TP 3 : Apprentissage non supervisé sous `Python`
K-MEANS et CAH

IMT Atlantique – FIL A3

Apprentissage Automatique

Objectifs :
Programmer quelques méthodes de clustering, évaluer leurs performances, en utilisant la bibliothèque `Scikit-Learn` de `Python`.

Evaluation
Vous devez rendre un code python qui tourne sur netbook Jupyter + un mini rapport (format pdf) présentant le travail réalisé, les résultats obtenus et une analyse (intéressante) de ces résultats.

1 Données de travail

1.1 Importation, préparation des données et graphiques

On dispose de 2 jeux de données[1] représentant un ensemble de fromages (29 observations) décrits par leurs propriétés nutritives (ex. protéines, lipides, calcium, magnésium, etc. ; 9 variables) :

- le fichier `"fromage_all.txt"` contient une description complète des 9 propriétés nutritives des 29 fromages ;
- le fichier `"fromage.txt"` est une version simplifiée qui regroupe les 4 propriétés nutritives les plus importantes.

L'objectif est d'identifier des groupes de fromages partageant des caractéristiques similaires. Nous utiliserons pour cela deux approches basées sur deux librairies spécialisés pour `Python` :

1. la méthode des centres mobiles (K-MEANS – Librairie `Scikit-Learn`) ;
2. la classification ascendante hiérarchique (CAH – Librairie `SciPy`)

Les instructions Python suivantes permettent de charger le jeu de données `fromages.txt` ainsi que d'autres librairies utiles pour la suite du TP :

```
#importation des librairies
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

#chargement des données
import pandas as pd
data = pd.read_table('fromage.txt',decimal = ".")
```

1. Les données proviennent de la page de cours de Marie Chavent : `http://www.math.u-bordeaux.fr/~mchave100p/teaching/`.

1.2 Travail à faire

Écrivez la séquence d'instructions qui permettent de :

1. Charger le jeu de données `fromages.txt` et visualiser les infos de chaque variable séparément avec la méthode `data.info()`.
2. Afficher la moyenne et l'écarts-type par attribut.
3. Afficher le nuage de points entre les deux derniers attributs ;
   — Regardez la description complète de la commande `scatter` sur `http://matplotlib.org/api/pyplot_summary.html`.
4. Étudier les relations existantes entre les différentes attributs.
   — Utilisez la méthode `paiplot` de librairie `Seaborn` qui affiche par paire toutes les variables numériques dans une grille à plusieurs axes.
5. Que pouvez-vous en déduire sur les distribution des différents attributs ainsi que leurs corrélations ?

1.3 Mise à l'échelle des données

Pour un grand nombre de modèles de *machine learning*, il est préférable de travailler sur des données mises à l'échelle (par exemple centrées-réduites, ou à valeurs entre 0 et 1). Pour cela, on va utiliser le module `preprocessing` de `sklearn` qui contient tout un tas de méthodes pour préparer vos données avant de les utiliser. Pour tous ces pré-traitements, le fonctionnement est le même :

(a) on construit un objet de la classe visée en lui précisant certains hyper-paramètres ;
(b) on appelle sa méthode `fit` pour ajuster les paramètres (par exemple moyenne et écart-type dans le cas où l'on souhaite centrer-réduire nos données) ;
(c) on appelle sa méthode `transform` pour appliquer la transformation à nos données.

> **Note**
> Attention, les objets sklearn attendent des données quantitatives pour être estimés. Vous devrez donc ne leur passer qu'un sous-ensemble de votre dataframe composé de ses colonnes quantitatives.

Travail à faire.

1. Utilisez la classe `StandardScaler` pour centrer-réduire les données de votre dataframe.

```
#centrer et réduire les données
from sklearn import preprocessing
std_scale = preprocessing.StandardScaler().fit(data)
data_scaled = std_scale.transform(data)
```

2 La méthode k-means

2.1 Configuration de la méthode

Nous importerons le paquet `sklearn.cluster` pour utiliser la méthode K-MEANS de la classe `sklearn.cluster.KMeans`. La description de l'implémentation de la méthode des K-moyennes (K-MEANS) se trouve dans : `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`.

L'exécution de l'algorithme K-MEANS se fait par la commande suivante :
`KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', n_jobs=1)` :

- `n_clusters` : le nombre de classes (par défaut `n_clusters = 8`).
- `init` : {`'k-means++'`, `'random'` ou un `'ndarray'` } : est une méthode d'initialisation, par défaut `'k-means++'` :

— `'k-means++'` : sélectionne intelligemment les centres initiaux afin d'accélérer la convergence.

— `'random'` : choisit k observations (rangées) au hasard parmi les données pour les centres initiaux.

— `'ndarray'` : passe en paramètre les centres initiaux sous la forme (n_clusters, n_features).

- `n_jobs=1` permet d'exécuter les `n_init` itérations en parallèle.

Les résultats de l'algorithme K-MEANS sont accessibles via différents attributs :
`cluster_centers_` : contient les attributs en sortie : les centres, `labels_` : les numéros de cluster de chaque observation, `inertie` : la somme des distances au carré des observations vers leur centre de cluster le plus proche.

Les commandes suivantes :

- `km = KMeans(n_clusters=`k`)` permet de créer un modèle pour un ensemble de k centres,
- L'instruction `km.fit(X)` utilise les données pour définir le modèle de clustering,
- `predict(X)` prédit le cluster le plus proche auquel appartient chaque échantillon.

> **Note**
> Les données représentées par le paramètre X correspondent aux données centrées réduites.

2.2 Travail à faire

Dans l'algorithme K-MEANS, le nombre k de clusters est fixé au départ. A partir d'une partition initiale, on cherche à améliorer itérativement la partition en minimisant la somme des carrés des distances euclidiennes de chaque point à la moyenne des points de son cluster, notée `WCSS`. Écrivez les séquences d'instructions qui permettent de :

1. Effectuer un clustering du jeu de données `formage.txt` en utilisant l'algorithme K-Means avec $k = 2$ et en conservant les autres paramètres par défaut.
2. Visualisez les résultats de cette classification.
3. Afficher le groupe de chaque fromage selon les valeurs des deux attributs `"lipides"` et `"proteines"` en fonction des labels retournés par K-MEANS. Vous devez également afficher les centroids correspondants sur la graphique. Utilisez les commandes `scatter` et `plot`.

On souhaite à présent déterminer la valeur optimale du paramètre k à l'aide de la méthode `Elbow`. Le principe de cette méthode est d'analyser le changement substantiel de la valeur d'inertie [2] en fonction de l'augmentation du nombre de clusters.

4. Variez le nombre de groupes (`n_clusters`) entre 2 et 10, tracez le graphique d'évolution de la valeur finale atteinte par la valeur d'inertie, pour chacune des valeurs de `n_clusters`.
5. Quel est le meilleur nombre de clusters k pour le jeu de données ?
6. Effectuer un nouveau clustering avec la valeur optimale k trouvée précédemment et afficher à nouveau le groupe de chaque fromage selon les valeurs des deux attributs `"lipides"` et `"proteines"`. Que constatez-vous ? Expliquez.

3 La Classification Ascendante Hiérarchique

3.1 Configuration de la méthode

La classification ascendante hiérarchique (ou CAH) procède par fusions successives d'ensembles de points (clusters), en considérant initialement tous les points comme des clusters singletons, on

2. indicateur de qualité de la solution qui mesure l'aptitude des individus à être plus proches de ses congénères du même groupe que des individus des autres groupe : paramètre `WCSS`.

fusionne à chaque étape les 2 clusters les plus proches au sens d'une distance, jusqu'à obtenir un seul cluster contenant tous les points.

La description de l'implémentation de la méthode des CAH se trouve dans :

`http://scikitlearn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering`

L'exécution de l'algorithme CAH se fait par l'utilisation `sklearn.cluster.AgglomerativeClustering` avec la commande suivante :

```
AgglomerativeClustering(n_clusters=2, affinity='euclidean', connectivity=None,
compute_full_tree='auto', linkage='ward')
```

- `n_clusters` : le nombre des cluster à trouver (2 par défaut).
- `affinity` : ("euclidean" par défaut) "euclidean", "l1", "l2", "manhattan", "cosine", 'precomputed', est la métrique utilisé pour calculer la stratégie (linkage).
- `linkage` : {"ward", "complete", "average"}, ("ward" par défaut). Définit une distance entre groupes d'individus (appelé stratégie d'agrégation). Stratégie du saut minimum ou single linkage (la distance entre groupes est la plus petite distance entre éléments des deux groupes). Stratégie du saut maximum ou du diamètre ou complete linkage (la distance entre groupes est la plus grande distance entre éléments des deux groupes). Méthode du saut Ward (en espace euclidien, on agrège de manière à avoir un gain minimum d'inertie intra-classe à chaque itération).

3.2 La classe dendogramme

CAH, à la différence de K-MEANS, fournit un outil d'aide à la détection du nombre de classes. Une méthode connue pour trouver le nombre de clusters optimal est d'utiliser un `dendrogramme`. Il permet de visualiser les regroupements successifs jusqu'à obtenir un unique cluster. Il est souvent pertinent de choisir le partitionnement correspondant **au plus grand saut entre deux clusters consécutifs**.

Le nombre de clusters correspond alors au nombre de lignes verticales traversée par la coupe horizontale du dendrogramme. La description de l'implémentation de la classe se trouve dans : `https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html`

La génération d'un dendogramme se fait en deux étapes, par l'utilisation de `scipy.cluster.hierarchy` :

1. génération de la matrice des liens avec la méthode
   ```
   Z = linkage(data,method='ward',metric='euclidean')
   ```
   - `data` : jeu de données initial ;
   - `method` :{"ward", "complete", "average"}, ("ward" par défaut). Identique au paramètre `linkage`.
   - `metric` : ("euclidean" par défaut), est la métrique utilisée pour calculer la distance.
2. affichage du dendogramme avec la méthode `dendrogram(Z)` ; pour les autres paramètres, les valeurs par défaut sont suffisantes.

La méthode `scipy.cluster.hierarchy.fcluster(Z, t, criterion='inconsistent', depth=2, R=None, monocrit=None)` permet de matérialiser les différents groupes du dendogramme sous la forme d'un clustering à plat (i.e. chaque individu est associé à une classe) :

- `Z` : le dendogramme associé aux données initiales ;
- `t` (valeur scalaire) : il s'agit du seuil (la heuteur) à utiliser pour former le clustering à plat.

- `criterion` : {"inconsistent", "distance", "maxclust", "monocrit", "maxclust_monocrit"}, ils correspondent aux critères utilisés pour former le clustering à plat.
  le critère "distanc" permet de former des clusters de sorte que les observations originales dans chaque groupe sont distantes d'au plus t.

3.3 Travail à faire

1. Afficher le dendogramme associé au jeu de données `fromage_all.txt`. Pensez à mettre à l'échelle les données du jeu.
2. Faites le lien entre le dendrogramme et les valeurs de seuil permettant de trouver des clusters raisonnables pour le jeu de données `fromage_all.txt`.
3. Effectuer un clustering du jeu de données en utilisant l'algorithme CAH avec le critère d'inertie de `Ward`, puis afficher les groupes des différents fromages. Que pouvez-vous en déduire sur cette classification ?
4. Modifier l'attribut linkage : "complete", "average". Affichez les dendogrammes associés. Quel est le meilleur nombre de clusters ? Que pouvez-vous en déduire sur la nouvelle classification ?

4 Références

1. K-Means Clustering in Python : A Practical Guide.
   `https://realpython.com/k-means-clustering-python/`.
2. K-Means Clustering of Iris Dataset.
   `https://www.kaggle.com/khotijahs1/k-means-clustering-of-iris-dataset`
3. K-Means Clustering in Python with scikit-learn.
   `https://www.datacamp.com/community/tutorials/k-means-clustering-python`
4. scikit-learn : machine learning in Python.
   `http://scipy-lectures.org/packages/scikit-learn/`
5. A Beginner's Guide to Hierarchical Clustering and how to Perform it in Python.
   `https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering`
6. SciPy Hierarchical Clustering and Dendrogram Tutorial. `https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/`