Assignment 2

Gengcong Yan - 1009903
CS-E4650 Methods of Data mining

October 22, 2021

1 Task 1

1.1 Task1.a

Compare single-link, average link, Ward's method and complete-link hierarchical clustering, when $K = [2, 8]$. Calculate Silhouette index[1] validation with different K:

Comparison of single-link, complete-link clustering methods, average link and Ward's method with $K = [2, 8]$ were implemented in program ASS2_1.ipynb (attached). The best results of each clustering methods are shown in Table 1. Single-link obtained clearly the best SI. Therefore, it looks that the single-link produced best clustering. The process code are below.

Table 1: Comparison of five clustering methods ($K = [2, 8]$) in Task 1a. SI=Silhouette index.

method	Best K	SI
single-link	2	0.657
complete-link	2	0.597
average-link	2	0.597
Ward	3	0.546

```
#linkage{'ward', 'complete', 'average', 'single'}
link="single"

for numc in range(2,9):
    cluster = AgglomerativeClustering(n_clusters=numc, affinity
       ='euclidean', linkage=link)
    predicted_labels=cluster.fit_predict(ratdata_ind)
    # print("Predicted_labels:\n",predicted_labels)
    silhouette_avg = silhouette_score(ratdata_ind,
       predicted_labels)
    print("For n_clusters ={}, Linkage method={} The average
       silhouette_score is :{}".format(numc,link,silhouette_avg
       ))
```

1.2 Task1.b

We perform PCA on original data to present new data suing only the first principal component. The best results of each clustering methods using new data are shown in Table 2. Single-link with $K = 3$ obtained clearly the best SI. Therefore, it looks that the single-link produced best clustering even after PCA.

Table 2: Comparison of five clustering methods ($K = [2, 8]$) in Task 1a. SI=Silhouette index.

method	Best K	SI
single-link	3	0.675
complete-link	2	0.622
average-link	3	0.605
Ward	3	0.620

1.3 Task1.c

From Task above, we can know the best clustering in original data and data after PCA are using single-link metric with $K = 2$ and $K = 3$, respectively. The methods didn't find the same clusters. And the clusters size are not similar, because some samples' feature number in data are apparently wrong. These rat data are divided into one cluster. we know that such as the feature *gonind* of rats 246, 258, 322 are bigger than 1. This feature can not greater than 1 because it is a proportional ratio to weight. From this task, we can know that classification is used not only to distinguish different kinds of data, but also to filter out the outliers in the data at certain times.

	0	1	2	3	4	5	6	7
0	0.000	1.000	0.571	0.333	0.750	0.571	0.750	0.571
1	1.000	0.000	1.000	0.889	0.750	0.889	0.750	0.889
2	0.571	1.000	0.000	0.750	0.750	0.333	0.571	0.750
3	0.333	0.889	0.750	0.000	0.889	0.750	0.889	0.571
4	0.750	0.750	0.750	0.889	0.000	0.750	0.333	0.889
5	0.571	0.889	0.333	0.750	0.750	0.000	0.571	0.750
6	0.750	0.750	0.571	0.889	0.333	0.571	0.000	0.889
7	0.571	0.889	0.750	0.571	0.889	0.750	0.889	0.000

Figure 1: Pairwise Jaccard distances

2 Task 2

2.1 Task2.a

The code of Jaccard distances compution are below:

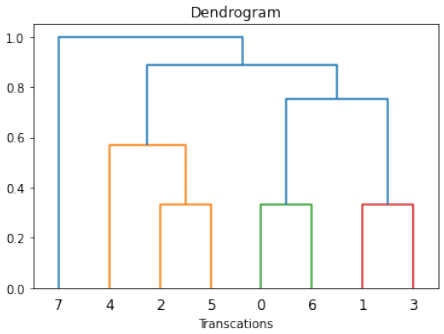
```
def jaccard_dis(arr1, arr2):
    a = set(arr1)
    b = set(arr2)
    c = a.intersection(b)
    return round(1- float(len(c)) / (len(a) + len(b) - len(c))
       ,3)

def get_all_jdis(num,data):
    j_dis=np.zeros((num,num))
    oned=[]
    for i in range(num):
        temp=[]
        for j in range(i+1,num):
            j_dis[i][j]=j_dis[j][i]=jaccard_dis(data[i],data[j
               ])
            oned.append(j_dis[i][j])
    return oned,pd.DataFrame(j_dis)
```

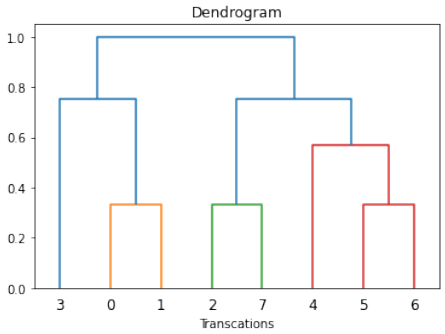
The pairwise Jaccard distances of data are shown in Fig.1.

2.2 Task2.b

we simulated the agglomerative hierarchical clustering algorithm with the complete linkage metric. It's also possible to obtain 2 different clusterings depending on data order. After shuffling the data several times, we can get the corresponding cluster dendrogram in Fig.2a and Fig.2b.



(a) Dendrogram 1



(b) Dendrogram 2

Figure 2: Dendrograms on complete-link metric

Explain why clusters merged, using Fig.2a as an example:

Firstly, every point will be seen as a cluster in the beginning, then based on the pairwise jaccard distances of data, we can find the smallest distance in the points and merged them into a new cluster. After that, we recompute the distances between clusters and merge the nearest clusters iteratively until there are only one cluster left.

$$\begin{bmatrix} 0 & 6 & 0.333 \\ 1 & 3 & 0.333 \\ 2 & 5 & 0.333 \\ 4 & 10 & 0.571 \\ 8 & 9 & 0.75 \\ 11 & 12 & 0.889 \\ 7 & 13 & 1. \end{bmatrix}$$

From the computation result above we can know, [0,6] firstly merged into new cluster 8, then $[1, 3] \rightarrow 9, [2, 5] \rightarrow 10$. Now we can see 4 and 10 are merged into new cluster 11, the cluster 10 are also newly formed from the merges before. Repeat this process until there is only one cluster, the algorithm terminated. we now obtain the clustering results in dendrgrams.

2.3 Task2.c

Now, we repeated the task above only with the single linkage metric. The results didn't change and irrelevant to data order. The dendrogram is shown in Fig.3.

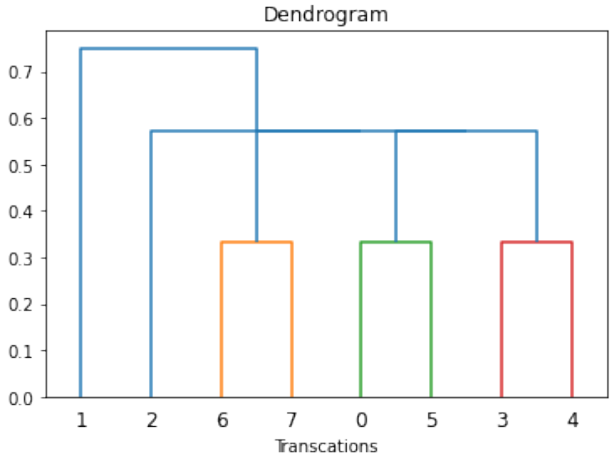


Figure 3: Dendrogram on single-link metric

3 Task3

The code for computing 3 validation indexes are below:

```
def Compute_indexs_KM(data,ground,min,max):
    print("Kmeans")
    indexs=[]
    for numc in range(min,max+1):
        KM = KMeans(n_clusters=numc)
        predicted_labels=KM.fit_predict(data)

        silhouette_avg = silhouette_score(data,
           predicted_labels)
        dh=davies_bouldin_score(data, predicted_labels)
        label_true=ground.to_numpy().reshape(1,-1)[0]
        nmi=normalized_mutual_info_score(label_true,
           predicted_labels)
        indexs.append([silhouette_avg,dh,nmi])
    return np.round(pd.DataFrame(indexs,columns=["SI","DB","NMI
       "]),3)

def Compute_indexs_SC(data,ground,min,max,method):
    print("Spectral Cluster with "+method)
    indexs=[]
    for numc in range(min,max+1):
        SC= SpectralClustering(n_clusters=numc,affinity=method)
        predicted_labels=SC.fit_predict(data)

        silhouette_avg = silhouette_score(data,
           predicted_labels)
        dh=davies_bouldin_score(data, predicted_labels)
        label_true=ground.to_numpy().reshape(1,-1)[0]
        nmi=normalized_mutual_info_score(label_true,
           predicted_labels)
        indexs.append([silhouette_avg,dh,nmi])
    return np.round(pd.DataFrame(indexs,columns=["SI","DB","NMI
       "]),3)
```

3.1 Task3.a

Abbreviations: SI=Silhouette index[1], NMI=Normalized Mutual Information, DB=Davies-Bouldin index[2].

In *balls.txt*, we preform cluster method K-means and spectral clustering using a Gaussian kernel and a Laplacian matrix with different cluster number between 2 and 5. The results are in Table5:

From Table above we can see, when $K = 3$, the clustering results are the best among

Table 3: Comparison of 3 clustering methods ($K = [2, 5]$)

method	K=2			**K=3**		
Indexs	SI	DB	NMI	SI	DB	NMI
K-means	0.668	0.523	0.735	0.902	0.136	1.000
SC (Gaussian)	0.577	0.700	0.734	0.902	0.136	1.000
SC (Laplacian)	0.668	0.523	0.735	0.902	0.136	1.000
method	K=4			K=5		
Indexs	SI	DB	NMI	SI	DB	NMI
K-means	0.710	0.653	0.905	0.519	0.941	0.832
SC (Gaussian)	0.711	0.653	0.905	0.518	0.960	0.830
SC (Laplacian)	0.708	0.657	0.904	0.514	0.976	0.827

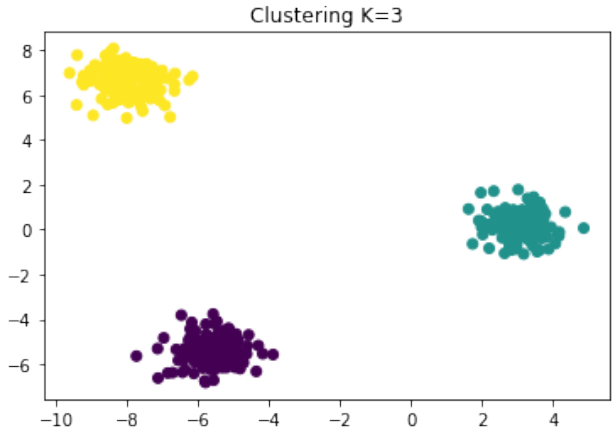


Figure 4: K-means(K=3)

different situation, and the validation index are best. The bigger SI and NMI are, the better the clustering results are. DB are opposite.we can draw the predict labels in Fig.4,5,6 for visually check. When k is not equal to 3, the image shows clearly incorrect predicted labels. So in the classification method, the predetermined K is quite important and determines the performance of the whole classification prediction. But when the amount of data is huge or the difference between the data is not obvious, it is difficult for us to judge the best k value

3.2 Task3.b

In *spirals.txt*, we preform cluster method K-means and spectral clustering using a Gaussian kernel and a Laplacian matrix with different cluster number between 2 and 5. The results are in Table5:

The NMI index can better captured the performance of Spectral Cluster with Gaussian kernel. the data in *spirals.txt* are spiral shape ,which is hard for K-means and Spectral Cluster with Laplacian to distinguish successfully. But with RBF kernel, we may get easily distinguishable results by mapping the data of the preserved features to the space of other

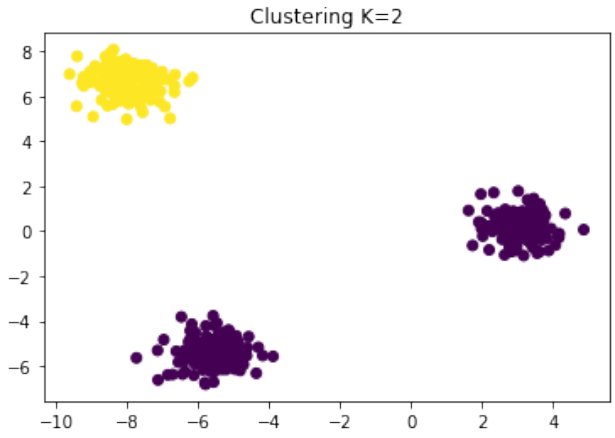


Figure 5: K-means(K=2)

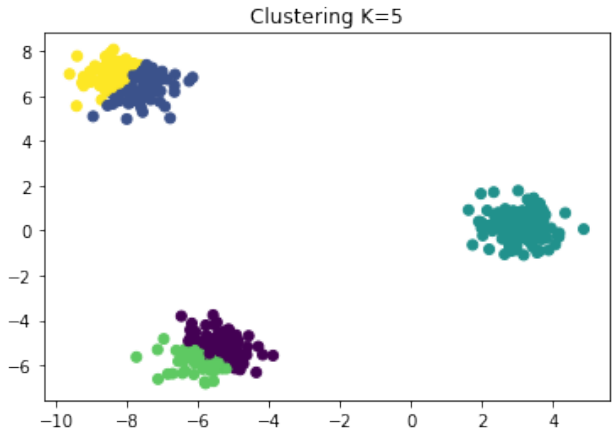


Figure 6: K-means(K=5)

Table 4: Comparison of 3 clustering methods ($K = [2, 5]$)

method	K=2			**K=3**		
Indexs	SI	DB	NMI	SI	DB	NMI
K-means	0.348	1.168	0.001	0.360	0.880	0.000
SC (Gaussian)	0.025	6.314	**0.729**	0.001	5.882	**1.000**
SC (Laplacian)	0.345	1.173	0.000	0.362	0.896	0.002
method	K=4			K=5		
Indexs	SI	DB	NMI	SI	DB	NMI
K-means	0.354	0.881	0.003	0.347	0.895	0.009
SC (Gaussian)	-0.012	6.914	**0.910**	0.015	5.459	**0.833**
SC (Laplacian)	0.330	0.927	0.052	0.279	1.818	0.134

dimensions. The graphs in Fig.7,8,9 shows the clustering results.

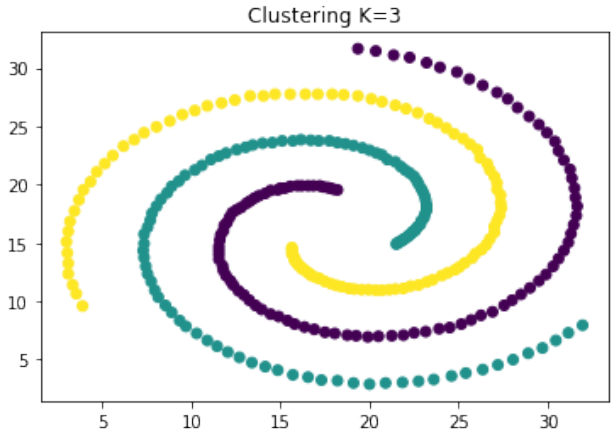


Figure 7: Spectral Cluster with RBF(K=3)

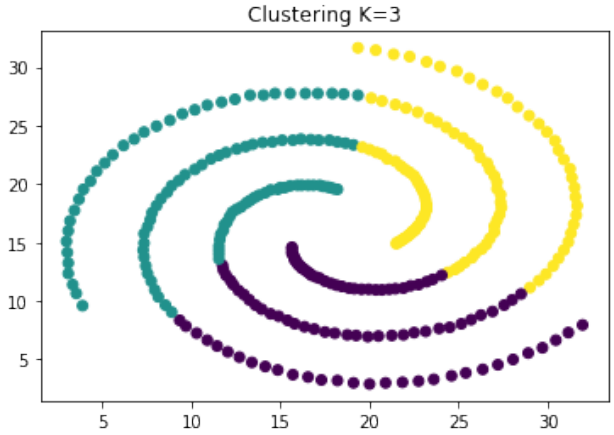


Figure 8: Spectral Cluster with Laplacian(K=3)

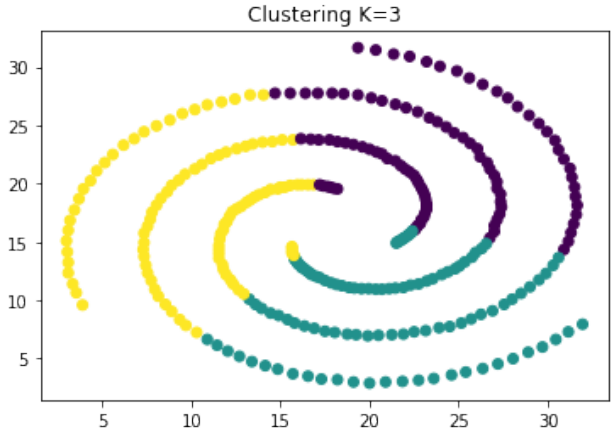


Figure 9: K-means(K=3)

3.3 Task3.c

1. SI is a measure of how similar an object is to its own cluster compared to other clusters. The silhouette ranges from -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. We average SI on all points to show the performance in general. If most points have a high value, then the clustering configuration is appropriate.

2. DH is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Clusters which are farther apart and less dispersed will result in a better score. The DB index is generally higher for convex clusters than other concepts of clusters, such as density based clusters. The usage of centroid distance limits the distance metric to Euclidean space.

3. NMI is a good external measure for determining the quality of clustering, we need the class labels of the data to determine. Since it's normalized we can measure and compare NMI between different clusterings having different number of clusters. Mutual Information tells us the reduction in the entropy of class labels that we get if we know the ground cluster labels. But external measures require the knowledge of the ground truth classes while almost never available in practice or requires manual assignment by human annotators.

NMI score in spectral cluster with gaussian kernel perform better in data of spiral shape, we may get easily distinguishable results by mapping the data of the preserved features to the space of other dimensions. NMI is an external index and gain more information, in which the comparison are between the ground truth labels and predicted labels. But SI and DB are both internal indexes, which mean the positions of data themselves are also important. They measure the performance based on the distance between data and similarity between clusters.So in plot of spirals, even though the correct classification result is obtained using spectral cluster with gaussian kernel, because the aggregation of points inside the class is not gathered together as in the ordinary case, but presents a curve. This leads to a completely wrong evaluation result if following SI and DB. Clusters which are farther apart and less dispersed will result in a better score. We can not use internal indices to determine optimal K for spectral clustering. In the exercise, the internal indices performed wrong evaluation on spiral data.

4 Task4

5 Task4.a

The code of new index compution:

```
def Gauss_sim_2d(P1,P2,gamma=1.0):
    dis=np.linalg.norm(P1-P2)
    return np.exp(-gamma * dis ** 2)

def new_validation_index(data,pred_labels):
    new_data=data.to_numpy().astype("float64")
    lens=len(new_data)
    T=0
    for i in range(lens):
        a,b=0,0
        for j in range(lens):
            if j!=i:
                sim=Gauss_sim_2d(new_data[i],new_data[j])
                a+=(pred_labels[i]==pred_labels[j])*sim
                b+=sim
        T+=a/b
    T=T/lens
    return T
```

Table 5: Comparison of 3 clustering methods ($K = [2, 5]$)

method	New Index			
K=	2	3	4	5
K-means	0.977	0.946	0.95	0.943
SC (Gaussian)	1.0	1.0	0.9965	0.9921
SC (Laplacian)	0.9782	0.975	0.9663	0.9592

Spectral clustering with RBF are still best.

5.1 Task4.b

Comparing with SI and DB index above, we can see τ index is a better choice in this spiral case. Using Spectral clustering with RBF when $K == 3$ obtain $\tau = 1.0$. But accordingly, imprecise results obtained using other methods also yielded better values with this validation method. Because this validation method currently considers only distance-based similarity. So as long as the points that are as close as possible are grouped in the same cluster, better index values can be calculated. The non-convex case of clusters is not taken into account.

5.2 Task4.c

Following the hint in the description, I took the k-nearest-neighbour graph adjacency matrix into consideration as well. Instead of calculating the similarity of all points and checking if they are in the same cluster, we now calculate the similarity if the points considered as neighbors are in the same cluster or not. The number of the nearest neighbors as a basis is a hyperparameter to be considered. The formula are below:

$$\tau = \frac{1}{n}\sum_{i=1}^{n}\frac{\sum^{j\in neighbour} c_{ij}s(X_i,X_j)}{\sum^{j\in neighbour} s(X_i,X_j)} \tag{1}$$

Based on the index above, we obtain all $\tau = 1.0$ when it performed on Spectral Cluster with Gaussian kernel and $K = 3$, the number of nearest neighbours are 5,10,20. But under the same conditions $SI = 0.001$ and $DB = 5.882$, which means almost failed classification results based on these two indexes. But In fact The performance of spectral sluster with gaussian kernel are pretty good from the graph. If all points compute their neighbors in the same clusters as themselves as much as possible, naturally this classification result performs well under this validation metric. The disadvantage of this metric is due to the need to calculate the number of k nearest neighbors, which is a hyperparameter that makes it difficult to determine an optimal value.

The code are below:

```
def indexs_using_neighbour(data,pred_labels,n_nghbours=20):

    kg=kneighbors_graph(sprialsdata_train,n_neighbors=
       n_nghbours)

    new_data=data.to_numpy().astype("float64")
    lens=len(new_data)
    T=0
    for i in range(lens):
        a,b=0,0
        for j in range(lens):
            if j!=i:
                sim=Gauss_sim_2d(new_data[i],new_data[j])
                a+= kg[i,j] *(pred_labels[i]==pred_labels[j])*
                   sim
                b+=kg[i,j] * sim
        T+=a/b
    T=T/lens
    return T
```

References

[1] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[2] D. Davies and D. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.