COMP809 – Data Mining and Machine Learning

Lab 9 – Agglomerative and Density based Clustering

This lab will cover the two approaches to clustering discussed in session 9.

Study the code provided below and run it in Python. The code uses the same half moons dataset that we met in the lecture session.

Vary the critical parameters for each of the parameters for the two clusterers and observe the effect. In particular, vary the n_clusters parameter for the agglomerative clusterer (keep the distance_threshold value at its default value of 0). For DBSCAN keep the min_samples value fixed at 4 (=2*D) and vary the eps value from its default value of 0.5.

Extend the code to use one of the datasets required for the assessment. Once it is working use PCA as a pre processing step prior applying the two clusterers. You will need to decide how many PCA dimensions (D) to extract. As a practical value, use a value of D=5.

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.datasets import make_moons
from sklearn.cluster import DBSCAN

X, y = make_blobs(n_samples=150,
                  n_features=2,
                  centers=3,
                  cluster_std=0.5,
                  shuffle=True,
                  random_state=0)

km = KMeans(n_clusters=3,
            init='random',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)

y_km = km.fit_predict(X)

# Organizing clusters as a hierarchical tree
# Clustering data using Agglomerative Clustering in bottom-up fashion
#For more details - https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

ac = AgglomerativeClustering(n_clusters=3,
                             affinity='euclidean',
                             linkage='complete')
labels = ac.fit_predict(X)
print('Cluster labels Agglokerative with 3 clusters: %s' % labels)

# increase  the level of abstraction by decreasing the number of clusters

ac = AgglomerativeClustering(n_clusters=2,
                             affinity='euclidean',
                             linkage='complete')
```

```
labels = ac.fit_predict(X)
print('Cluster labels Agglomrtative with 2 clusters: %s' % labels)

#Now we are going to create two interleaving half circles using sklearn make_moons function.
#Read more on- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

X, y = make_moons(n_samples=200, noise=0.05, random_state=0)
plt.scatter(X[:, 0], X[:, 1])
plt.tight_layout()
plt.show()

# Compare K-means and hierarchical clustering clustering:

#Configure the plot
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 3))

#Apply k-means with 2 clusters
km = KMeans(n_clusters=2, random_state=0)
y_km = km.fit_predict(X)

ax1.scatter(X[y_km == 0, 0], X[y_km == 0, 1],
            edgecolor='black',
            c='lightblue', marker='o', s=40, label='cluster 1')
ax1.scatter(X[y_km == 1, 0], X[y_km == 1, 1],
            edgecolor='black',
            c='red', marker='s', s=40, label='cluster 2')
ax1.set_title('K-means clustering')

#Apply agglomerative with 2 clusters
ac = AgglomerativeClustering(n_clusters=2,
                             affinity='euclidean',
                             linkage='complete')
y_ac = ac.fit_predict(X)
ax2.scatter(X[y_ac == 0, 0], X[y_ac == 0, 1], c='lightblue',
            edgecolor='black',
            marker='o', s=40, label='cluster 1')
ax2.scatter(X[y_ac == 1, 0], X[y_ac == 1, 1], c='red',
            edgecolor='black',
            marker='s', s=40, label='cluster 2')
```

```
ax2.set_title(' Agglomerative Clustering’)
plt.legend()
plt.show()

# Now apply the DBSCAN clusterer with min_samples =4 and the eps
parameter = 0.3.
Choosing optimal values for these two parameters are critical
and using unsuitable values may highly effect to the performance
of the algorithm/ results.
In this particular case eps = 0.5 (default value) failed to
detect two distinguished clusters.

#Read more about sklearn DBSCAN clustering module -
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

db = DBSCAN(eps=0.3, min_samples=4, metric='euclidean')
y_db = db.fit_predict(X)
plt.scatter(X[y_db == 0, 0], X[y_db == 0, 1],
            c='lightblue', marker='o', s=40,
            edgecolor='black',
            label='cluster 1')
plt.scatter(X[y_db == 1, 0], X[y_db == 1, 1],
            c='red', marker='s', s=40,
            edgecolor='black',
            label='cluster 2')
plt.legend()
plt.tight_layout()
plt.show()
```