data/golub_class2.csv", sep=',',header=None)
df2.shape
```

Out[7]:

```
(72, 2)
```

Question 3 - Run a PCA on the Golub data, and visualize the result.

In [ ]:

```
def circleOfCorrelations(pc_infos, ebouli):
    fig = plt.figure(figsize = (12,12))
    plt.Circle((0,0),radius=10, color='g', fill=False)
    circle1=plt.Circle((0,0),radius=1, color='g', fill=False)
    fig = plt.gcf()
    fig.gca().add_artist(circle1)
    for idx in range(len(pc_infos["PC-0"])):
        x = pc_infos["PC-0"][idx]
        y = pc_infos["PC-1"][idx]
        plt.plot([0.0,x],[0.0,y],'k-')
        plt.plot(x, y, 'rx')
        plt.annotate(pc_infos.index[idx], xy=(x,y))
    plt.xlabel("PC-0 (%s%%)" % str(ebouli[0])[:4].lstrip("0."))
    plt.ylabel("PC-1 (%s%%)" % str(ebouli[1])[:4].lstrip("0."))
    plt.xlim((-1,1))
    plt.ylim((-1,1))
    plt.title("Circle of Correlations")

def myPCA(df, clusters=None):
    # Normalize data
    df_norm = (df - df.mean()) / df.std()
    # PCA
    pca = PCA(n_components='mle')
    pca_res = pca.fit_transform(df_norm.values)
    # Ebouli
    ebouli = pd.Series(pca.explained_variance_ratio_)
    ebouli.plot(kind='bar', title="Ebouli des valeurs propres")
    plt.show()
    # Circle of correlations
    coef = np.transpose(pca.components_)
    cols = ['PC-'+str(x) for x in range(len(ebouli))]
    pc_infos = pd.DataFrame(coef, columns=cols, index=df_norm.columns)
    circleOfCorrelations(pc_infos, ebouli)
    plt.show()
    return pc_infos, ebouli

golub_df_norm = (golub_df - golub_df.mean()) / golub_df.std()
figure, correlation_matrix = plot_pca_correlation_graph(golub_df_norm,
                                                        golub_class,
                                                        figure_axis_size=10)
```

Question 4 - Use the MDS function from the package MDS to run a MDS on these data.

Compare with the PCA result and comment.

In [ ]:

```
embedding = MDS()
X_transformed = embedding.fit_transform(golub_df.values[:100])
X_transformed.shape
```

Question 5 - Apply LLE to the Golub data, with 3, 5, 8, 10, 12 and 15 neighbors.

Using the LocallyLinearEmbedding package from sklearn.

Analyze the results and determine the best neighborhood model. Use the plt.subplot function rather than the regular plot function for this question.

```
In [ ]:
```

```
embedding = LocallyLinearEmbedding()
X_transformed = embedding.fit_transform(golub_df[:100])
X_transformed.shape
```

Question 6 - Same question for the Isomap() function from the Isomap package.

Use 4, 8, 10, 13, 16 and 20 neighbors instead.

Exercice B.2 - Alon

Load the Alon data, describe their content and apply the same algorithms as in the previous exercise.

Load the file alon.csv

```
In [70]:
```

```
alon_df = pd.read_csv("data/alon.csv", sep=';')
alon_df.head()
```

```
Out[70]:
```

	Hsa.3004	Hsa.13491	Hsa.13491.1	Hsa.37254	Hsa.541	Hsa.20836	Hsa.1977	Hsa.44472
0	15.161878	9.437886	7.228324	6.864347	3.073544	9.096942	3.388666	4.495803
1	13.131232	9.450446	6.686033	4.931565	2.367608	7.719571	5.128649	3.539225
2	4.658592	8.748834	6.667209	5.803129	1.199826	1.727416	1.406442	1.597463
3	10.563792	13.364874	10.047634	6.530421	3.026227	3.253446	2.188840	2.514740
4	7.022636	8.122147	7.426341	7.575234	4.537741	6.294042	4.272008	6.355163

5 rows × 2000 columns

Open the label in the file alon_class.

Once again, be very careful with the opening parameters.

In [69]:

```
# Open the class file
alon_class = pd.read_csv("data/alon_class.csv", sep=',')
alon_class.shape
```

Out[69]:

```
(62, 1)
```

Run a PCA on the Alon data, and visualize the result.

In [ ]:

```

```

Use the MDS function from the package MDS to run a MDS on these data.

Compare with the PCA result and comment.

In [60]:

```
embedding = MDS()
X_transformed = embedding.fit_transform(alon_df.values)
X_transformed.shape
```

Out[60]:

```
(62, 2)
```

Apply LLE to the Alon data, with 3, 5, 8, 10, 12 and 15 neighbors.

Using the LocallyLinearEmbedding package from sklearn.

Analyze the results and determine the best neighborhood model. Use the plt.subplot function rather than the regular plot function for this question.

In [2]:

```
embedding = LocallyLinearEmbedding(n_neighbors=5)
X_transformed = embedding.fit_transform(alon_df[:100])
X_transformed.shape

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 Component PCA', fontsize = 20)

targets = alon_class.x.unique()
colors = ['r', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = alon_class['x'] == target
    ax.scatter(alon_df.loc[indicesToKeep, 'principal component 1']
               , alon_df.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 40)
ax.legend(targets)
ax.grid()
```

```
---------------------------------------------------------------------
-------
NameError                                 Traceback (most recent cal
l last)
<ipython-input-2-a954ad8f1616> in <module>
      1 embedding = LocallyLinearEmbedding(n_neighbors=5)
----> 2 X_transformed = embedding.fit_transform(alon_df[:100])
      3 X_transformed.shape
      4 
      5 fig = plt.figure(figsize = (8,8))

NameError: name 'alon_df' is not defined
```

Same question for the Isomap() function from the Isomap package.

Use 4, 8, 10, 13, 16 and 20 neighbors instead.

In [ ]: