Fouille de Données et Apprentissage Artificiel
TP 1 - Clustering

issam Falih
issam.falih@uca.fr

Instructions : Préparez un rapport incluant le code source et vos résultats, et déposez-le sur Moodle. Il est recommandé d'utliliser un jupyter notebook ou encore l'IDE spider.

1 Normalisation de données

La normalisation de données est une étape importante dans le processus de traitement de données. Par exemple, de nombreux éléments utilisés dans la fonction objective d'un algorithme d'apprentissage (tels que le noyau RBF de Support Vector Machines ou la L1 et L2 régularisé des modèles linéaires) supposent que toutes les variables sont centrées autour de zéro et ont la variance dans le même ordre. Si une caractéristique a une variance qui est des ordres de grandeur plus grand que les autres, il pourrait dominer la fonction objectif et de faire l'estimateur incapable d'apprendre correctement comme prévu. En pratique, nous ignorons souvent la forme de la distribution de données et on simplement transforme les données en les centrent en retirant la valeur moyenne de chaque variable, puis en divisant les variables par leur écart-type.

Importez les librairies numpy (calcul scientifique) et preprocessing (prétraitement de données)

1. Créez la matrice X suivante :

```
      1, -1, 2,
X =   2, 0, 0,
      0, 1, -1
```

2. Visualisez X et calculez la moyenne et la variance de X.
3. Utilisez la fonction `scale` pour normaliser la matrice X. Que constatez vous ?
4. Calculer la moyenne et la variance de la matrice X normalisé. Expliquez le résultat obtenu.

2 Normalisation MinMax

Un autre type de normalisation est de normaliser les caractéristiques (variables) de données entre un minimum et une valeur maximale donnée, souvent entre zéro et un. Ceci peut être réalisé en utilisant la fonction `MinMaxScaler`.

1. Créez la matrice de données X2 suivante :

```
      1, -1, 2,
X2 =  2, 0, 0,
      0, 1, -1
```

1. Visualisez la matrice et calculez la moyenne sur les variables.

1. Normalisez les données dans l'intervalle [0 1]. Visualisez les données normalisées et calculez la moyenne sur les variables. Que constatez-vous ?

3 Analyse des Iris de Fisher avec l'algorithme K-Moyennes

1. Ouvrez le fichier `iris.data` en utilisant la commande **read_table(**$\cdots$**)** avec les bons paramètres.

2. La dernière colonne de vos données représente le label correspondant à l'espèce d'Iris associée. Stockez ces labels dans un vecteur séparé et enlevez-le de votre jeu de données.

3. Utilisez la commande **sklearn.decomposition.PCA(** pour réaliser une analyse en composante principale de vos données. Puis, afin de récupérer vos données projetées dans 2 composantes.

Vous devriez maintenant avoir 2 jeux de données, le jeux de données original stocké dans une première variable ("`data`" par exemple), et le même jeu de données projeté sur 2 composantes stockée dans la variable `X`.

4. Utilisez l'algorithme K-Means sur votre jeu de données `X` afin d'obtenir 3 clusters et visualisez les résultats. Affichez également les centroides sur la figure.

5. Répétez la question 4) plusieurs fois. Que constatez-vous ? Expliquez.

6. En utilisant la commande **plot(**$\cdots$**)** du package plot `matplotlib` comme dans la question 4, projetez les labels que vous aviez stockés dans un vecteur à part à la question 2). Comparez visuellement les résultats de vos clustering avec ces labels. Commentez.

7. Afficher le tableau de contingence comparant vos résultats aux labels théoriques.

8. Choisissez une de vos solutions de clustering et calculez l'indice de Silhouette associé à vos données. Commentez.

9. Refaites les questions 4) à 8) en utilisant les données originales ("`data`") plutôt que les données projetées. Les différences sont-elles importantes ? Vous expliquerez quels peuvent être les avantages et inconvénients d'utiliser les données originales ou bien les données projetées.

4 Clustering hiérarchique sur les données Iris

Dans cet exercice, on souhaite effectuer un clustering hiérarchique ascendant sur les données brutes Iris (sans labels).

1. La commande **AgglomerativeClustering** $(\cdots)$ permet d'effectuer un clustering hiérarchique ascendant en Python. Utilisez l'aide de Python afin de déterminer quels sont les paramètres de cette commande et comment l'utiliser avec vos données.

2. Effectuez un clustering hiérarchique ascendant avec un linkage complet et affichez le dendrogramme résultant.

3. Couper le dendrogramme de manière à avoir 3 branches, i.e une partition avec 3 clusters à partir de votre résultat de clustering hiérarchique ascendant.

4. Affichez le tableau de contingence comparant vos résultats avec les labels théoriques. Commentez vos résultats.

5. Refaites les questions 2) à 4) en utilisant un linkage moyen. Comparez les résultats et dites laquelle des deux méthodes vous semble meilleure.

6. Calculez les indices de Silhouette pour les 2 partitions (linkage complet et linkage moyen). Les résultats de la questions 5) sont-ils confirmés ? Commentez.

5 Nombre optimal de clusters sur les données atmosphère d'exoplanète

Dans cet exercice, on va évaluer le nombre de cluster optimal dans un jeu de données artificiel décrivant des compositions atmosphérique d'exoplanètes.

1. Ouvrez le fichier `planete.csv` et retirez la dernière colonne qui contient les labels.

2. Rappelez les propriétés de l'indice de Calinski-Harabasz et celles de l'indice de Davies-Boouldin.

3. Utilisez la méthode du coude `elbow` pour déterminer le nombre optimal de clusters pour l'algorithme des K moyennes à partir des indices de Davies-Bouldin et Calinski-Harabasz. Expliquez avec vos mots ce que fait cette fonction.

4. Commentez vos résultats. Vous pourrez utiliser des graphiques, des projections ACP et les labels pour appuyer vos explications.