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° 5

Méthodes Inductives

Analyse Linéaire Discriminante

Nous avons vu que l'**analyse en composantes principales** permet de transformer un ensemble de variables quantitatives corrélées, en de nouvelles variables décorrélées, appelées **composantes principales**. Cette transformation est réalisée de sorte que la variance soit maximale sur la première composante, ensuite sur la deuxième composante et ainsi de suite.

Dans certains cas, nous disposons, en plus des variables quantitatives, d'une variable qualitative dont les modalités permettent de « catégoriser » les individus en des classes différentes. L'ACP, de par sa définition, ne tient pas compte de ces différentes classes et traite l'ensemble des individus de la même façon : le seul critère qu'elle utilise pour construire les nouvelles composantes est la variance de l'ensemble des individus.

L'**analyse linéaire discriminante** (**ALD**), quant à elle, construit les composantes principales sur la base d'un autre critère : discriminer au mieux entre les individus appartenant à différentes classes (modalités différentes de la variable qualitative). Discriminer au mieux entre les différentes classes revient à minimiser les variances intra-classes (les variances internes de chaque classe) et maximiser la variance inter-classes (variance entre les centres des classes).

Le schéma de la figure 1 récapitule les différentes étapes de calcul permettant de construire les composantes principales de l'ALD et la projection des individus. Il est ainsi démontré que les composantes principales sont les vecteurs propres de la matrice W. Cette matrice est celle qui permet la maximisation (argument du maximum) de l'expression $\frac{W^T S_B W}{W^T S_W W}$. Le résultat est donné par : $W = S_W^{-1} S_B$.

Nous allons suivre ce schéma et l'appliquer sur le jeu de données « **iris** » (utilisé dans le TP 4, méthode ACP). Le fichier de données `iris.csv` contient 4 variables quantitatives et une variable qualitative, pour un échantillon de 150 individus :

`sepal.length`	longueur des sépales	Variable quantitative
`sepal.width`	largeur des sépales	//
`petal.length`	longueur des pétales	//
`petal.width`	largeur des pétales	//
`variety`	espèce	Variable qualitative

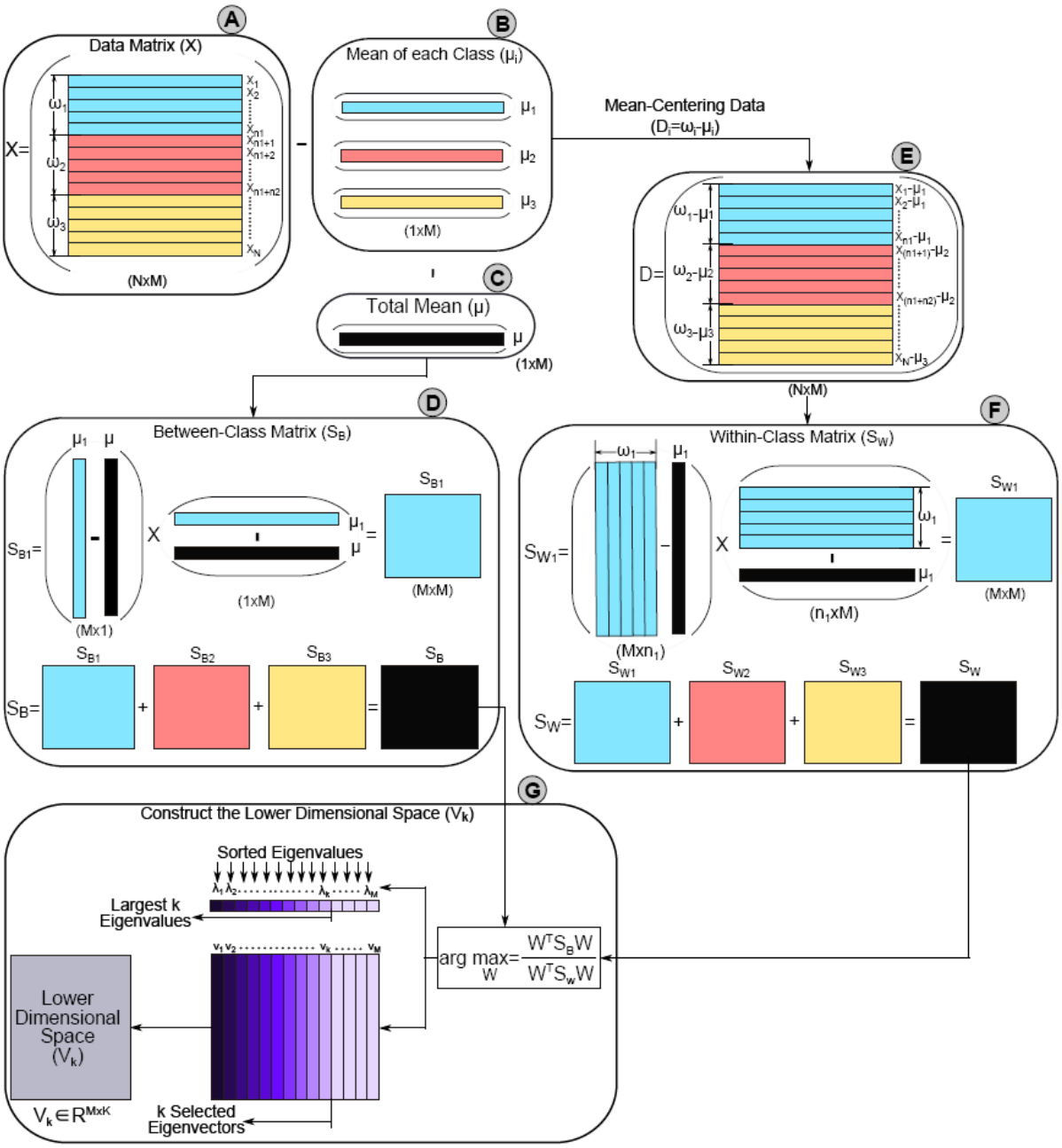


Figure 1. ALD : Etapes de calcul

La variable qualitative `variety` permet de catégoriser les individus en 3 classes (les trois modalités) : *Setosa, Versicolor* et *Virginica*. Les composantes principales de l'ALD sont les composantes permettant de « discriminer » au mieux entre ces trois classes.

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 des données

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

>>> X.head()
   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

>>> n = X.shape[0]                              # Nombre d'individus
>>> n
150
>>> m = X.shape[1]-1                            # Nombre de variables quantitatives
>>> m
4
```

Groupement des données

Avant de commencer les calculs, nous devons grouper la table `X` suivant les modalités de la variable qualitative `variety`. La méthode `groupeby` de `DataFrame` de `Pandas` permet de faire ce groupement :

```
>>> X_grouped = X.groupby(['variety'])
```

Calcul des moyennes globales et des moyennes des classes

Nous calculons ensuite la moyenne globale de chaque variable quantitative, ainsi que les moyennes correspondantes à chaque classe :

```
>>> mu = X.mean()                               # La Moyenne globale se calcule à
>>> mu                                          # partir de la table initiale "X"
sepal.length    5.843333
sepal.width     3.057333
petal.length    3.758000
petal.width     1.199333
dtype: float64


>>> mu_i = X_grouped.mean()                     # Les Moyennes des classes se calculent
                                                # à partir la table groupée
>>> mu_i                                        # "X_grouped"
            sepal.length  sepal.width  petal.length  petal.width
variety
Setosa             5.006        3.428         1.462        0.246
Versicolor         5.936        2.770         4.260        1.326
Virginica          6.588        2.974         5.552        2.026
```

Calcul de la matrice D

La matrice `D` s'obtient en soustrayant de chaque valeur la moyenne de la classe correspondante. La méthode `transform` permet de faire ce calcul :

```
>>> D = X_grouped.transform(lambda x: x-x.mean())
```

```
>>> D.head()
   sepal.length  sepal.width  petal.length  petal.width
0         0.094        0.072        -0.062       -0.046
1        -0.106       -0.428        -0.062       -0.046
2        -0.306       -0.228        -0.162       -0.046
3        -0.406       -0.328         0.038       -0.046
4        -0.006        0.172        -0.062       -0.046
```

Calcul des matrices S_B et S_W

Les matrices `SB` et `SW` se calculent sur la base des matrices et des vecteurs que nous venons de calculer, à savoir : `mu`, `mu_i` et `D`.

```
>>> SB = np.zeros((m,m))                          # Initialisation

>>> for v in np.unique(X.variety):                # Pour chaque modalité v
...   mu_i_centered = np.asarray(mu_i.loc[v] - mu).reshape(1,m)
...   SB += mu_i_centered.T.dot(mu_i_centered)
...
>>> SB                                            # Affichage de SB
array([[ 1.26424267, -0.39905333,  3.304968  ,  1.42558667],
       [-0.39905333,  0.22689867, -1.144792  , -0.45865333],
       [ 3.304968  , -1.144792  ,  8.742056  ,  3.73548   ],
       [ 1.42558667, -0.45865333,  3.73548   ,  1.60826667]])



>>> SW = np.zeros((m,m))                          # Initialisation

>>> for v in np.unique(X.variety):                # Pour chaque modalité v
...   D_i = np.asarray(D[X.variety==v])
...   SW += D_i.T.dot(D_i)
...
>>> SW                                            # Affichage de SW
array([[38.9562, 13.63  , 24.6246,  5.645 ],
       [13.63  , 16.962 ,  8.1208,  4.8084],
       [24.6246,  8.1208, 27.2226,  6.2718],
       [ 5.645 ,  4.8084,  6.2718,  6.1566]])
```

Calcul de la matrice W

$$W = S_W^{-1} S_B$$

```
>>> W = np.linalg.inv(SW).dot(SB)
>>> W
array([[-0.06116739,  0.02162765, -0.16223845, -0.069173  ],
       [-0.11123279,  0.04356437, -0.29929224, -0.12615479],
       [ 0.16154878, -0.05885437,  0.43023182,  0.18284129],
       [ 0.20994164, -0.06839709,  0.5509705 ,  0.2369176 ]])
```

Calcul des valeurs propres et des vecteurs propres de la matrice W

```
>>> eigen_vals, eigen_vecs = np.linalg.eig(W)
>>> eigen_vals
array([ 6.43838584e-01,  5.70782085e-03,  1.22941194e-16, -2.54049408e-17])
>>> eigen_vecs
array([[-0.20874182, -0.00653196,  0.12631523, -0.86698816],
       [-0.38620369, -0.58661055,  0.29160733,  0.21730559],
       [ 0.55401172,  0.25256154,  0.36445597,  0.18087961],
       [ 0.7073504 , -0.76945309, -0.87531792,  0.4103564 ]])
```

Projection des individus

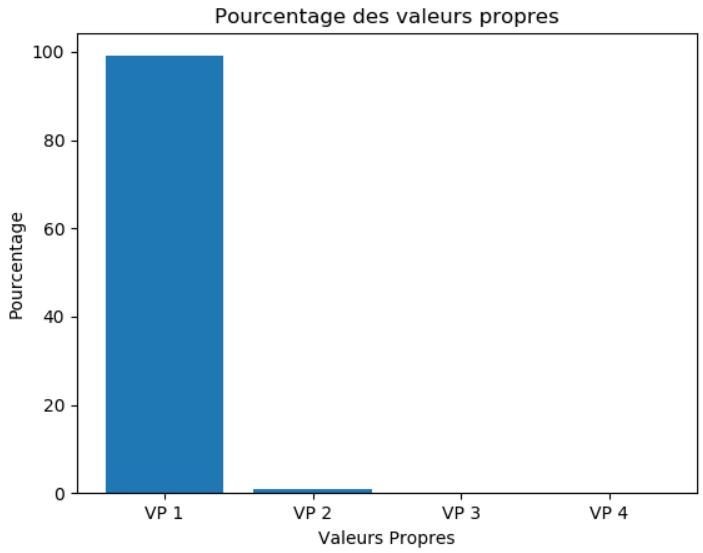
```
                                                    # On ne considère que les variables
                                                    # quantitatives : la dernière variable
                                                    # (qui est qualitative) est retirée de
                                                    # la matrice X avant la projection
>>> X_lda = np.asarray(X)[:,:-1].dot(eigen_vecs)
                                                    # X_lda est la matrice de projection
>>> X_lda.shape
(150, 4)                                            # 150 individus sur 4 composantes
```

Visualisations

1. Visualisation des valeurs propres

Ce graphique visualise le pourcentage des 4 valeurs propres, qu'on va nommer `VP1`, `VP2`, `VP3` et `VP4`. Les pourcentages de `VP2`, de `VP3` et de `VP4` sont négligeables par rapport à celui de `VP1`. La première composante principale permettra une meilleure discrimination.

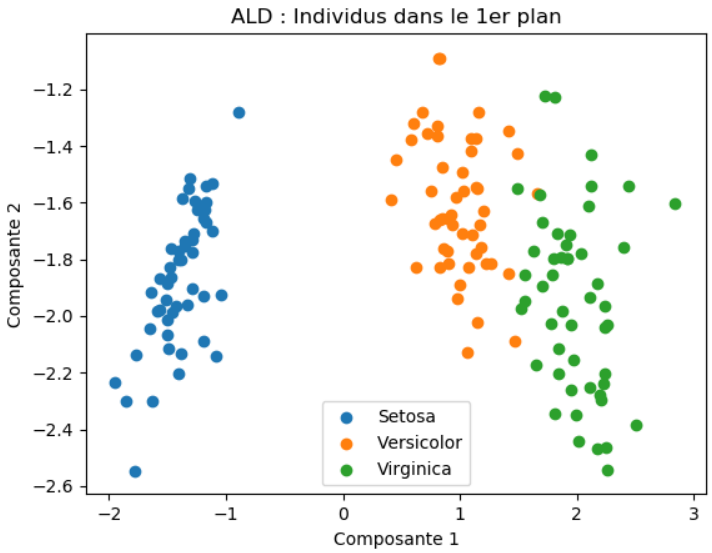
```
>>> plt.bar(['VP 1','VP 2','VP 3','VP 4'], eigen_vals/eigen_vals.sum()*100)
>>> plt.xlabel('Valeurs Propres')
>>> plt.ylabel('Pourcentage')
>>> plt.title('Pourcentage des valeurs propres')
>>> plt.show()
```



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) :

```
>>> for v in np.unique(X.variety):
...  plt.scatter(X_lda[X.variety==v,0], X_lda[X.variety==v,1])
...

>>> plt.title('ALD : Individus dans le 1er plan avec étiquettes')

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

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

>>> plt.legend(np.unique(X.variety))

>>> plt.show()
```



Dans ce graphique, nous remarquons qu'il y a une meilleure discrimination entre les 3 classes, comparée à la projection faite en utilisant l'ACP (ce qui est normal car l'ACP ne cherche pas forcément à discriminer entre les classes mais à maximiser la variance globale).

Nous remarquons également que c'est la composante 1 qui permet cette discrimination, ce qui est prévisible vu le pourcentage très élevé de la valeur propre correspondante.

Remarque : prévision de la classe d'un nouvel individu

Considérons la situation suivante :

- Nous disposons d'un ou de plusieurs individus qui ne font pas partie des 150 individus qui nous ont servi à trouver la représentation ci-dessus.
- Les valeurs des variables quantitatives sont disponibles pour ces nouveaux individus.
- La variable qualitative (la classe) est inconnue pour ces nouveaux individus.

Après avoir trouvé la meilleure représentation permettant une discrimination entre les classes, il est possible de déduire la classe de ces nouveaux individus. En effet, en les projetant dans le nouvel espace (tout comme la projection des individus initiaux), le principe consiste à calculer les distances de l'individu à classer aux centres de gravité des différentes classes et de l'affecter à la classe la plus proche. Il est à noter qu'il s'agit d'une mesure de distance appropriée (distance de Mahalanobis Fisher).

R. HACHEMI