Problem 8.1 (2 points). *Let $A \in \mathbb{R}^{n\times m}$. The Singular Values Decomposition (SVD) tells us that there exists two orthogonal matrices $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ and a matrix $\Sigma \in \mathbb{R}^{n\times m}$ such that $\Sigma_{1,1} \geq \Sigma_{2,2} \geq \cdots \geq 0$ and $\Sigma_{i,j} = 0$ for $i \neq j$*

$$A = U\Sigma V^{\mathsf{T}}.$$

The columns $u_1, \ldots, u_n$ of U (respectively the columns $v_1, \ldots, v_m$ of V) are called the left (resp. right) singular vectors of A. The non-negative numbers $\sigma_i \overset{\text{def}}{=} \Sigma_{i,i}$ are the singular values of A. Moreover we also know that $r \overset{\text{def}}{=} \operatorname{rank}(A) = \#\{i \,|\, \Sigma_{i,i} \neq 0\}$.

(**a**) *Let* $\widetilde{U} = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_r \\ | & & | \end{pmatrix} \in \mathbb{R}^{n\times r}$, $\widetilde{V} = \begin{pmatrix} | & & | \\ v_1 & \cdots & v_r \\ | & & | \end{pmatrix} \in \mathbb{R}^{m\times r}$ *and* $\widetilde{\Sigma} = \operatorname{Diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}$.
Show that $A = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^{\mathsf{T}}$.
Hint: show that the ijth entry of $U\Sigma V^T$ and $\tilde{U}\tilde{\Sigma}\tilde{V}^T$ match for all i, j.

We have the following derivations:

$$\begin{aligned}
\left(U\Sigma V^{\top}\right)_{ij} &= \sum_{k=1}^{n} (U\Sigma)_{ik} \left(V^{\top}\right)_{kj} && \text{(By Definition)} \\
&= \sum_{k=1}^{n} \sigma_k \left(\vec{u}_k\right)_i \left(\vec{v}_k^{\top}\right)_j && \text{(Extract the columns from U,V)} \\
&= \sum_{k=1}^{r} \sigma_k \left(\bar{u}_k\right)_i \left(\vec{v}_k^{\top}\right)_j && \text{(Omit the last few zero singular values)} \\
&= \sum_{k=1}^{r} (\tilde{U}\tilde{\Sigma})_{ik} \left(\tilde{V}^{\top}\right)_{kj} && \text{(Bring back the pieces)} \\
&= \left(\tilde{V}\tilde{\Sigma}\tilde{V}^{\top}\right)_{ij} && \text{(Assemble the results)}
\end{aligned}$$

(**b**) *Give orthonormal bases of* $\operatorname{Ker}(A)$ *in terms of the singular vectors of A.*

Claim: An orthonormal basis for $\ker(A)$ *should be V_{m-r} where V_{m-r} is the last $m-r$ columns of V.*
Proof. *We will show that* $\operatorname{Col}(V_{m-r}) = \ker(A)$. $\forall \vec{x} \in \operatorname{Col}(V_{m-r}), \vec{x} = V_{m-r}\vec{y}$ *for some* $\vec{y} \in R^{m-r}$ *then* $A\vec{x} = U_r\Sigma_r V_r^{\top} V_{m-r}\vec{y} = 0$, $\therefore \vec{x} \in \ker(A)$, $\therefore \operatorname{Col}(V_{m-r}) \subseteq \ker(A)$.

$$\begin{aligned}
&\because \dim(\operatorname{Col}(V_{m-r})) = m - r = m - \dim(\operatorname{Im}(A)) \\
&= m - \operatorname{rank}(A) \\
&= \dim(\ker(A)) \\
&\therefore \operatorname{Col}(V_{m-r}) = \ker(A)
\end{aligned}$$

$\therefore V_{m-r}$ *is an orthonormal basis for* $\ker(A)$, *which are just the last m-r right singular vectors of A.* □

Problem 8.2 (1 points)**.** *Given matrix $A \in \mathbb{R}^{n\times m}$ where $n > m$. Assume $A^\top A \in \mathbb{R}^{m\times m}$ is invertible, we define the projection matrix*

$$P = A(A^\top A)^{-1}A^\top \in \mathbb{R}^{n\times n}.$$

Show that P is symmetric, and its eigenvalues (in descending order) are given as $\lambda_1 = \lambda_2 = ... = \lambda_m = 1$, and $\lambda_{m+1} = ... = \lambda_n = 0$.
Hint: use the SVD of A.

Since by compact SVD, $A = U_m \sum_m V_m^\top$ since rank $\left(A^\top A\right) = \text{rank}(A) = m$ Where U_m is the first column of U, V_m is the first column of V Σ_m is the diagonal matrix from the first m diagonal entries of Σ $\therefore$ We will have the following:

$$\begin{aligned}
P &= \left(U_m\Sigma_m V_m^\top\right)\left(V_m\Sigma_m^\top U_m^\top U_m\Sigma_m V_m^\top\right)^{-1} V_m\Sigma_m^\top U_m^\top \\
&= U_m\Sigma_m V_m^\top V_m\Sigma_m^{-1}\Sigma_m^{-1}V_m^\top V_m\Sigma_m U_m^\top \\
&= U_m\Sigma_m\left(V_m^\top V_m\right)\Sigma_m^{-2}\left(V_m^\top V_m\right)\Sigma_m U_m^\top \\
&= U_m U_m^\top
\end{aligned}$$

$\therefore P^\top = U_m U_m^\top = P \Rightarrow P$ is symmetric We know that $P = U_m U_m^\top$

$$\begin{aligned}
&= U_m I_m U_m^\top \\
&= [U_m \mid U_{n-m}]\left[\frac{I_{m\times m} \mid 0}{0 \mid 0}\right]\left[\frac{U_m^\top}{U_{(n-m)}}\right] \\
&= UDU^\top
\end{aligned}$$

$$\therefore \lambda_1 = \lambda_2 = \ldots = \lambda_m = 1, \lambda_{m+1} = \ldots = \lambda_n = 0$$

Problem 8.3 (4 points). *In the lecture we focused on the graph Laplacian*

$$L = D - A,$$

where A is the adjacency matrix of the graph, and $D = \mathrm{Diag}(\deg(1), \ldots, \deg(n))$. Observe that the eigenvalues of the matrices A, L can scale with the number of vertices in the graph (which may be very large in practical settings). In this question, we will see that by applying proper normalization, the matrix eigenvalues can be much more well-behaved.

We introduce the normalized adjacency matrix and the normalized Laplacian as follows,

$$\tilde{A} = D^{-1/2} A D^{-1/2}, \quad \tilde{L} = D^{-1/2} L D^{-1/2} = \mathrm{Id}_n - \tilde{A},$$

where $D^{-1/2} = \mathrm{Diag}(\deg(1)^{-1/2}, \ldots, \deg(n)^{-1/2})$. Here we assume $\deg(i) \neq 0$ for all $i \in [n]$.

(**a**) *Show that the smallest eigenvalue of the normalized Laplacian $\lambda_{\min}(\tilde{L}) = 0$.*

We first show that $\tilde{L}$ is PSD. Denote the edge set of G to be E, then:

$$\begin{aligned}
\because \vec{x}^\top \tilde{L} \vec{x} &= \vec{x}^\top D^{-1/2} L D^{-1/2} \vec{x} && \text{(By definition)} \\
&= \vec{y}^\top L \vec{y} && \text{(Change of variable)} \\
&= \sum_{(i,j)\in E} (y_i - y_j)^2 \geqslant 0 && \text{(By quadratic definition)} \\
\therefore \tilde{L} \text{ is } & \text{ P.S.D. } \therefore \lambda_{\min}(\tilde{L}) \geqslant 0
\end{aligned}$$

Proof. *We next prove for P.S.D matrices $A_{n\times n}, \vec{x} \in \ker(A) \Leftrightarrow \vec{x}^\top A \vec{x} = 0$*
$(\Rightarrow) : \forall \vec{x} \in \mathrm{Ker}(A)$, we have $A\vec{x} = 0, \therefore \vec{x}^\top A \vec{x} = 0$
$(\Leftarrow) :\because A$ is P.S.D. by spectrum the orem we have $A = UDU^\top$

$$\begin{aligned}
\therefore \vec{x}^\top A \vec{x} = 0 &\Leftrightarrow \vec{x}^\top U D U^\top \vec{x} = 0 \\
&\Leftrightarrow U^\top \vec{x} D U^\top \vec{x} = 0 \\
&\Leftrightarrow \sum_{i=1}^{n} \lambda_i \left(\vec{u}_i^\top \vec{x}\right)^2 = 0 \\
&\Leftrightarrow \sum_{i=1}^{n} \left(\sqrt{\lambda_i} \vec{u}_i^\top \vec{x}\right)^2 = 0 \\
&\Leftrightarrow \sqrt{\lambda_i} \vec{u}_i^\top \vec{x} = 0 \quad \forall i = 1, 2, \cdots, n \\
&\Leftrightarrow D^{\frac{1}{2}} U^\top \vec{x} = \overrightarrow{0}
\end{aligned}$$

$$\begin{aligned}
\therefore A\vec{x} = UDU^\top \vec{x} &= U D^{\frac{1}{2}} U^\top U \left(D^{\frac{1}{2}} U^\top \vec{x}\right) \\
&= 0
\end{aligned}$$

$$\therefore \vec{x}^\top A \vec{x} = 0 \Leftrightarrow A\vec{x} = 0$$

□

Finally, we could prove $\lambda_{min}(\tilde{L}) = 0$.

$$\begin{aligned}
\because \vec{x}^\top \tilde{L} \vec{x} = 0 &\Leftrightarrow \sum_{(i,j)\in E} (x_i - x_j)^2 = 0 \\
&\Leftrightarrow x_i = x_j \quad \forall (i,j) \in E \\
\because \vec{x}^\top \tilde{L} \vec{x} = 0 &\Leftrightarrow \vec{x} \in \ker(\tilde{L})
\end{aligned}$$

$\therefore$ *There exists $\vec{x} \in \ker(\tilde{L})$such that*

$$x_i = x_j, \forall (i,j) \in E.$$
$$\therefore \lambda_{\min}(\tilde{L}) = 0$$

(**b**) *Show that the largest eigenvalue of the normalized Laplacian can be written as*

$$\lambda_{\max}(\tilde{L}) = \max_{v \in \mathbb{R}^n, v \neq 0} \frac{v^\top L v}{v^\top D v}.$$

Based on this formulation, prove that $\lambda_{\max}(\tilde{L}) \leq 2$.
Hint: Use the quadratic form definition $x^\top L x = \sum_{i \sim j}(x_i - x_j)^2$.
Since by Rayleigh Quotient we have:

$$\lambda_{\max}(\tilde{L}) = \max_{\vec{v} \in R^n, \vec{v} \neq 0} \frac{\vec{v}^\top \tilde{L} \vec{v}}{\vec{v}^\top \vec{v}}$$
$$\therefore \frac{\vec{v}^\top \tilde{L} \vec{v}}{\vec{v}^\top \vec{v}} = \frac{\vec{v}^\top D^{-1/2} L D^{-1/2} \vec{v}}{\vec{v}^\top \vec{v}}$$

Letting $\vec{t}^\top = \vec{v}^\top D^{-1/2}$ *and* $\vec{v} = D^{1/2}\vec{t}$

$$\therefore \frac{\vec{v}^\top \tilde{L} \vec{v}}{\vec{v}^\top \vec{v}} = \frac{\vec{t}^\top L \vec{t}}{\vec{t}^\top D \vec{t}}$$

$\because D^{1/2}$ *is an invertible matrix by assumption*

$$\therefore \max_{\vec{v} \in R^n, \vec{v} \neq 0} \frac{\vec{v}^\top \tilde{L} \vec{v}}{\vec{v}^\top \vec{v}} = \max_{\vec{t} \in R^n, \vec{t} \neq 0} \frac{\vec{t}^\top L \vec{t}}{\vec{t}^\top D \vec{t}}$$

, which concludes our proof for the first part.
To show the second part, we first prove that $\vec{v}^\top D \vec{v} = \sum_{(i,j)E} v_i^2 + v_j^2$.
Proof. *Since for each edge we add to the graph G,*

$$\Delta D = \begin{bmatrix} 0 & & & 0 \\ & 1 & 0 & \\ & 0 & 1 & \\ 0 & & & 0 \end{bmatrix}$$

, where only the $\Delta D_{ii} = \Delta_{jj} = 1$ *and all other entries are zero's.*

$$\therefore \vec{v}^\top D \vec{v} = \sum_{(i,j) \in E} \vec{v}^\top \Delta D \vec{v}$$
$$= \sum_{(i,j) \in E} v_i^2 + v_j^2$$

□

Then we can use this lemma to prove the final result as follows:

$$\therefore \frac{\vec{v}^\top L \vec{v}}{\vec{v}^\top D \vec{v}} = \frac{\sum_{(i,j) \in E} (v_i - v_j)^2}{\sum_{(i,j) \in E} {v_i}^2 + {v_j}^2}$$
$$= \frac{\sum_{(i,j) \in E} v_i^2 - 2 v_i v_j + v_j^2}{\sum_{(i,j) \in E} v_i^2 + v_j^2}$$
$$\leqslant \frac{\sum_{(i,j) \in E} v_i^2 + v_j^2 + v_i^2 + v_j^2}{\sum_{(i,j) \in E} v_i^2 + v_j^2}$$
$$= 2$$
$$\therefore \max_{\vec{V} \in R^n, \vec{v} \neq 0} \frac{\vec{v}^\top L \vec{v}}{\vec{V}^\top D \vec{v}} = 2 = \lambda_{\max}(\tilde{L}).$$

(**c**) *Show that for all* $i \in [n]$*, we have* $|\lambda_i(\tilde{A})| \leq 1$*, that is, all the eigenvalues of* $\tilde{A}$ *fall between* -1 *and* 1.
Hint: apply the conclusion in the previous question.
We have the following reasoning:

$$
\begin{aligned}
&\textit{Since } \tilde{L} = Id_n - \tilde{A} \\
&\therefore \lambda_k(\tilde{L}) = 1 - \lambda_k(\tilde{A}) \\
&\because 0 \leq \lambda_k(\tilde{L}) \leq 2 \quad \forall k = 1, 2, \cdots, n. \\
&\therefore -1 \leq \lambda_k(\tilde{A}) \leq 1 \quad \forall k = 1, 2, \cdots, n \\
&\therefore \left|\lambda_k(\tilde{A})\right| \leq 1 \quad \forall k = 1, 2, \cdots, n.
\end{aligned}
$$

Problem 8.4 (3 points)**.** *The goal of this problem is to use spectral clustering techniques on real data. The file* `adjacency.txt` *contains the adjacency matrix of a graph taken from a social network. This graphs has* $n = 328$ *nodes (that corresponds to users). An edge between user* i *and user* j *means that* i *and* j *are "friends" in the social network. The notebook* `friends.ipynb` *contains functions to read the adjacency matrix as well as instructions/questions.*

Details on coding: *In this question, you should use the normalized Laplacian matrix defined in the previous question:*

$$\tilde{L} = D^{-1/2} L D^{-1/2} = \mathrm{Id}_n - D^{-1/2} A D^{-1/2},$$

rather than the original L*. The reason for using this normalized version is that the "unnormalized Laplacian"* L *does not perform well when the degrees in the graph are very broadly distributed, i.e. very heterogeneous. In such situations, the normalized Laplacian* $\tilde{L}$ *is supposed to lead to a more consistent clustering.*

Submission: *It is intended that you code in Python and use the provided Jupyter Notebook. Please only submit a pdf version of your notebook (right-click → 'print' → 'Save as pdf').*

Jupyter notebook pdf link
Jupyter notebook link

In [6]:
```
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

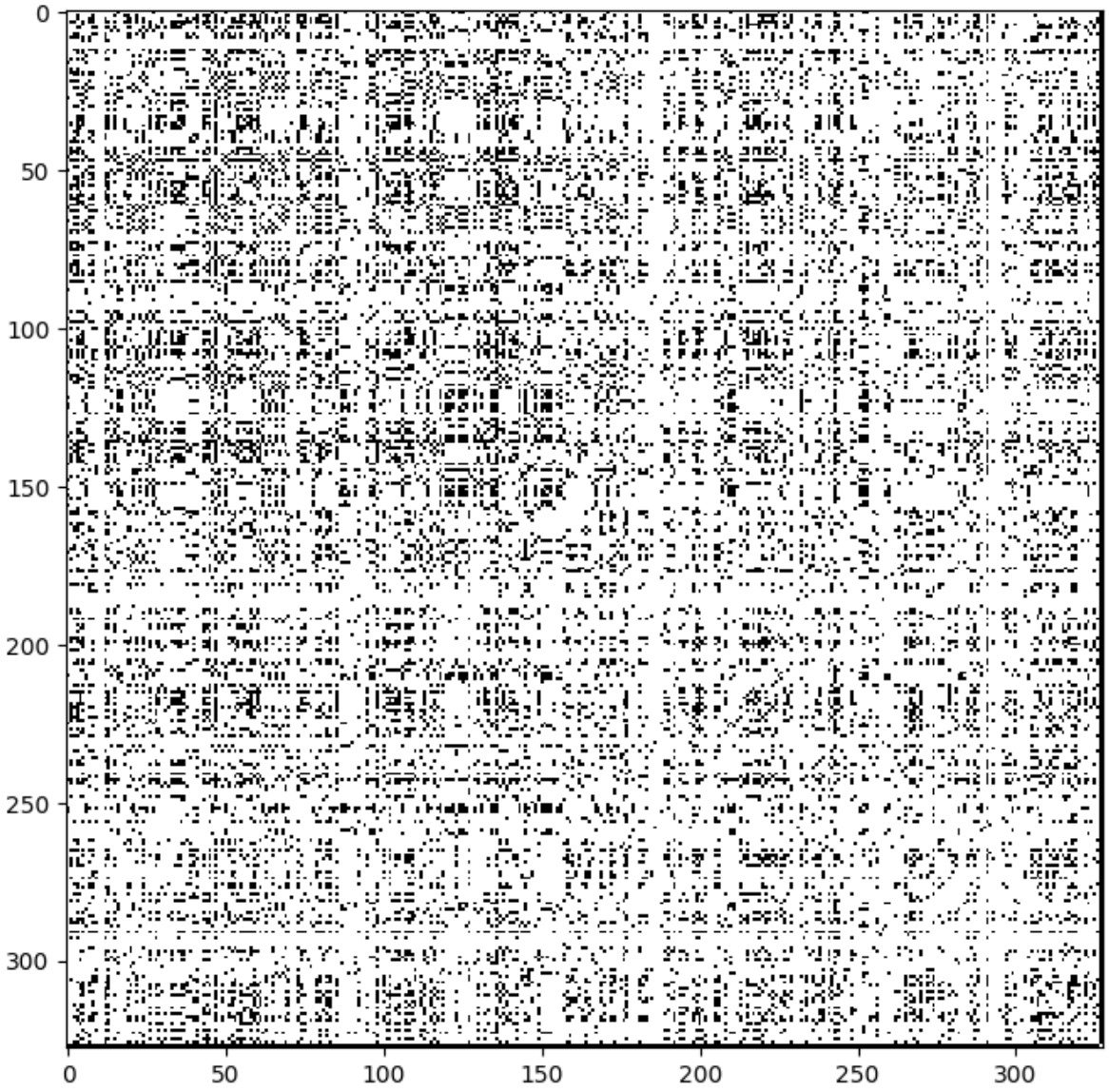
In [7]:
```
# Reads the adjacency matrix from file
A = np.loadtxt('adjacency.txt')
print(f'There are {A.shape[0]} nodes in the graph.')
```

```
There are 328 nodes in the graph.
```

As you can see above, the adjacency matrix is relatively large (328x328): there are 328 persons in the graph. In order to visualize this adjacency matrix, it is convenient to use the 'imshow' function. This plots the 328x328 image where the pixel (i,j) is black if and only if A[i,j]=1.

In [8]:
```
plt.figure(figsize=(8,8))
plt.imshow(A,aspect='equal',cmap='Greys',  interpolation='none')
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x17ae385c220>
```



(a) Construct in the cell below the degree matrix:

$$D_{i,i} = \deg(i) \qquad \text{and} \qquad D_{i,j} = 0 \ \text{ if } i \neq j,$$

the Laplacian matrix:

$$L = D - A$$

and the normalized Laplacian matrix:

$$\tilde{L} = D^{-1/2} L D^{-1/2}.$$

In [25]:
```
# Your answer here
D = np.diag(A.sum(axis = 1))
L = D - A
D_sqrt = np.diag(np.power(A.sum(axis=1),-0.5))
L_norm = D_sqrt @ L @ D_sqrt
L_norm
```

Out[25]:
```
array([[ 1.        ,  0.        ,  0.        , ...,  0.        ,
         0.        , -0.01064251],
       [ 0.        ,  1.        ,  0.        , ...,  0.        ,
         0.        , -0.00606998],
       [ 0.        ,  0.        ,  1.        , ...,  0.        ,
        -0.01628656, -0.00685914],
       ...,
       [ 0.        ,  0.        ,  0.        , ...,  1.        ,
         0.        , -0.00680698],
       [ 0.        ,  0.        , -0.01628656, ...,  0.        ,
         1.        , -0.00726126],
       [-0.01064251, -0.00606998, -0.00685914, ..., -0.00680698,
        -0.00726126,  1.        ]])
```

(b) Using the command 'linalg.eigh' from numpy, compute the eigenvalues and the eigenvectors of $\tilde{L}$.

In [26]:
```
# Your answer here
eigenvalues, eigenvectors = np.linalg.eigh(L_norm)
print(eigenvalues, eigenvectors)
```

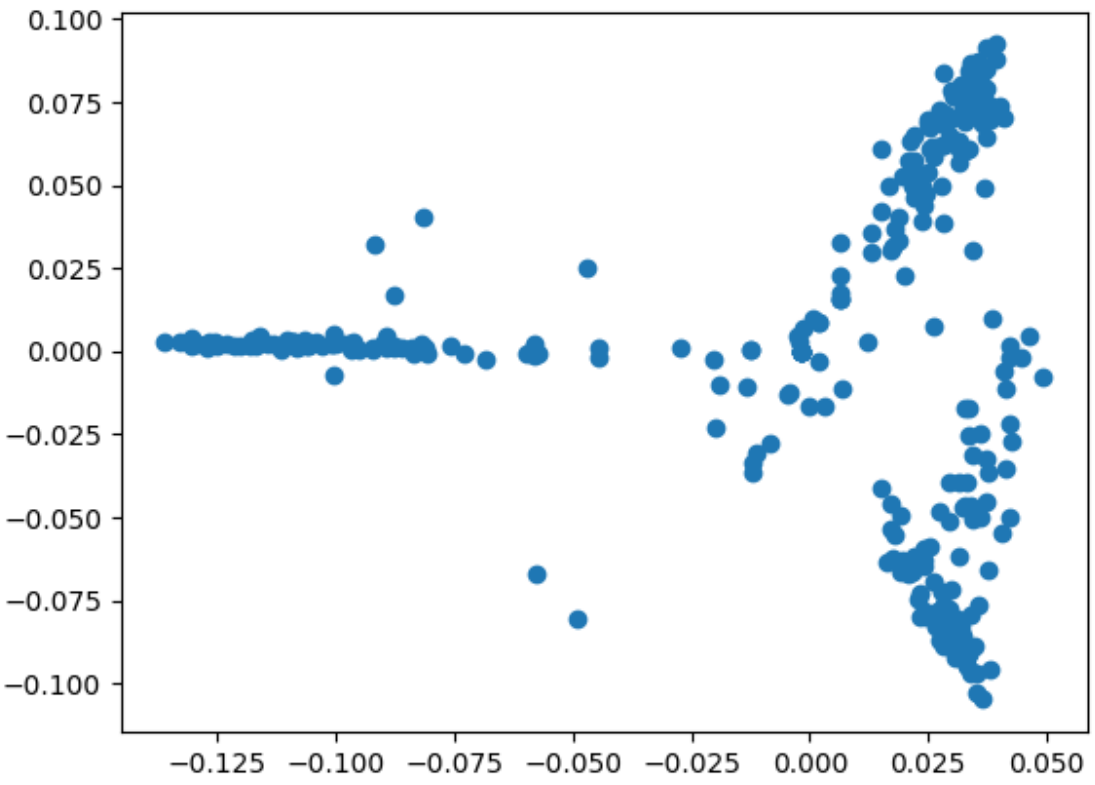
```
[-8.60335003e-16  8.22971080e-02  2.73920284e-01  2.86756239e-01
  4.12340841e-01  4.41921035e-01  6.16054279e-01  6.70249866e-01
  7.06406288e-01  7.25463028e-01  7.35075504e-01  7.52268234e-01
  7.71169767e-01  7.73015222e-01  7.88819638e-01  7.93830094e-01
  8.03472018e-01  8.14921509e-01  8.21827484e-01  8.27206976e-01
  8.35085854e-01  8.38589162e-01  8.46760685e-01  8.56016240e-01
  8.58888980e-01  8.61157975e-01  8.65395654e-01  8.68032309e-01
  8.71228049e-01  8.75612490e-01  8.79472847e-01  8.81071746e-01
  8.83567703e-01  8.85609635e-01  8.87491226e-01  8.90039358e-01
  8.93125892e-01  8.96071979e-01  8.97872008e-01  9.00108767e-01
  9.02215754e-01  9.04086375e-01  9.06058925e-01  9.07454646e-01
  9.09056960e-01  9.09610504e-01  9.13082816e-01  9.13504100e-01
  9.15522027e-01  9.16340436e-01  9.17403135e-01  9.19270844e-01
  9.22130905e-01  9.23161207e-01  9.23511054e-01  9.24832666e-01
  9.28088113e-01  9.29143852e-01  9.30923431e-01  9.32402646e-01
  9.33870511e-01  9.35555401e-01  9.36692116e-01  9.38080974e-01
  9.39859181e-01  9.41188489e-01  9.42451551e-01  9.44524080e-01
  9.45495347e-01  9.46814646e-01  9.47076227e-01  9.48401324e-01
  9.48702506e-01  9.50156125e-01  9.50672500e-01  9.51089565e-01
  9.52229499e-01  9.54700979e-01  9.55973606e-01  9.57012037e-01
  9.59007875e-01  9.59340528e-01  9.60095483e-01  9.61813213e-01
  9.62496471e-01  9.63666669e-01  9.63774402e-01  9.64619656e-01
  9.65020720e-01  9.65381367e-01  9.66932735e-01  9.67219909e-01
  9.69151813e-01  9.69662636e-01  9.70255098e-01  9.71038681e-01
  9.71844822e-01  9.72518729e-01  9.73098475e-01  9.74397971e-01
  9.74765109e-01  9.76393231e-01  9.77111815e-01  9.77513934e-01
  9.78096613e-01  9.79256225e-01  9.79893301e-01  9.80091360e-01
  9.81342758e-01  9.81868729e-01  9.82715996e-01  9.83755570e-01
  9.84490267e-01  9.85156265e-01  9.85601087e-01  9.86227000e-01
  9.87272162e-01  9.87546526e-01  9.88666227e-01  9.88890536e-01
  9.89439372e-01  9.89939160e-01  9.90497496e-01  9.91449441e-01
  9.92825820e-01  9.93252781e-01  9.94124653e-01  9.94843424e-01
  9.95395267e-01  9.96278082e-01  9.97150885e-01  9.98352354e-01
  9.98574988e-01  9.98971618e-01  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00029329e+00  1.00060897e+00  1.00123040e+00
  1.00217855e+00  1.00264326e+00  1.00318027e+00  1.00360296e+00
  1.00394648e+00  1.00452451e+00  1.00493913e+00  1.00585384e+00
  1.00682744e+00  1.00754310e+00  1.00846734e+00  1.00910240e+00
  1.00984873e+00  1.01066596e+00  1.01081159e+00  1.01131858e+00
  1.01173508e+00  1.01252273e+00  1.01264331e+00  1.01317383e+00
  1.01406340e+00  1.01495706e+00  1.01529273e+00  1.01598070e+00
  1.01700814e+00  1.01735886e+00  1.01789071e+00  1.01846635e+00
  1.01865757e+00  1.01941545e+00  1.02005446e+00  1.02114281e+00
  1.02136189e+00  1.02232889e+00  1.02256850e+00  1.02303408e+00
  1.02348028e+00  1.02382565e+00  1.02407066e+00  1.02531368e+00
  1.02605812e+00  1.02650922e+00  1.02730736e+00  1.02792683e+00
  1.02874064e+00  1.02939545e+00  1.02974518e+00  1.03062856e+00
  1.03145498e+00  1.03241489e+00  1.03278909e+00  1.03328519e+00
  1.03474330e+00  1.03574221e+00  1.03671091e+00  1.03693224e+00
  1.03748780e+00  1.03774168e+00  1.03846617e+00  1.03882506e+00
  1.03914011e+00  1.04164351e+00  1.04213483e+00  1.04249023e+00
  1.04303488e+00  1.04390826e+00  1.04485040e+00  1.04495977e+00
  1.04512557e+00  1.04659637e+00  1.04815093e+00  1.04847570e+00
  1.04879178e+00  1.05002517e+00  1.05073620e+00  1.05111870e+00
```

```
 1.05182571e+00  1.05278393e+00  1.05335259e+00  1.05422111e+00
 1.05520493e+00  1.05628715e+00  1.05670284e+00  1.05759106e+00
 1.05874514e+00  1.05907374e+00  1.05967862e+00  1.06110234e+00
 1.06149099e+00  1.06207963e+00  1.06310380e+00  1.06331023e+00
 1.06488980e+00  1.06609391e+00  1.06761350e+00  1.06873409e+00
 1.06901424e+00  1.06983455e+00  1.07030620e+00  1.07189652e+00
 1.07242659e+00  1.07412998e+00  1.07453284e+00  1.07514340e+00
 1.07554058e+00  1.07696311e+00  1.07767367e+00  1.07821995e+00
 1.08019425e+00  1.08054105e+00  1.08161381e+00  1.08312387e+00
 1.08375073e+00  1.08409038e+00  1.08556125e+00  1.08664060e+00
 1.08761192e+00  1.08912356e+00  1.08937828e+00  1.09082270e+00
 1.09233753e+00  1.09283179e+00  1.09383110e+00  1.09502089e+00
 1.09630653e+00  1.09827775e+00  1.09876234e+00  1.09958891e+00
 1.10096741e+00  1.10140510e+00  1.10351237e+00  1.10384381e+00
 1.10602352e+00  1.10798429e+00  1.10845139e+00  1.10954290e+00
 1.11080279e+00  1.11159317e+00  1.11306899e+00  1.11441063e+00
 1.11561448e+00  1.11638074e+00  1.11844457e+00  1.11906802e+00
 1.12265873e+00  1.12388268e+00  1.12517735e+00  1.12569225e+00
 1.12793035e+00  1.13018630e+00  1.13179435e+00  1.13239995e+00
 1.13358140e+00  1.13734581e+00  1.13883209e+00  1.14147333e+00
 1.14285714e+00  1.14285714e+00  1.14387864e+00  1.14516077e+00
 1.14874599e+00  1.15097199e+00  1.15137460e+00  1.15407050e+00
 1.16007708e+00  1.16168594e+00  1.16181259e+00  1.16613952e+00
 1.17101907e+00  1.17472079e+00  1.17527485e+00  1.19182376e+00
 1.19796968e+00  1.20891304e+00  1.23304226e+00  1.27349456e+00
 1.28324671e+00  1.39234807e+00  1.41442916e+00  1.57937217e+00] [[ 3.73659
565e-02 -8.21492875e-02  9.47994044e-04 ... -5.44469096e-03
   5.32314013e-03 -1.17415024e-04]
 [ 6.55138721e-02  3.34978246e-02 -1.72780496e-02 ...  3.81467899e-05
   1.39720393e-03  1.95839484e-03]
 [ 5.79763540e-02  3.25034504e-02  7.28978656e-02 ... -1.37581831e-04
   9.68810202e-04 -1.28229038e-04]
 ...
 [ 5.84206238e-02  2.87958439e-02 -8.34034886e-02 ... -6.56493088e-04
  -8.72425941e-05 -5.93146577e-05]
 [ 5.47656465e-02  3.06424242e-02  7.66912615e-02 ... -2.46122914e-04
   1.28299897e-03 -1.21988865e-04]
 [ 1.30037346e-01 -2.74479658e-02  8.36567360e-04 ...  2.21492906e-02
  -1.04092103e-01  7.72424456e-03]]
```

(c) We would like to cluster the nodes (i.e. the users) in 3 groups. Using the eigenvectors of $\tilde{L}$, assign to each node a point in $\mathbb{R}^2$, exactly as explained in last lecture (also in 'Algorithm 1' of the notes) where you replace L by $\tilde{L}$. Plot these points using the 'scatter' function of matplotlib.

In [32]:
```
# Your answer here
# k = 3, we only need the first three eigenvectors
eigenvectors_3 = eigenvectors[:, 1:3]
points = eigenvectors_3
plt.scatter(points[:,0],points[:,1])
```

Out[32]:
```
<matplotlib.collections.PathCollection at 0x17aed9587c0>
```

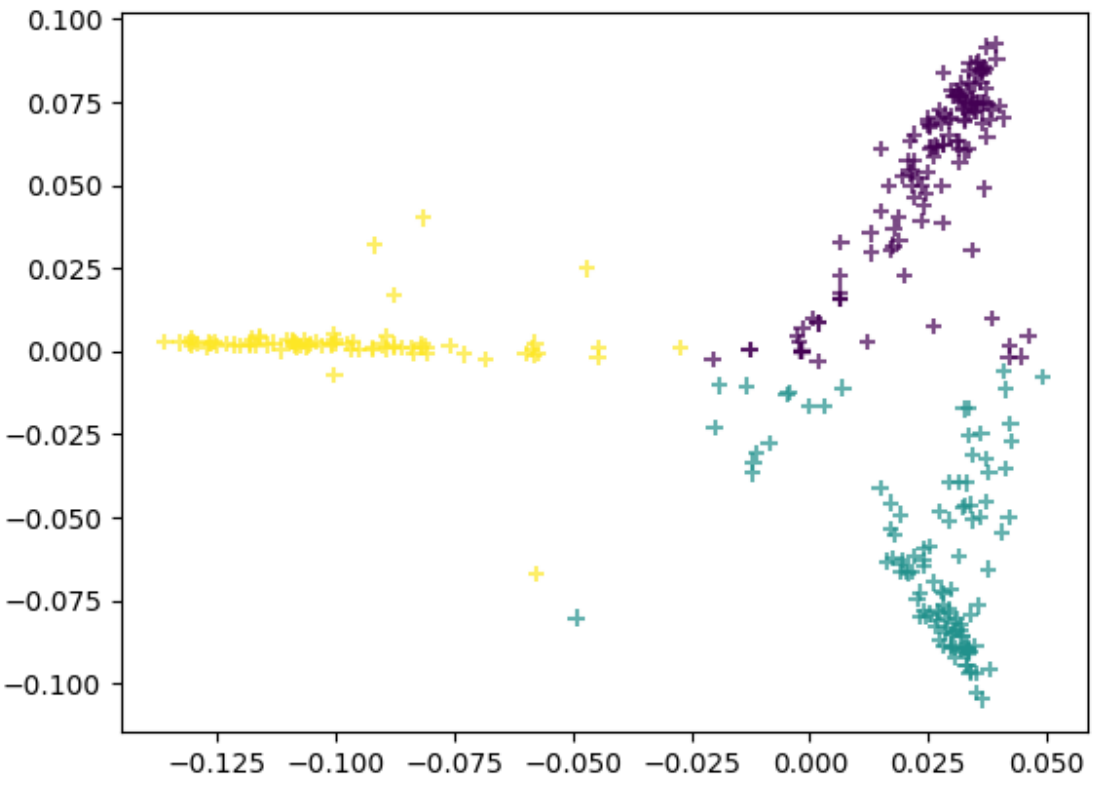


(d) Using the K-means algorithm (use the built-in function from scikit-learn), cluster the embeddings in $\mathbb{R}^2$ of the nodes in 3 groups.

In [41]:
```
# Replace ??? by the matrix of the points computed in (c)
# Each row corresponds to a data point
kmeans = KMeans(n_clusters=3, random_state=0).fit(points)
labels=kmeans.labels_
# labels contains the membership of each node 0,1 or 2
print(labels)
# This colors each point of R^2 according to its label
# replace "x/y coordinates" by the coordinates you computed in (c)
plt.scatter( points[:,0], points[:,1], alpha=0.7, marker='+', c = labels)
```

```
D:\Program_Files\Python\Python310\Lib\site-packages\sklearn\cluster\_kmeans.
py:1412: 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
  super()._check_params_vs_input(X, default_n_init=10)
[2 1 0 0 1 1 2 0 0 0 0 1 1 1 2 1 2 2 1 0 0 2 0 2 0 2 1 2 1 1 0 1 0 1 1 1 1
 1 0 0 0 0 0 0 1 1 1 2 1 2 0 2 0 0 0 0 1 0 1 1 1 0 1 0 2 0 2 0 2 0 2 0 0 0 1 0
 1 1 1 0 2 0 0 1 0 0 2 0 2 2 2 2 1 2 0 0 0 1 2 2 0 2 0 1 1 1 1 1 2 1 1 1 0
 2 0 1 2 0 1 0 2 0 2 2 2 0 2 2 2 0 0 2 1 2 1 1 2 2 1 0 1 0 1 1 1 2 0 2 0 0
 2 1 2 2 2 2 1 2 2 1 0 0 0 0 1 0 0 0 1 2 2 0 0 2 0 0 2 0 1 1 0 2 0 0 0 0 0
 0 1 0 2 1 1 0 0 2 1 1 1 1 0 1 1 1 0 0 0 0 0 2 1 2 2 2 1 0 1 1 1 1 1 1 0 1
 1 0 1 2 2 1 1 0 0 1 2 1 0 1 0 1 1 0 0 1 0 2 1 0 0 0 1 1 0 2 2 2 0 0 0 0 1
 2 2 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 2 2 2 1 0 1 0 2 1 0 0 1 0 1
 0 1 1 2 1 2 0 0 2 0 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 0 1 2 0 1 0 2]
```

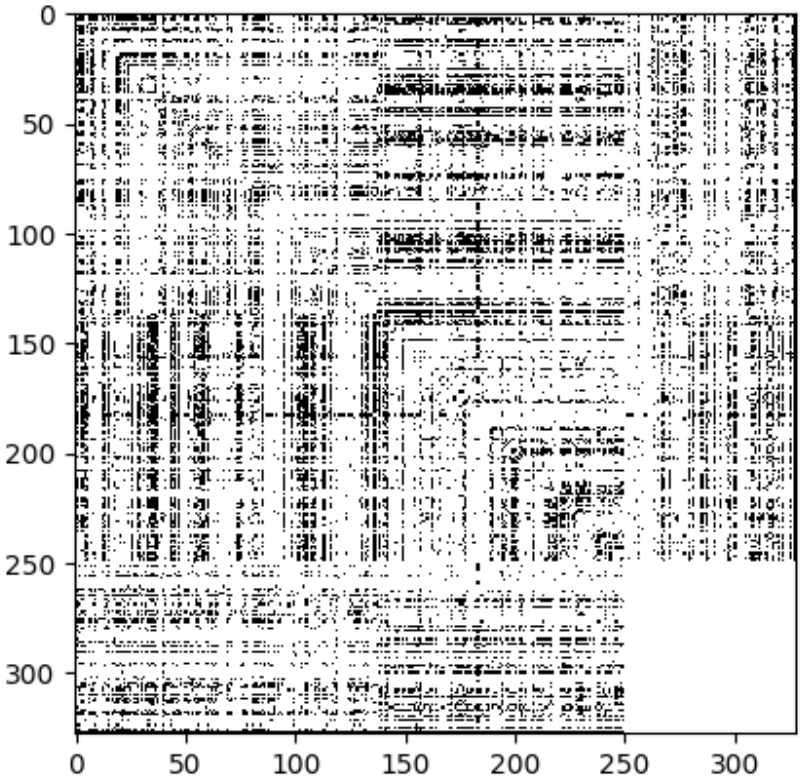
Out[41]: `<matplotlib.collections.PathCollection at 0x17aedee8e80>`



(e) Re-order the adjacency matrix according to the clusters computed in the previous question. That is, reorder the columns and rows of A to obtain a new adjacency matrix (that represents of course the same graph) such that the n_1 nodes of the first cluster correspond to the first n_1 rows/columns, the n_2 nodes of the second cluster correspond to the next n_2 rows/columns, and the n_3 nodes of the third cluster correspond to the last n_3 rows/columns. Plot the reordered adjacency matrix using 'imshow'.

In [85]:
```
## Your answer here
import pandas as pd
from functools import reduce
point_label_df = pd.DataFrame(np.hstack((points,labels.reshape(-1,1))))
concatenated = reduce(lambda accu, curr: accu + curr,[list(point_label_df[pc
reordered_adj = np.zeros_like(A)

counter = 0
for i in concatenated:
    reordered_adj[counter,:] = A[i,:]
    reordered_adj[:,counter] = A[:,i]
    counter += 1

plt.imshow(reordered_adj, aspect='equal',cmap='Greys',  interpolation='none'
```

Out[85]:
```
<matplotlib.image.AxesImage at 0x17af2291810>
```

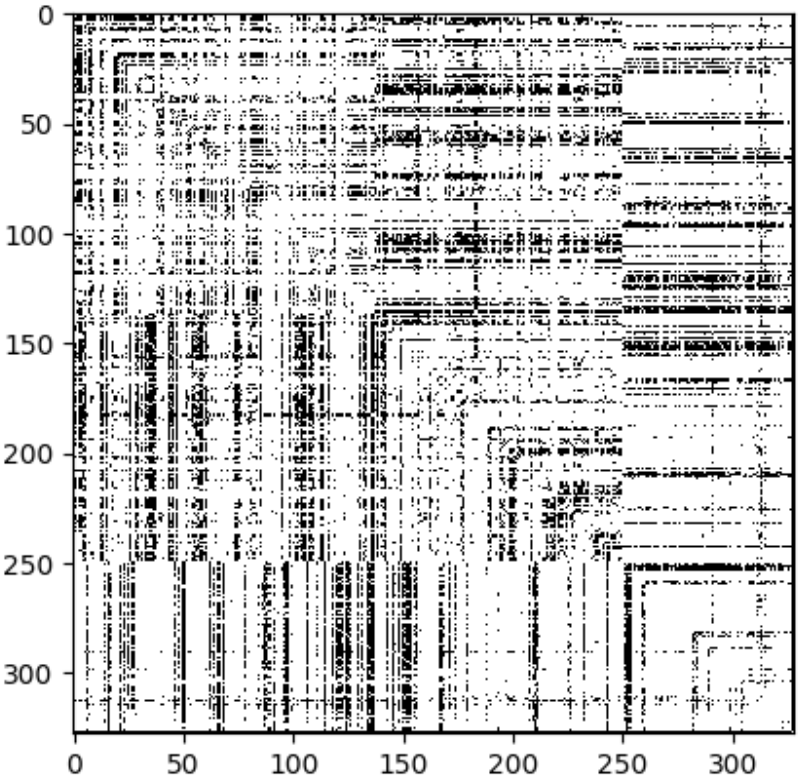


In [110…

```python
# Full Algorithm Packages
class Spectral_Graph_Clustering_Algorithm():

    def __init__(self, filename):
        self.filepath = filename
        self.A = self.load_data(filename)


    def load_data(self, filepath):
        A = np.loadtxt(filepath)
        return A

    def compute_L_A_D(self, A, normalized = True):
        D = np.diag(A.sum(axis = 1))
        L = D - A
        D_sqrt = np.diag(np.power(A.sum(axis = 1),- 0.5))
        L_norm = D_sqrt @ L @ D_sqrt
        L_norm
        return L_norm if normalized else L

    def compute_data_for_KMeans(self,L,k):
        eigenvalues, eigenvectors = np.linalg.eigh(L)

        data_matrix = eigenvectors[:, 1:k]

        self.eigenmatrix = data_matrix

        return data_matrix
```

```
    def perform_KMeans(self,data, k):
        import pandas as pd
        from functools import reduce
        kmeans = KMeans(n_clusters=k, random_state=0).fit(data)
        labels = kmeans.labels_

        return labels


    def reorder_adj_matrix(self,A,k):
        point_label_df = pd.DataFrame(np.hstack((points,labels.reshape(-1,1)
        concatenated = reduce(lambda accu, curr: accu + curr,[list(point_lab
        reordered_adj = np.zeros_like(A)
        counter = 0
        for i in concatenated:
            reordered_adj[counter,:] = A[i,:]
            reordered_adj[:,counter] = A[:,i]
            counter += 1

        return reordered_adj

    def fit(self, k, normalized = True):
        A = self.load_data(self.filepath)
        L = self.compute_L_A_D(A)
        data_matrix = self.compute_data_for_KMeans(L, k)
        labels = self.perform_KMeans(data_matrix, k)
        reordered_adj = self.reorder_adj_matrix(A,k)
        self.reordered_adj = reordered_adj
        self.L = L
        return reordered_adj


    def plot_adj_matrix(self,):
        plt.imshow(self.reordered_adj, aspect='equal',cmap='Greys',  interpo

    def plot_clusters(self,):
        plt.scatter(self.eigenmatrix[:,0],self.eigenmatrix[:,1])
        plt.show()



adj_filepath = "./adjacency.txt"
SGCA = Spectral_Graph_Clustering_Algorithm(adj_filepath)


SGCA.load_data(adj_filepath)
reordered_adj = SGCA.fit(k = 6)
SGCA.plot_adj_matrix()
# SGCA.plot_clusters()
plt.show()
```

```
D:\Program_Files\Python\Python310\Lib\site-packages\sklearn\cluster\_kmeans.
py:1412: 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
  super()._check_params_vs_input(X, default_n_init=10)
```



In [ ]:

Problem 8.5 ($\star$). *Let G be a connected graph with n nodes. Define $L \in \mathbb{R}^{n\times n}$ the associate Laplacian matrix and $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ its eigenvalues. Let G' be a graph constructed from G by simply adding an edge. Similarly denote by λ_2' its second smallest eigenvalue. Show that $\lambda_2' \ge \lambda_2$.*

We first prove some useful lemmas.

Lemma 8.1

Lemma 0: Suppose we have two subspaces $U, V \subseteq R^n$. then $\dim(U+V) \leqslant \dim(R^n) = n$ Proof: $\forall \vec{x} \in U+V$, we have $\vec{x} = \vec{u}_x + \vec{v}_x$, where $\vec{u}_x \in U, \vec{v_x} \in V$

$$\begin{aligned}
&\because U \subseteq R^n, V \subseteq R^n \\
&\therefore \vec{u}_x \in R^n, \vec{V}_x \in R^n \\
&\therefore \vec{u}_x + \vec{v}_x = \vec{x} \in R^n \\
&\therefore U+V \subseteq R^n \\
&\therefore \dim(U+V) \le \dim(R^n) = n
\end{aligned}$$

Lemma 8.2

Lemma 1: For $A_{n\times n} \in S^n$, suppose we have a subspace $V \subseteq R^n$ and $\dim(v) = k$ then $\exists \vec{x}, \vec{y} \in V, \|\vec{x}\|_2 = \|\vec{y}\|_2 = 1$, sit $\vec{x}^\top A\vec{x} \leqslant \lambda_k(A)$ and $\vec{y}^\top A\vec{y} \geqslant \lambda_{n-1+1}(A)$ Proof: Since $A \in S_+^n$, by spectrum theorem we have: $A = UDU^\top$ Now we construct a subspace Q where Q is spanned by $U_k = \begin{bmatrix} \vec{u}_k & \vec{u}_{k+1} & \cdots & \vec{u}_n \end{bmatrix}$, the last $n-k+1$ eigenvectors of A. Thus $\dim(Q) = \dim(\mathrm{col}(U_k)) = n-k+1$.

$$\begin{aligned}
&\because \dim(Q+V) = \dim(Q) + \dim(V) - \dim(Q\cap V) \\
&= n-k+1+k-\dim(Q\cap V) \\
&\leqslant \dim(R^n) \\
&= n \\
&\therefore \dim(Q\cap V) \geqslant 1 \\
&\therefore \exists \vec{x} \in Q\cap V \text{ and } \vec{x} \neq \vec{0}.
\end{aligned}$$

Now we choose $\vec{x} = U_k\vec{\varepsilon}$ where $\|\vec{\varepsilon}\|_2 = 1$, then $\|\vec{x}\|_2 = 1$, we have:

$$\begin{aligned}
\vec{x}^\top A\vec{x} &= \vec{\varepsilon}^\top U_k^\top UDU^\top U_k\vec{\varepsilon} \\
&= \sum_{i=k}^{n} \lambda_i \varepsilon_i^2 \\
&\le \lambda_k(A) \sum_{i=k}^{n} \varepsilon_i^2 \\
&= \lambda_k(A)\|\vec{\varepsilon}\|_2 \\
&= \lambda_k(A)
\end{aligned}$$

The second inequality can be obtained by applying the first inequality on $(-A)$.

Lemma 8.3

Lemma 2: Let $A \in S^n$, and let V denote a subspace of R^n. Then for $k = 1, 2, \cdots, n$ it holds that:

$$\begin{aligned}\lambda_k(A) &= \max_{\dim V=k} \min_{\vec{x}\in V_1 \|\vec{x}\|_2=1} \vec{x}^\top A\vec{x} \\ &= \min_{\dim V=n-k+1} \max_{\vec{x}\in V, \|\vec{x}\|_2=1} \vec{x}^\top A\vec{x}\end{aligned}$$

Proof: Let $\mu_k(A) = \max_{\dim V=k} \min_{\vec{x}\in V_1 \|\vec{x}\|_2=1} \vec{x}^\top A\vec{x}$, then:

$$\mu_k(A) \leqslant \max_{\dim V=k} \lambda_k(A) \text{ (by Lemma 8.2)}$$

$$= \lambda_k(A) \quad \text{(Since the compile is } V \text{)}$$

If we choose $V_k = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_k \end{bmatrix}$ where $\vec{u}_1 \cdots \vec{u}_k's$ are the first k-eigenvectors of A, then, letting $\vec{x} = V_k\vec{y}$, $\|\vec{y}\|_2 = 1$

$$\begin{aligned}\mu_k(A) &\geqslant \min_{\vec{x}\in V_k, \|\vec{x}\|_k=1} \vec{x}^\top A\vec{x} \\ &= \min_{\|\vec{y}\|_2=1} \sum_{i=1}^{k} \lambda_i y_i^2 \\ &= \lambda_k(A) \sum_{i=1}^{k} y_i^2 \\ &= \lambda_k(A)\end{aligned}$$

$\therefore \lambda_k(A) = \max_{\dim V=k} \min_{\vec{x}\in V, \|\vec{x}\|_2=1} \vec{x}^\top A\vec{x}$, where

$$\operatorname*{argmax}_{\dim V=k} \min_{\vec{x}\in V, \|\vec{x}\|_2=1} \vec{x}^\top A\vec{x} = V_k.$$

Proof. Finally we could do the proof for this problem. Suppose the Laplacian matrix for G is L and the Laplcian matrix for G' is L'. We know that $L, L' \in \mathbb{S}_n^+$, then $\Delta L = L' - L \in \mathbb{S}_n^+$, then by applying the lemma 8.3 we have $\lambda_2(L + \Delta L) \geq \lambda_2(L) + \lambda_{min}(\Delta L)$, which implies that $\lambda_2' \geq \lambda_2$. □