TEL AVIV UNIVERSITY

Mathematical Methods

For

Data Science and Signal Processing

2022-2023

Assignment 2

Or Trabelsi

ID1: 301613501

Question 1

Given a square $n \times n$ an asymmetric matrix M, we can build the symmetric matrix A by the following relation:

$$A_{ij} = A_{ji} = \frac{M_{ij} + M_{ij}^T}{2} \; for \; i,j = 0,1,\dots,n$$

Or in equivalent matrix form

$$A = \frac{M \cdot M^T}{2}$$

Let us estimate the leading eigenvector $\vec{u}$ of matrix A by the ***Power Iteration*** algorithm

$$\vec{u}_t = \frac{\hat{u}_t}{\|\hat{u}_t\|} = \frac{A \cdot \vec{u}_{t-1}}{\|A \cdot \vec{u}_{t-1}\|}, \qquad for \; iterations \; t = 1, \dots, T$$

While the initial $\vec{u}_0$ vector will be randomized generated.

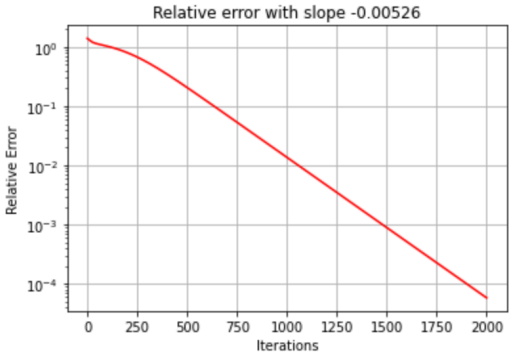
In each iteration we check the relative error between the estimated eigenvector and the calculated one $\vec{u}_{ref}$ (using ***scipy*** library), by the given formula:

$$relative \; error = \min_{z \in \{\pm 1\}} \frac{\left\| z \cdot \vec{u}_t - \vec{u}_{ref} \right\|}{\left\| \vec{u}_{ref} \right\|}$$

In the following we can see the overall results while after a few hundreds of iterations, the relative error became with decreasing linear behavior. This linear slope is not surprise, since we saw in class that finally the iterative will converge to a geometric line with ratio $\left|\frac{\lambda_1}{\lambda_2}\right|$, where λ_1 and λ_2 are, respectively, the leading and second eigenvalues of the matrix A.

Calculating the slope (with log), we get:

```
Expected slope: -0.0024
Actual slope: -0.0053
```



The reason to take the minimum between the positive and negative versions of the relative error is that the eigenvalues of a matrix can be either positive or negative. This means that the true dominant eigenvalue could be either positive or negative, and the current estimate of the dominant eigenvalue could also be either positive or negative.

Question 2

First of all, we generated the square wave signal

$$x[m] = \begin{cases} 1, & 0 \leq m < 10 \\ 0, & 40 \leq m < 50 \end{cases}$$

Then, we calculated the $n = 2000$ observations

$$y_i = R_{l_i} x + \epsilon_i, \quad i = 1, \dots, n$$

Such that,

$$X_{m\times 1} = \begin{bmatrix} x_0 \\ \vdots \\ x_m \end{bmatrix}, \qquad Y_{m\times n} = \begin{bmatrix} y_{1,0} & \cdots & y_{n,0} \\ \vdots & \ddots & \vdots \\ y_{1,m} & \cdots & y_{n,m} \end{bmatrix} = [y_1 \quad \dots \quad y_n]$$

The Diffusion Mapping implementation was done according to the lecture slides.
For each two observation vectors y_i and y_j, we calculated the distance by the Gaussian kernel.

$$W(i,j) = w_{i,j} = w_{ji} = k(y_i, y_j) = \exp\left(-\frac{|y_i - y_j|}{\tau_g}\right)$$

Then we calculate the diagonal matrices D, D^{-1} to compute the random walker probability matrix M.

$$\deg(i) = \sum_j w_{i,j}\,, \qquad for\ i = 1, \dots, n$$

$$D = diag(deg) \qquad and \qquad D^{-1} = diag(\deg^{-1})$$

s.t

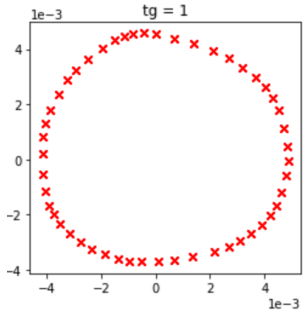
$$M = D^{-1} \cdot W$$

Now we can construct the S matrix

$$S = D^{1/2} \cdot W \cdot D^{-1/2}$$

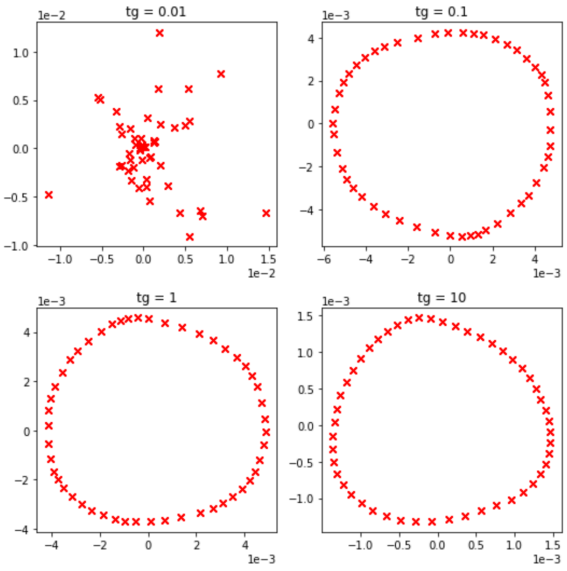
Now, we can use the ***scipy*** function from the previous section to calculate the d eigenvectors that corresponded to the largest eigenvalues, while $d = 2$ is the dimensions of the diffusion mapping we want.
Now we calculate the Φ matrix that contains the ϕ_k right eigenvectors of the matrix M.
By taking the Φ rows we retrieved the following diffusion maps.

First, I used $\tau_g = 1$ and after the



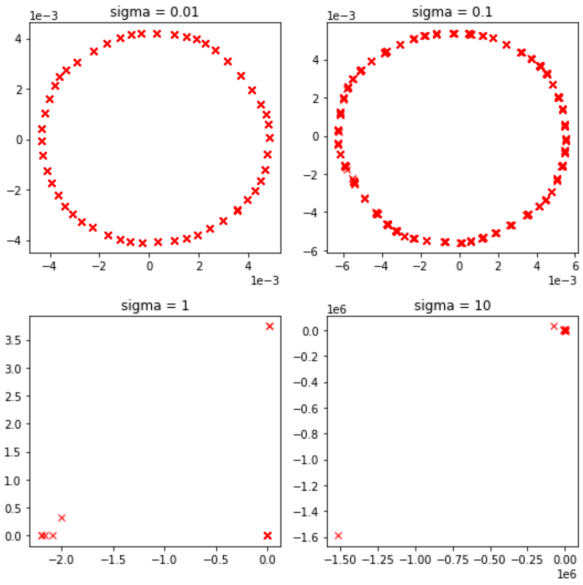
The diffusion mapping in this section took many observations that represented by a circular shifting of a given signal. Since we used the diffusion mapping algorithm to represent the high dimensional data in a 2D space, it makes sense that the shape will be some circular structure.

Then, I changed the distance scaling factor τ_g values in the kernel calculation in order to check the guided question.



From this, it can be seen that while the scale is too low, it is actually expanding the distance between the observations and made the graph sparsely with non-circular structure.

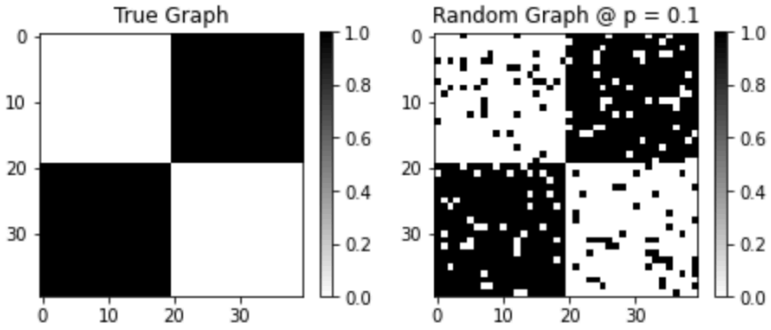
Now, let us remain with the initial $\tau_g = 1$ and add



We can see here, that noise with small variance doesn't affect the diffusion mapping results, but while the noise became "heavy" the structure of the mapping is broken.

Question 3

First, let us generate the required basics graphs.



Now, we generated the random graphs depends the probability $p = [0.1, 0.5]$ and cluster the vertices into the two sets from the "true graph" by finding the maximal cut.

Since max-cut is a Hard-NP problem, we will use the Semi-Definite Programming relaxation (SDP) of our max cut problem by using the convex solver CVXPY while our problem is defined by

$$\max \frac{1}{2} \sum_{i<j} w_{i,j}(1 - X_{i,j})$$

Where $w_{i,j}$ is the edges weights and X is the ***Gram*** matrix $X = U^T U$ and the column of U is u_i

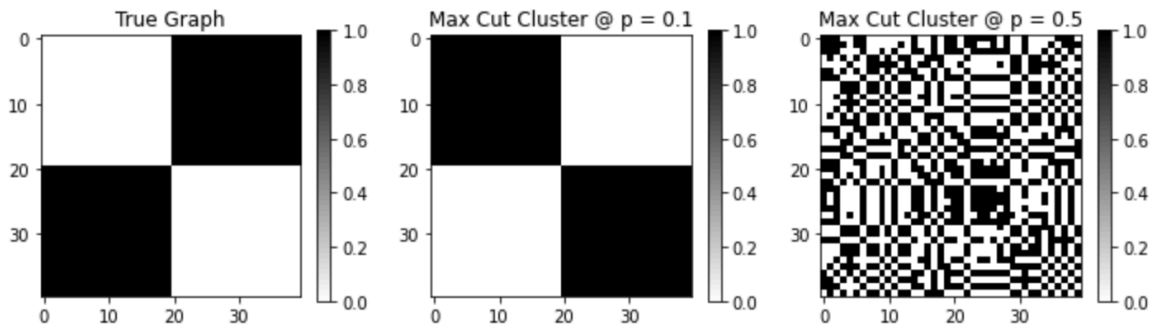
With the following constrains:

1. X is PSD matrix such that $X \gg 0$
2. The $U's$ columns norm will be equal to 1, which means $\|u_i\| = 1 \; s.t. \; X_{i,i} = 1$

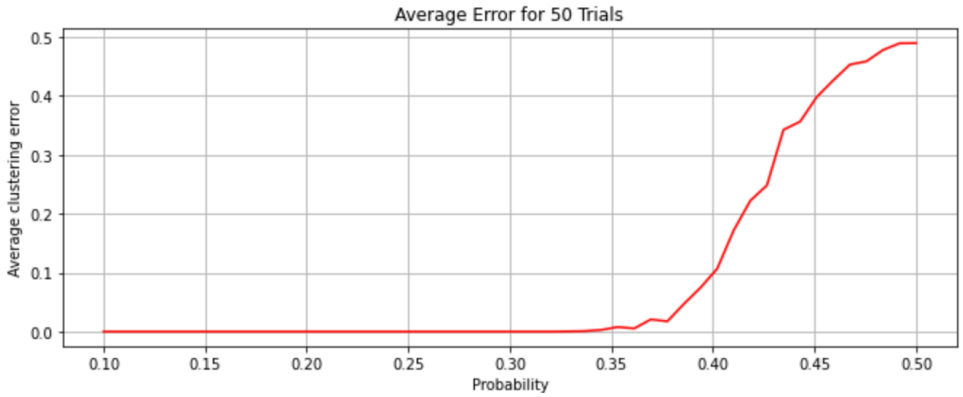
As we required, we did the experiment 50 times for different graphs with different probabilities as mentioned above.

Finally, even though, we are talking about a hard-NP problem, after a few seconds we got solution to our clustering problem.

In the following image, we can see comparison of the best and worst solutions compared to the ground truth graph.



The graph below shows how the PSD relaxation achieved perfect results for graphs that generated with probability lower the 0.3. Notice that the graph was calculated as defined in the home assignment instructions – the number of misclassified vertices divided by the total number of vertices.



Question 4

Let us generate the $n = 100$ random angles $\theta := (\theta_1, \ldots, \theta_n)$ and the rotation matrix $H = hh^*$, where $h = \left(e^{i\theta_1}, \ldots, e^{i\theta_n}\right)^T$. Then, we created the corrupted matrix $H_{corrupt}$ with the required "outliers model".

First, we implemented the spectral relaxation model in order to estimate the rotation angles. Since we know that $e^{j\hat{\theta}_i} = \frac{\hat{h}(i)}{|\hat{h}(i)|}$ for $i = 1, \ldots, n$, we began to calculate the vector $\hat{h} = e^{j\hat{\theta}}$ by calculating the top eigenvector of the matrix $H_{corrupt}$. Then we extract the aligned angles $\hat{\theta}_{al}$ by the following steps.

$$\hat{h}_{al} = \hat{h} * \left(\hat{h}^* \cdot h\right) \qquad s.t. \qquad \hat{\theta}_{al} = \angle(\hat{h}_{al}) \; modulu \; 2\pi$$

The error was calculated as an average error by the following form

Denote,

$$f_{err}(i) = \begin{cases} 0, & \hat{\theta}_{al_i} = \theta_i \\ 1, & otherwise \end{cases}$$

$$err = \frac{1}{n}\sum_{i=1}^{n} f_{err}(i) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} 0, & \angle\left(\hat{h} * \left(\hat{h}^* \cdot h\right)\right)_i \; modulu \; 2\pi = \theta_i \\ 1, & otherwise \end{cases}$$

Notice that we round the results for a few digits in order to prevent numerical errors.

In similar way to question 3, but for $Z = zz^*$ we used here also the SDP method to solve the maximization problem:

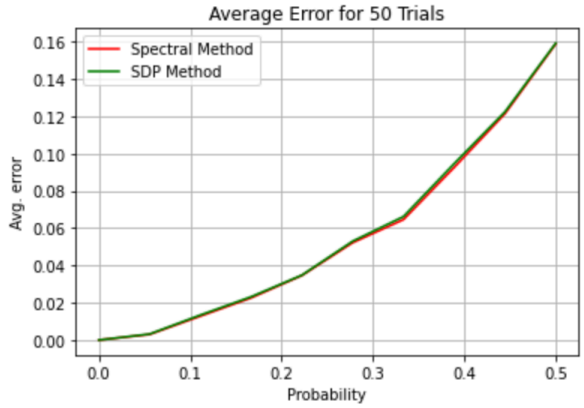
$$\max_{Z} Tr(HZ)$$

with the following constrains:

1. Z is PSD matrix such that $Z \gg 0$
2. $diag(Z) = 1$
3. $Rank(Z) = 1$

From the solution of this problem, we calculated the z vector using the eigen vector of Z.

Now, with the same way we described above, we extract the angles where here the z was taken instead of $\hat{h}$.
Finally, we got the following average error graph as function of the probability p of the "outliers model".



To prove that $e^{i\theta_{al}} = \frac{\hat{h}^* h}{|\hat{h}^* h|}$, we can start by expressing h and $\hat{h}^*$ as defined in the problem

$$h = e^{i\theta}, \qquad \hat{h}^* = e^{-i\hat{\theta}}$$

Substituting these expressions into the equation we want to prove, we get:

$$e^{i\theta_{al}} = \frac{e^{-i\hat{\theta}} e^{i\theta}}{\left|e^{-i\hat{\theta}} e^{i\theta}\right|} = \frac{e^{-i\hat{\theta}} e^{i\theta}}{\left|e^{i(\theta-\hat{\theta})}\right|} = \frac{e^{-i\hat{\theta}} e^{i\theta}}{1} = e^{-i\hat{\theta}} e^{i\theta}$$

Rearrange the equation will give us the following expression:

$$e^{i\theta_{al}} = e^{i(\theta-\hat{\theta})}$$

If our estimated is defined up to a global angular shift, so we proved that $e^{i\theta_{al}} = \frac{\hat{h}^* h}{|\hat{h}^* h|}$

■