ISEP
Machine Learning
October 14th 2024

Unsupervised learning : Clustering

1 Part I : Exercises

Exercise 1. Suppose that we have four observations, for which we compute a dissimilarity matrix, given by :

$$\begin{bmatrix} & 0.3 & 0.4 & 0.7 \\ 0.3 & & 0.5 & 0.8 \\ 0.4 & 0.5 & & 0.45 \\ 0.7 & 0.8 & 0.45 & \end{bmatrix}$$

For instance, the dissimilarity between the first and second observations is 0.3, and the dissimilarity between the second and fourth observations is 0.8.

(a) On the basis of this dissimilarity matrix, sketch the dendrogram that results from hierarchically clustering these four observations using complete linkage. Be sure to indicate on the plot the height at which each fusion occurs, as well as the observations corresponding to each leaf in the dendrogram.

(b) Repeat (a), using single linkage clustering.

(c) Cut the dendrogram obtained in (a) such that two clusters result. Which observations are in each cluster ?

(d) Cut the dendrogram obtained in (b) such that two clusters result. Which observations are in each cluster ? Compare the results with those obtained in the previous question.

Exercise 2. Suppose that for a particular data set, we perform hierarchical clustering using single linkage and using complete linkage. We obtain two dendrograms.

a) At a certain point on the single linkage dendrogram, the clusters {1, 2, 3} and {4, 5} fuse. On the complete linkage dendrogram, the clusters {1, 2, 3} and {4, 5} also fuse at a certain point. Which fusion will occur higher on the tree, or will they fuse at the same height, or is there not enough information to tell ?

b) At a certain point on the single linkage dendrogram, the clusters {5} and {6} fuse. On the complete linkage dendrogram, the clusters {5} and {6} also fuse at a certain point. Which fusion will occur higher on the tree, or will they fuse at the same height, or is there not enough information to tell ?

Exercise 3. In this problem, you will perform K-means clustering manually, with $K = 2$, on a small example with $n = 6$ observations and $p = 2$ features. The observations are as follows.

Obs	X_1	X_2
1	1	4
2	1	3
3	0	4
4	5	2
5	6	2
6	4	0

(a) Plot the observations.

(b) Randomly assign a cluster label to each observation. You can use the `choice()` command from the `random` module in `Python` or `choices()` for Python versions 3.6 and up. Report the cluster labels for each observation.

(c) Compute the centroid for each cluster.

(d) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation. For that, plot the observations and the centroids.

(e) Repeat (c) and (d) until the obtained clusters stop changing.

(f) Finally, In your plot, color the observations according to the obtained cluster labels.

2 Part II : Practical application

2.1 K-means clustering

In Python you can use the function `KMeans()` from the module `sklearn.cluster` to perform K-means clustering. To begin you will perform K-means with simulated data. Follow the steps :

(a) The simulated data will consist in 50 observations described by two normal-distributed variables. In order to define classes in the data the first 25 observations have a mean shift relative to the next 25 observations.

```
import numpy as np
X = np.random.randn(50,2)
X[0:25, 0] = X[0:25, 0] + 3
X[0:25, 1] = X[0:25, 1] - 4
```

You can plot the observations and notice that there are two well separated clusters :

```
import matplotlib.pyplot as plt
plt.plot(X[:,0], X[:,1],"o")
plt.xlabel("X1")
plt.ylabel("X2")
```

(b) Perform K-means clustering with $K = 2$. Use the following command :

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 2, random_state = 100).fit(X)
```

Then, the cluster assignments can be obtained by running `print(kmeans.labels_)`. The final clusters centroids can be known using the command `print(kmeans.cluster_centers_)`.

In addition, you can plot the observations using a different color per cluster by running the following code :

```
for i in range(0,X.shape[0]):
    if kmeans.labels_[i] == 0:
        plt.plot(X[i:,0], X[i:,1],"o",color= "red")
    else:
        plt.plot(X[i:,0], X[i:,1],"o",color= "blue")
centroid1=kmeans.cluster_centers_[0]
```

```
centroid2=kmeans.cluster_centers_[1]
plt.plot(centroid1[0],centroid1[1],"*",color= "black",markersize = 15)
plt.plot(centroid2[0],centroid2[1],"*",color= "black",markersize = 15)
plt.xlabel("X1")
plt.ylabel("X2")
```

(c) [graded question] Now, you are going to perform $K-$means with real data. The file *Live_20210128.csv* contains statistics about Facebook pages of 10 Thai fashion and cosmetics retail sellers. Each observation represents a post of different nature (video, photo, status or link). The features are variables describing each post such as the date, the number of reactions (*num_reactions*), number of comments (*num_comments*), number of likes (*num_likes*), etc. Import the data set and get familiar with the data. After dropping the last 4 columns (empty), answer the following questions : how many observations are there ? How many variables are there ? You will denote the resulting dataframe object `Live`.

(d) [graded question] Perform descriptive statistics with the dataset and comment the results. Do you think is it suitable to scale the data before performing clustering analysis ? Justify your answer.

(e) [graded question] In the following you are going to perform clustering analysis using only 5 features of the `Live` dataset, that is, *num_reactions*, *num_comments*, *num_shares*, *num_likes* and *num_loves* prepare this dataset and scale the data. You will denote the scaled dataset `Live_scaled`. You can use the following commands :

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

(f) [graded question] The purpose of this study is to get clusters of posts based on people's reactions on Facebook. Perform K-means clustering with $K = 4$ considering. Use the following code with the given values of the parameters *n_init* and *random_state*.

```
from sklearn.cluster import KMeans
Live_Kmeans4=KMeans(n_clusters = 4, n_init = 50, random_state =
    1000).fit(Live_scaled)
Live_Kmeans4_labels=Live_Kmeans4.labels_
Live_Kmeans4_labels
```

Although, we are performing unsupervised learning, we can use a categorical variable to interpret the obtained clusters. To this end, you can calculate a matching matrix, which is a two-dimension matrix ("real groups" versus "clusters"). The variable *status_type* contains the type of post. Use the following code to calculate the matching matrix and interpret the results by trying to match each type of post to a given cluster detected by the algorithm.

```
pd.crosstab(index = Live.status_type,
            columns = Live_Kmeans4_labels,
            rownames = ['Real groups'],
            colnames = ['K–means clusters'])
```

where `Live.status_type` are the labels defined by the variable *status_type* and `Live_Kmeans4_labels` are the cluster labels returned by the `KMeans()` function and `pd` is the `pandas` library.
Comment on the results.

(g) [graded question] Now you are going to use the elbow method to choose the optimal number of clusters. Run the following code to calculate the within cluster variation for number of clusters ranging from 1 to 15. Make a plot and choose the best number of clusters. Justify your answer.

```
distortions = []
K = range(1,15)
for k in K:
    kmeanModel = KMeans(n_clusters=k, n_init = 50, random_state =
        1000)
    kmeanModel.fit(Live_scaled)
    distortions.append(kmeanModel.inertia_)
```

(h) [graded question] Perform K-meand with the chosen value of K in the previous question, calculate the matching matrix and interpret the results.

2.2 Hierarchical clustering

In Python, the `linkage()` function from `SciPy` implements hierarchical clustering . For instance, in order to perform hierarchical clustering using complete linkage for the simulated data generated in the previous section, you can execute the following code :

```
from scipy.cluster.hierarchy import linkage
hc_complete = linkage(X, "complete")
```

Then, in order to plot the associated dendrogram using the previous results, you can use the code :

```
from scipy.cluster.hierarchy import dendrogram
import matplotlib.pyplot as plt
plt.figure(figsize=(25, 10))
dendrogram(hc_complete)
plt.show()
```

We remark the 2 clusters are clearly identified, which is normal because we performed hierarchical clustering with simulated data.
To determine the cluster labels for each observation associated with a given cut of the dendrogram, you can use the `cut_tree()` function from the `scipy.cluster.hierarchy` module . For example, to obtain two clusters you can execute the code :

```
from scipy.cluster.hierarchy import cut_tree
print(cut_tree(hc_complete, n_clusters = 2).T)
```

(a) [graded question] Perform hierarchical clustering using the `Live_scaled` data using complete linkage. To begin you will cut the dendrogram at 4 clusters. Similarly to the previous section, calculate the matching matrix and comment the results.

(b) [graded question] Now you will plot the dendrogram using the parameter `truncate_mode = 'lastp'` and `p=15`. According to the dendrogram at which level (number of clusters) is suitable to cut the dendrogram ? Why ?

(c) [graded question] Perform clustering evaluation by calculating the Silhouette coefficient and Davies-Bouldin index for K-means clustering and Hierarchical clustering for $K = 6$. Use the functions `silhouette_score` and `davies_bouldin_score` from the `metrics` library. Compare and conclude.

References

— James, Gareth ; Witten, Daniela ; Hastie, Trevor and Tibshirani, Robert. "An Introduction to Statistical Learning with Applications in R", 2nd edition, New York : "Springer texts in statistics", 2021. Site web : `https://hastie.su.domains/ISLR2/ISLRv2_website.pdf`.

— J Crouser : "SDS 293 - Machine Learning labs". `http://www.science.smith.edu/~jcrouser/SDS293/`. Visited on October 18th.

— Dehouche, Nassim. (2019). Facebook Live Sellers in Thailand. UCI Machine Learning Repository.