Institut National des Télécommunications et des Technologies de l'Information et de la Communication INT-TIC d'Oran Algérie

Analyse de Données

3ème année ingénieur

Fiche de TP N° 4

Méthodes Descriptives Multi-variées

Analyse en Composantes Principales

Dans un tableau de données quantitatives, constitué de lignes représentant les individus (les observations) et de colonnes représentant les variables, l'information contenue dans les données est souvent redondante. En d'autres termes, les variables sont liées entre elles. Cette liaison peut être mesurée statistiquement par la corrélation.

En analyse de données, l'**analyse en composantes principales** permet de transformer un ensemble de variables quantitatives corrélées, en de nouvelles variables décorrélées, appelées **composantes principales**. Cette transformation a les propriétés suivantes :

- Les composantes principales sont des combinaisons linéaires des variables d'origine.
- Le nombre de composantes principales est égal au nombre des variables d'origine (ce qui permet d'avoir un espace de même dimension).
- Elles sont décorrélées entre elles (ce qui permet d'avoir un espace orthogonal).
- Les composantes sont construites de sorte qu'elles résument au mieux l'information : la première composante est construite pour avoir une variance maximale (une meilleure distribution des données, une meilleure discrimination), la deuxième composante est ensuite construite pour être décorrélées avec la première et pour avoir la plus grande variance possible, et ainsi de suite. Les composantes ont donc une variance décroissante. Par conséquent, on peut ne retenir que les premières composantes et ignorer le reste afin d'avoir un espace de dimension réduite (réduction de dimensionnalité).
- L'inertie globale (la somme des variances de toutes les variables) est la même, mais distribuée différemment entre les nouvelles variables (les premières composantes auront un pourcentage important de cette inertie).

Les mêmes individus seront donc exprimés avec ces composantes, au lieu des variables d'origine.

Il est démontré que les vecteurs propres de la matrice de corrélation des variables d'origine vérifient ces propriétés. La première composante principale est le vecteur propre associé à la plus grande valeur propre, la deuxième composante principale est le vecteur propre associé à la deuxième plus grande valeur propre, et ainsi de suite. Plus la valeur propre est grande, plus l'inertie résumée par le vecteur propre associé à cette valeur propre est importante.

Ces aspects seront illustrés dans ce TP à travers un exemple.

Le fichier de données utilisé dans ce TP est le jeu de données « **iris** », connu également sous le nom de « **Iris de Fisher** ». Ce jeu de données contient des mesures de morphologie des fleurs d'iris de trois espèces : *Setosa*, *Versicolor* et *Virginica*. Le fichier de données, nommé `iris.csv`, contient les 5 variables suivantes (4 variables quantitatives et une variable qualitative), pour un échantillon de 150 individus :

`sepal.length`	longueur des sépales
`sepal.width`	largeur des sépales
`petal.length`	longueur des pétales
`petal.width`	largeur des pétales
`variety`	espèce

Les modules utilisés sont les mêmes utilisés dans les TPs précédents : `numpy`, `pandas` et `matplotlib`.

```
>>> import numpy as np
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> import os
```

Importation et préparation des données

```
>>> os.chdir('C:\\...\\...\\DataExamples')
>>> table = pd.read_csv('iris.csv', sep=',')

>>> table.head()                        # Affichage des premières lignes
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
```

Avant de commencer, nous allons isoler les noms des variables dans un vecteur nommé `var_names` (de taille 5x1), la variable `variety` dans un vecteur nommé `labels` (de taille 150x1), et les 4 variables quantitatives (`sepal.length`, `sepal.width`, `petal.length` et `petal.width`) dans une matrice nommée `data` (de taille 150x4). Pour cela, nous allons utiliser la structure de données `ndarray` de `numpy`.

```
>>> var_names = np.asarray(table.columns)      # Vecteur contenant les noms des
                                               # variables
>>> var_names
array(['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
       'variety'], dtype=object)



>>> labels = table.iloc[:,-1]                  # Extraction de la dernière colonne
                                               # (la variable "variety")
>>> labels = np.asarray(labels)                # Conversion en un ndarray de numpy
>>> labels.shape
(150,)
```

```
>>> data = table.iloc[:,:-1]                    # Extraction des 4 premières colonnes
                                                # (les variables quantitatives)
>>> data.head()
   sepal.length  sepal.width  petal.length  petal.width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2

>>> data = np.asarray(data)                     # Conversion en un ndarray de numpy
>>> labels.shape
(150, 4)
```

Standardisation des variables quantitatives

Avant de procéder au calcul des composantes principales, il convient standardiser (centrer et réduire) toutes les variables.

```
>>> mean = data.mean(axis=0)                    # Calcul de la moyenne des colonnes
>>> mean
array([5.84333333, 3.05733333, 3.758     , 1.19933333])

>>> std = data.std(axis=0)                      # Calcul de l'écart type des colonnes
>>> std
array([0.82530129, 0.43441097, 1.75940407, 0.75969263])

>>> data_cr = (data-mean)/std                   # Données centrées et réduites :
                                                # de chaque variable (colonne) on
                                                # soustrait la moyenne et on divise
                                                # par l'écart type


>>> data_cr.mean(axis=0)                        # Vérification
array([-4.73695157e-16, -7.81597009e-16, -4.26325641e-16, -4.73695157e-16])
                                                # Moyennes presque nulles

>>> data_cr.std(axis=0)
array([1., 1., 1., 1.])                         # Ecarts types égaux à 1
```

Calcul de la matrice de variance covariance

```
>>> cov = np.dot(data_cr.T, data_cr)/(n-1)      # Calcul de la matrice de variance
                                                # covariance


>>> cov = np.cov(data_cr.T)                     # Donne le même résultat

>>> cov
array([[ 1.00671141, -0.11835884,  0.87760447,  0.82343066],
       [-0.11835884,  1.00671141, -0.43131554, -0.36858315],
       [ 0.87760447, -0.43131554,  1.00671141,  0.96932762],
       [ 0.82343066, -0.36858315,  0.96932762,  1.00671141]])
```

On remarque que les variances sur la diagonale sont proches de 1, car les variables sont centrées et réduites. La matrice de variance covariance est dans ce cas très proche de la matrice de corrélation :

```
>>> cor = np.corrcoef(data.T)
>>> cor
array([[ 1.        , -0.11756978,  0.87175378,  0.81794113],
       [-0.11756978,  1.        , -0.4284401 , -0.36612593],
       [ 0.87175378, -0.4284401 ,  1.        ,  0.96286543],
       [ 0.81794113, -0.36612593,  0.96286543,  1.        ]])
```

Calcul des composantes principales

Les composantes principales sont les vecteurs propres de la matrice de variance covariance (ou de la matrice de corrélation). Les valeurs propres associées sont proportionnelles au pourcentage d'inertie des vecteurs propres correspondants : plus la valeur propre est grande, plus l'inertie du vecteur propre est importante.

Le calcul des vecteurs propres et des valeurs propres se fait comme suit :

```
>>> eigen_vals, eigen_vecs = np.linalg.eig(cov)

>>> eigen_vals                                  # Affichage des valeurs propres
array([2.93808505, 0.9201649 , 0.14774182, 0.02085386])

>>> eigen_vals/eigen_vals.sum()*100             # Pourcentage des valeurs propres
array([72.96244541, 22.85076179,  3.66892189,  0.51787091])
```

On remarque que les pourcentages des deux premières valeurs propres avoisinent respectivement 73% et 23%. Cela veut dire que les deux premiers vecteurs propres (les deux premières colonnes de `eigen_vecs`) résument respectivement 73% et 23% de l'information (l'inertie).

Dans cet exemple, les valeurs propres sont triées dans un ordre décroissant. Ce n'est qu'un cas particulier. Dans le cas général, nous devons procéder au tri (dans un ordre décroissant). Les vecteurs propres (les colonnes de `eigen_vecs`) doivent également être triés dans le même ordre :

```
>>> index = np.argsort(eigen_vals)[::-1]       # Indices des éléments triés dans un
>>> index                                      # ordre décroissant
array([0, 1, 2, 3], dtype=int64)

>>> eigen_vals = eigen_vals[index]             # Tri des valeurs propres
>>> eigen_vecs = eigen_vecs[:,index]           # Tri des vecteurs propres
```

Dans ce cas bien précis, le tri ne va pas changer l'ordre, puisque les valeurs propres sont déjà triées.

La matrice `eigen_vecs` est de taille 4x4. Chaque colonne représente un vecteur propre. Un vecteur propre est une combinaison linéaire des quatre variables d'origine. Les valeurs de la colonne correspondante sont donc les coefficients de cette combinaison linéaire.

```
>>> eigen_vecs
array([[ 0.52106591, -0.37741762, -0.71956635,  0.26128628],
       [-0.26934744, -0.92329566,  0.24438178, -0.12350962],
       [ 0.5804131 , -0.02449161,  0.14212637, -0.80144925],
       [ 0.56485654, -0.06694199,  0.63427274,  0.52359713]])
```

Plus explicitement, les 4 composantes principales (qu'on va nommer `CP1`, `CP2`, `CP3` et `CP4`) peuvent être exprimées de la façon suivante :

```
CP1 =  0.52*sepal.length - 0.27*sepal.width + 0.58*petal.length + 0.56*petal.width
CP2 = -0.38*sepal.length - 0.92*sepal.width - 0.02*petal.length - 0.07*petal.width
CP3 = -0.72*sepal.length + 0.24*sepal.width + 0.14*petal.length + 0.63*petal.width
CP4 =  0.26*sepal.length - 0.12*sepal.width - 0.80*petal.length + 0.52*petal.width
```

Il suffit de substituer les valeurs des 4 variables pour un individu (une ligne de notre matrice de données initiale centrée réduite) pour calculer les projections sur les composantes

principales. Par exemple, pour le premier individu, on obtient :

```
CP1 =  0.52*(-0.90) - 0.27*(1.02) + 0.58*(-1.34) + 0.56*(-1.32) = -2.26
CP2 = -0.38*(-0.90) - 0.92*(1.02) - 0.02*(-1.34) - 0.07*(-1.32) = -0.48
CP3 = -0.72*(-0.90) + 0.24*(1.02) + 0.14*(-1.34) + 0.63*(-1.32) = -0.13
CP4 =  0.26*(-0.90) - 0.12*(1.02) - 0.80*(-1.34) + 0.52*(-1.32) =  0.02
```

Ce calcul est réalisé en une seule instruction pour tous les individus avec le produit scalaire suivant :

```
>>> data_pca = np.dot(data_cr, eigen_vecs)
```

Ou bien :

```
>>> data_pca = data_cr.dot(eigen_vecs)        # Même résultat
```

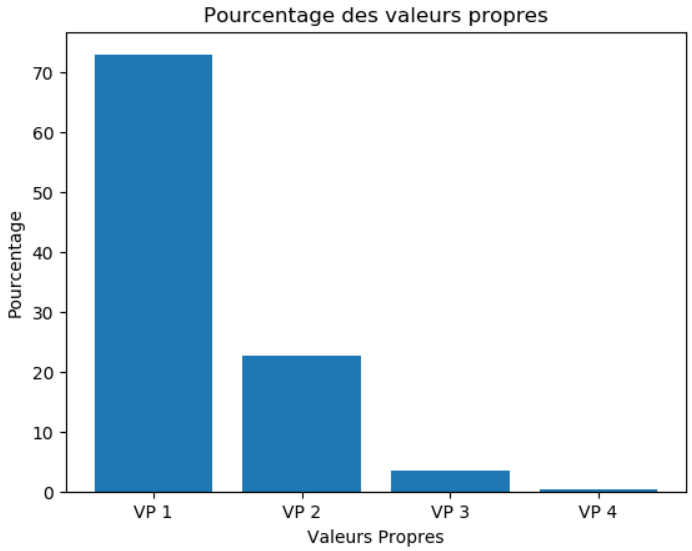
`data_pca` a la même taille que `data_cr` (150x4), où les lignes représentent les individus, et les colonnes représentent les 4 composantes principales. Nous avons donc obtenu une nouvelle matrice de données qui préserve toute l'information contenue dans la matrice de données d'origine, mais avec de nouvelles variables (les composantes principales) qui ont la propriété d'être orthogonales (décorrélées) et où la majeure partie de l'information (l'inertie) est représentée par les deux premières variables (96%). On peut donc se contenter des deux premières composantes, et considérer les deux dernières comme un résidu.

Visualisations

1. Visualisation des valeurs propres

Ce graphique visualise le pourcentage des 4 valeurs propres, qu'on va nommer `VP1`, `VP2`, `VP3` et `VP4`. Le pourcentage de `VP3` et `VP4` est négligeable par rapport à celui de `VP1` et `VP2`. On peut donc ne retenir que les 2 premières composantes principales.

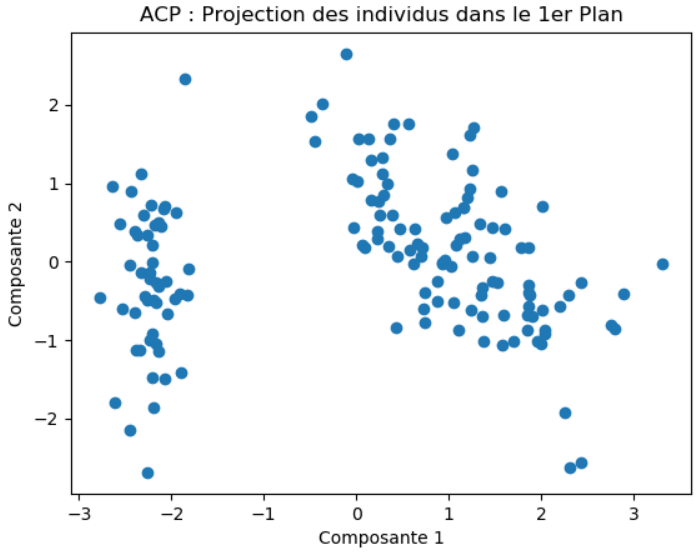
```
>>> plt.bar(['VP 1','VP 2','VP 3','VP 4'], eigen_vals/eigen_vals.sum()*100)
>>> plt.xlabel('Valeurs Propres')
>>> plt.ylabel('Pourcentage')
>>> plt.title('Pourcentage des valeurs propres')
>>> plt.show()
```



2. Visualisation des individus

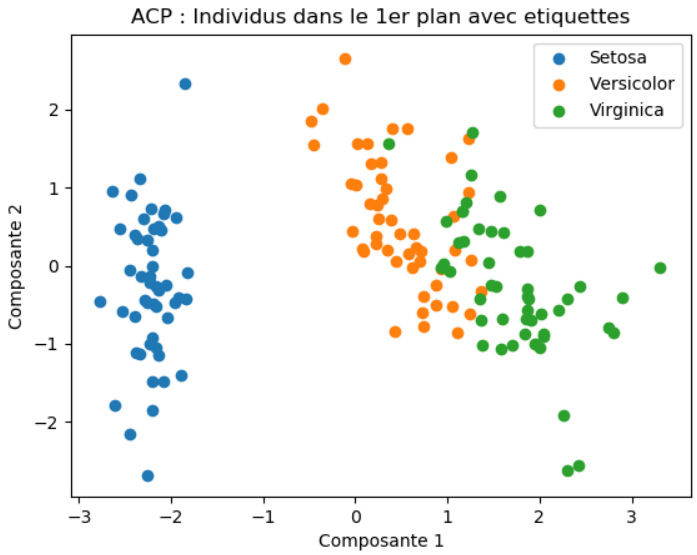
Ce graphique visualise l'ensemble des 150 individus dans le 1er plan principal (c-à-d les 2 premières composantes principales) :

```
>>> plt.scatter(data_pca[:,0], data_pca[:,1])
>>> plt.xlabel('Composante 1')
>>> plt.ylabel('Composante 2')
>>> plt.title('ACP : Projection des individus dans le 1er Plan')
>>> plt.show()
```



Optionnellement, on peut visualiser sur le même graphique l'espèce de chaque individu (en utilisant la variable catégorique `variety`, que nous avons sauvegardé dans le vecteur `labels`).

```
>>> for lb in np.unique(labels):
...  plt.scatter(data_pca[labels==lb,0], data_pca[labels==lb,1])
...

>>> plt.title('ACP : Individus dans le 1er plan avec etiquettes')

>>> plt.xlabel('Composante 1')

>>> plt.ylabel('Composante 2')

>>> plt.legend(np.unique(labels))

>>> plt.show()
```



3. Visualisations des variables

Nous avons vu que les composantes principales sont exprimées en fonction des variables d'origine, avec une combinaison linéaire. Les coefficients de cette combinaison sont les éléments de la matrice des vecteurs propres `eigen_vecs`.

Inversement, il est possible d'exprimer les variables d'origine en fonction des composantes principales, en multipliant les coefficients de chaque vecteur propre par la racine carrée de la valeur propre correspondante :

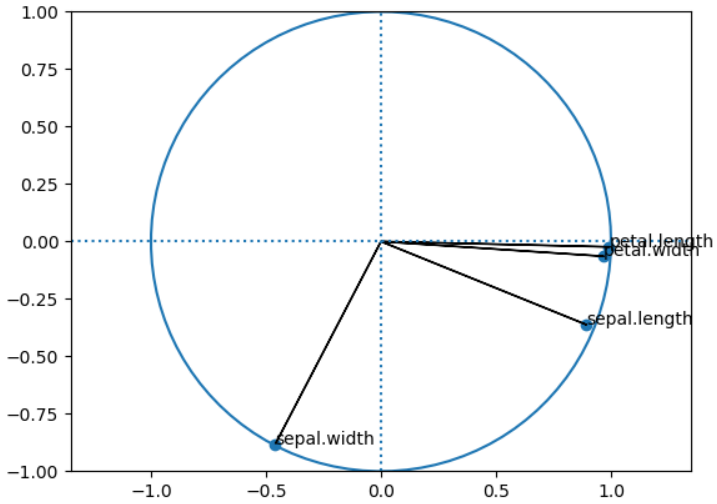
```
>>> variables_pca = np.sqrt(eigen_vals) * eigen_vecs

>>> variables_pca
array([[ 0.89016876, -0.36082989, -0.27565767,  0.03760602],
       [-0.46014271, -0.88271627,  0.09361987, -0.01777631],
       [ 0.99155518, -0.02341519,  0.05444699, -0.11534978],
       [ 0.96497896, -0.06399985,  0.24298265,  0.0753595 ]])
```

Chaque ligne de `variables_pca` représente une variable d'origine. Les colonnes représentent les coefficients de la combinaison linéaire pour les 4 composantes principale.

On peut représenter graphiquement les variables d'origine dans le premier plan principal (en ne prenant en considération que les deux premières colonnes) :

```
>>> plt.scatter(variables_pca[:,0], variables_pca [:,1])
```

Il est de coutumes de visualiser graphiquement cette projection à l'intérieur du cercle unité, comme l'illustre la figure suivante :



Ci-après le code permettant d'avoir ce graphique :

```
>>> an = np.linspace(0, 2*np.pi, 100)          # Générer et dessiner les points du
>>> plt.plot(np.cos(an), np.sin(an))           # cercle unité

>>> plt.axis('equal')                          # Pour avoir la même échelle pour les
                                               # deux axes (abscisses et ordonnées)
>>> plt.axis([-1, 1, -1, 1])                   # Limiter les intervalles entre -1 et 1

                                               # Dessiner les deux droites pointillées
>>> plt.axvline(0, ls=':'), plt.axhline(0, ls=':')

                                               # Projection des variables sur le 1er
                                               # plan principal
>>> plt.scatter(variables_pca[:,0], variables _pca[:,1])

>>> for i in range(var_names.size-1):
                                               # Ajout de textes : noms des variables
...   plt.annotate(var_names, (variables_pca[i,0], variables_pca[i,1]))
                                               # Dessiner les vecteurs allant du
                                               # centre aux variables
...   plt.arrow(0,0, var_projections[i,0], var_projections[i,1])

...
>>> plt.show()                                 # Affichage
```

Dans ce graphique, on remarque que les 4 points représentant les variables d'origine sont proche de la frontière du cercle unité. Cela veut dire que les 4 variables contribuent dans la construction des 2 composantes principales, et qu'il n'y a pas de variable qui peut être négligée. En même temps, les 4 variables présentent une corrélation importante entre elles. L'ACP a permis donc de les résumer dans deux variables : les 2 premières composantes principales.

R. HACHEMI