In this homework, I implemented a spectral clustering algorithm in Python.

First imported libraries which will be needed in project. multivariate_normal is to create multivariate normal Gaussians from specific mean vector and covariance matrix. numpy.linalg.eig is to create eigen values and eigen vectors.

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.spatial as spa

from scipy.stats import multivariate_normal
from numpy.linalg import eig # to create eigen vectors
from numpy.linalg import matrix_power
```

Figure 1: import libraries

Part 1

I read data from hw08_data_set which contains 300 data points generated randomly from five bivariate Gaussian densities.

Part 2

First I defined Euc_Distance(point1, point2) function to calculate Euclidean distances.

I defined threshold delta parameter to the 1.25 as said in the pdf. Then, I created B matrix which is connectivity matrix by below formula.

$$b_{ij} = \begin{cases} 1, & \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 < \delta \\ 0, & \text{otherwise.} \end{cases}$$
$$b_{ii} = 0$$

Figure 2: Connectivity matrix formula

After defining B matrix, visualized connectivity matrix as follows:

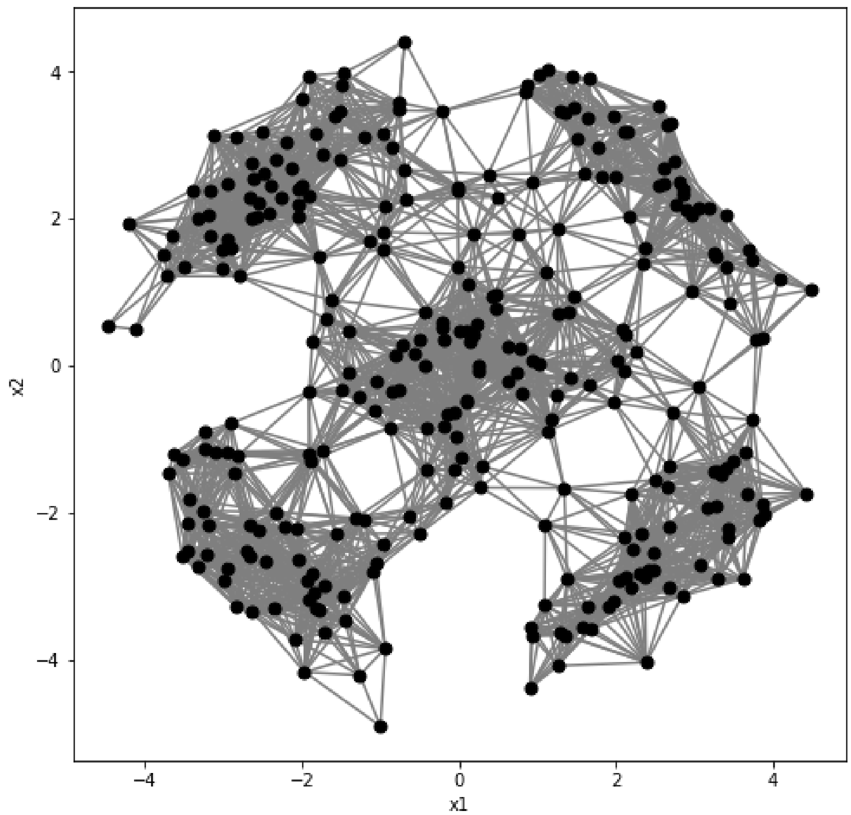

Figure 3: Connectivity matrix(B) visualization

Part 3

In this part, I calculated **D** and **L** matrices as described in the lecture notes.

$$d_{ii} = \sum_{j \neq i} b_{ij} \quad \forall i$$

Laplacian Matrix: $L_{N \times N} = D_{N \times N} - B_{N \times N}$

↳ each row (column) sums up to 0.

$$L_{\text{SYMMETRIC}} = D^{-1/2} . L . D^{-1/2} = D^{-1/2} . (D - B) D^{-1/2} = \boxed{I - D^{-1/2} . B . D^{-1/2}}$$

Figure 4: D,L matrices formulas given in the lectures

```
print(D)
```

```
[[20.  0.  0. ...  0.  0.  0.]
 [ 0. 10.  0. ...  0.  0.  0.]
 [ 0.  0. 24. ...  0.  0.  0.]
 ...
 [ 0.  0.  0. ... 21.  0.  0.]
 [ 0.  0.  0. ...  0. 33.  0.]
 [ 0.  0.  0. ...  0.  0. 14.]]
```

```
print(L)
```

```
[[1. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]
```

.

Figure 5: D,L matrices outputs

Part 4

In this part I found eigenvectors and eigenvalues of normalized Laplacian matrix using linalg.eig function. Then I took 5 eigenvectors corresponding to the smallest eigenvalues to put them to the Z matrix .

$$\underline{\underline{Z}} = \begin{bmatrix} v_1 & v_2 & \cdots & v_R \end{bmatrix}_{N \times R}$$

↳ 1st smallest eigenvector ↳ R^{th} smallest eigenvector

Figure 6: Z matrix representation given in the lectures

```
R = 5
eigenValues , eigenVectors = eig(L)
Z = eigenVectors[:,np.argsort(eigenValues)[1:R+1]]
print(Z)
```

```
[[ 0.00225332  0.02970626 -0.1215077  -0.05350856 -0.05881185]
 [ 0.01934478  0.01583822 -0.0602155  -0.02464899  0.12571926]
 [ 0.00694396  0.02992297 -0.12951246 -0.05271498  0.05379749]
 ...
 [ 0.00067333  0.00106953 -0.03446799  0.06659001  0.03322699]
 [-0.01344962 -0.01237144 -0.01390019  0.11784073 -0.01399391]
 [-0.025202    0.03167207 -0.00258001  0.00728212 -0.03877666]]
```

Figure 7: Z matrix outputs

Part 5

In this part I run k-means clustering algorithm on **Z** matrix to find $K = 5$ clusters.

First, I assigned 29, 143, 204, 271, and 277 rows of Z matrix as initial centroid. Then used similar codes with lab11. However, in this time, we work on the Z matrix instead of X matrix as whole. After iterations we lastly updated centroids according to the found memberships.

Part 6

Last I plotted the clustering results obtained by k-means.

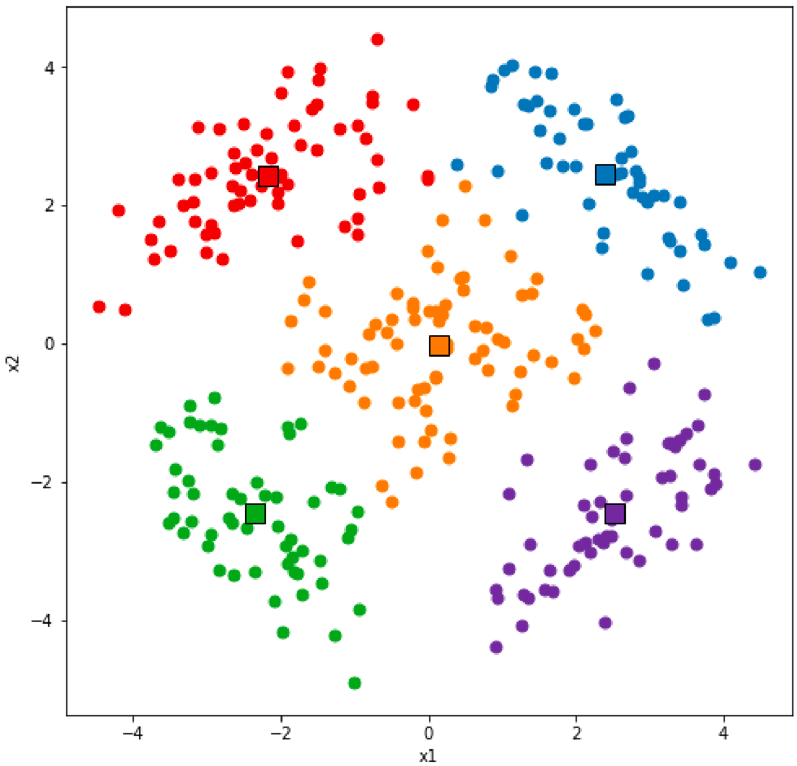

Figure 8: Visualization of clustering obtained by k-means on Z matrix