

## Homework Set 6, CPSC 8420, Fall 2024

Alshurbaji, Mohammad

Problem 1: The Affinity Matrix:

$$-d(x_i, x_j)^2 / \sigma$$

$$A_{ij} = e$$

$\sigma$ : User specified parameter

Steps:

- 1) Construct the affinity matrix  $A$ .
- 2) ~~the~~ Symmetrically Normalization to get matrix  $N$

$$N(i, j) = \frac{A(i, j)}{\sqrt{d(i) d(j)}} \quad ; \quad d(i) = \sum_k A(i, k)$$

- 3) Construct a matrix  $Y$ , whose columns are the first  $k$  eigenvectors of  $N$ .
- 4) Normalize each row of  $Y$  such that it is of unit length.
- 5) Cluster the dataset by running  $k$ -means on the set of the embedded points, where each row of  $Y$  is a data-point.

Q4: Comparison between  $k$ -Means and Spectral Clus.

1) Observed that  $k$ -Means and Spectral give the same results when  $\boxed{\text{Sigma} = 0.5}$  for Spectral.

2) Observed that  $k$ -means can be used when the data is randomly scattered and there is no shape for it. ~~but with~~ and it's not suitable if the data shaped. While Spectral Clustering is suitable for all cases.

3) Observed that Spectral Clustering is not good at  $\sigma = 0.5$ , which acting the same as  $k$ -Means.

# Homework set 6 - Mohammad Alshurbaji

December 3, 2024

```
[31]: import numpy as np
      from scipy.spatial.distance import pdist, squareform
      import scipy.io as sio #to read the matlab files
      from sklearn.cluster import KMeans
      import matplotlib.pyplot as plt

[32]: def load_dataset(matlabfile, variable_name):
      dataset = sio.loadmat(matlabfile)
      return dataset[variable_name]

[38]: def construct_affinity_matrix(data, sigma):
      Pdistance = squareform(pdist(data, 'euclidean'))
      #affinity_matrix = np.exp(-Pdistance**2 / (2 * sigma**2))
      affinity_matrix = np.exp(-Pdistance**2 / (2 * sigma**2))
      return affinity_matrix

      def symmetrically_normalize(affinity_matrix):
          dmatrix = np.sum(affinity_matrix, axis = 1)
          symmetrical_normalize = affinity_matrix / np.sqrt(np.outer(dmatrix, dmatrix))
          return symmetrical_normalize
      #Now: following the steps in the file to do the the spectral clustering
      def spectral_clustering(data, k, sigma):
          #Constructing the affinity matrix
          affinity_matrix = construct_affinity_matrix(data, sigma)
          #Symmetrical Normalization
          normalized_affinity = symmetrically_normalize(affinity_matrix)
          #eigenvalues and eigenvectors
          eigenvalues, eigenvectors = np.linalg.eigh(normalized_affinity)
          top_k_eigenvectors = eigenvectors[:, -k:]
          #Normalize rows of eigenvectors
          norms = np.linalg.norm(top_k_eigenvectors, axis = 1, keepdims = True)
          normalized_eigenvectors = top_k_eigenvectors / norms
          #K-means
          kmeans = KMeans(n_clusters = k, random_state = 0)
          clusters = kmeans.fit_predict(normalized_eigenvectors)
          return clusters
      #Plotting the clusters
```

```

def plot_clusters(data, clusters, title):
    plt.scatter(data[:,0], data[:,1], c = clusters, cmap='viridis', s=10)
    plt.title(title)
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.show()

#Plotting the eigenvalues
def plot_eigenvalues(normalized_affinity, dataset):
    eigenvalues = np.linalg.eigvalsh(normalized_affinity)
    sorted_eigenvalues = np.sort(eigenvalues)[::-1]
    plt.plot(range(1,11), sorted_eigenvalues[:10], '-o')
    plt.title(f'First 10 Eigenvalues: {dataset} and sigma = 0.05')
    plt.xlabel('Index')
    plt.ylabel('Eigenvalue')
    plt.show()

```

```

[40]: variable_mapping = {
        'concentric.mat': 'X1',
        'rectangles.mat': 'X2',
        'links.mat': 'X3',
        'text.mat': 'X4'
    }
    sigmas = [0.025,0.05,0.2,0.5]
    datasets = ['concentric.mat', 'links.mat', 'rectangles.mat', 'text.mat']
    k_values = [2,2,2,6]
    for i, dataset in enumerate(datasets):
        variable_name = variable_mapping[dataset]
        data = load_dataset(dataset, variable_name)
        k = k_values[i]
        #First part of the problem: K-means only
        kmeans = KMeans(n_clusters=k, n_init=10, random_state=0)
        clusters = kmeans.fit_predict(data)
        plot_clusters(data, clusters, f'K-Means for {dataset}')
        #Spectral Clustering
        for sigma in sigmas:
            clusters = spectral_clustering(data, k, sigma)
            plot_clusters(data, clusters, f'Spectral Clustering for {dataset},  

            ↪Sigma = {sigma}')

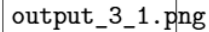
            if dataset in ['rectangles.mat', 'text.mat'] and sigma == 0.05:
                affinity_matrix = construct_affinity_matrix(data, 0.05)
                normalized_affinity = symmetrically_normalize(affinity_matrix)
                plot_eigenvalues(normalized_affinity, dataset)

```

C:\Users\Mohammad\anaconda3\Lib\site-packages\sklearn\cluster\\_kmeans.py:1382:  
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when

there are less chunks than available threads. You can avoid it by setting the environment variable OMP\_NUM\_THREADS=2.

```
warnings.warn(
```



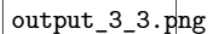
output\_3\_1.png

```
C:\Users\Mohammad\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:  
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in  
1.4. Set the value of `n_init` explicitly to suppress the warning
```

```
warnings.warn(
```

```
C:\Users\Mohammad\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:  
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when  
there are less chunks than available threads. You can avoid it by setting the  
environment variable OMP_NUM_THREADS=2.
```

```
warnings.warn(
```



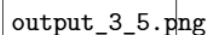
output\_3\_3.png

```
C:\Users\Mohammad\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:  
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in  
1.4. Set the value of `n_init` explicitly to suppress the warning
```

```
warnings.warn(
```

```
C:\Users\Mohammad\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:  
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when  
there are less chunks than available threads. You can avoid it by setting the  
environment variable OMP_NUM_THREADS=2.
```

```
warnings.warn(
```



output\_3\_5.png

Here are the  
generated charts:

