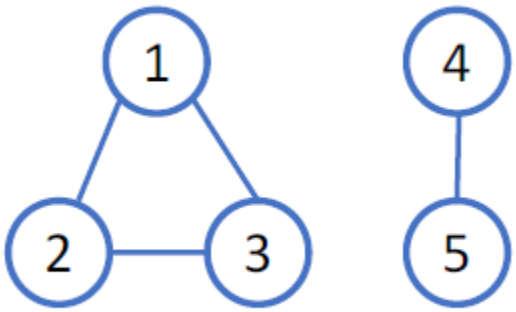
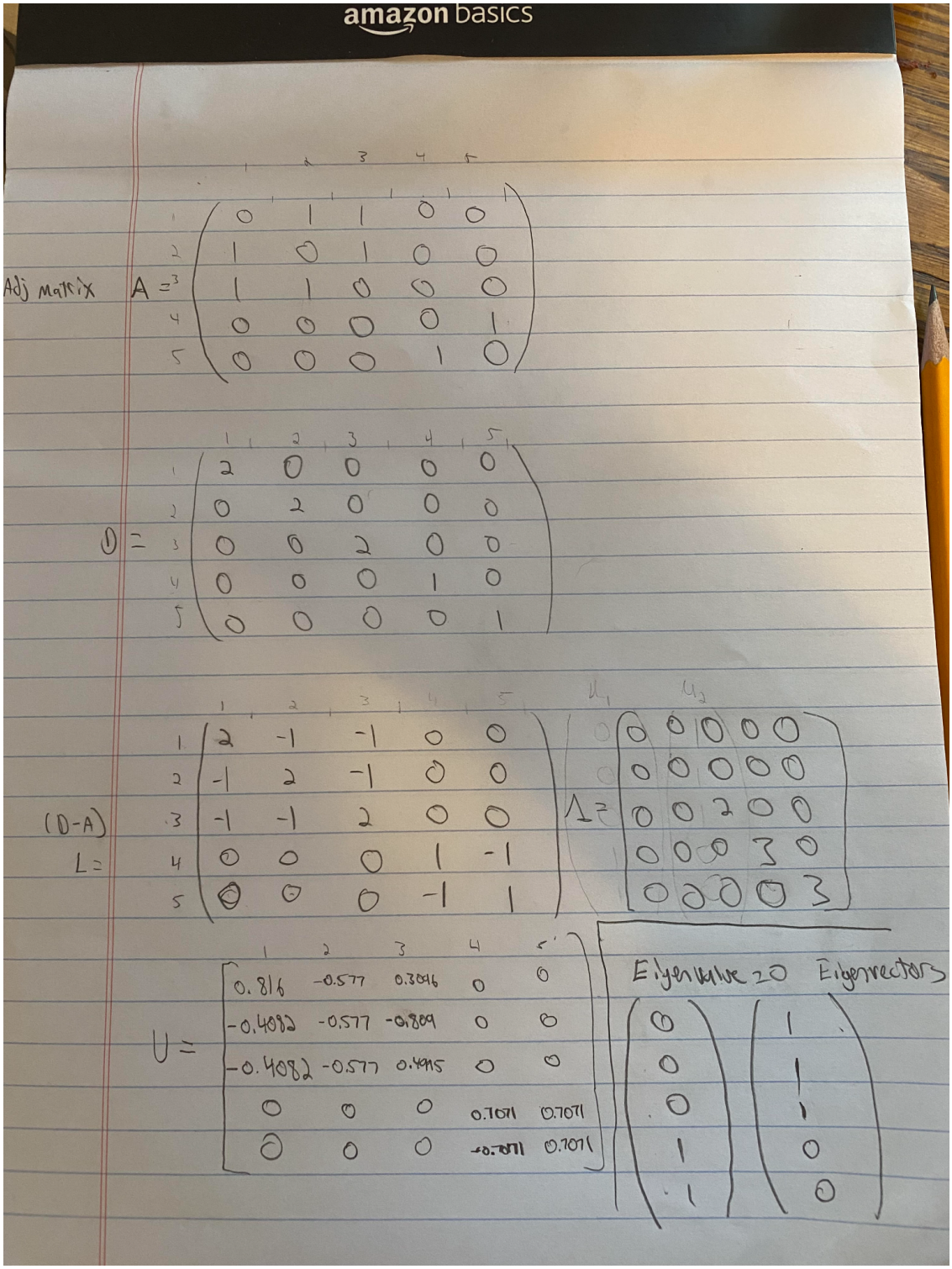
Concept Questions - Part 1

1. Unsupervised learning does not use labeled data, instead it involves algorithms trying to infer the structure of underlying data. Supervised learning requires labeled data for processing. In other words, the predictions are based on training sample(s) of previously solved cases, where the values of the dependent variable are known.
    a. Supervised Learning
        i. Benefit - Since we have labeled data(and a dependent variable) we have easier to interpret(or clear) measures of success. (R squared, AUC)
        ii. Drawback - Requires structured labeled data that is not always available in real-world scenarios
    b. Unsupervised Learning
        i. Benefit - Can be used to solve problems, not possible (or impractical) under supervised learning (image recognition, clustering)
        ii. Drawback - Computationally, unsupervised learning usually requires much more dimensionality or complexity to infer the structure of the input data
2. Yes - different starting points of initialization will lead to different results.K-means is a greedy algorithm, meaning it makes a locally optimal adjustment during iteration. And will not converge to a globally optimal, ‘solved’ solution each time. It is solving an NP(Non-polynomial) hard problem non-convex optimization problem,, that does not have a solved solution.
3. Will K-means converge in a finite number of steps?
    a. Proof
        i. Given Objective Function
            1. $$\frac{1}{m}\sum_{i=1}^{m}\left\|x^{i}-c^{\pi(i)}\right\|^{2}$$
            2. Data points = m
            3. Clusters = c
            4. Number of clusters = k
        ii. The minimum value of the objective function is finite. (0). Because there is a squared term in the objective function.
        iii. Kmeans will not oscillate
            a. The algorithm will only adjust to minimize the objective function.
        iv. Given that the objective is finite, the dataset is finite, and the algorithm always moves in one direction towards the minimum value, we can conclude a finite number of steps in K^M steps that can be taken to perform this algorithm

4. The similarity function is the main difference. K-means uses the l2 norm(Euclidean distance), while the generalized k-means algorithm you can use many different similarity functions (e.g., manhattan distance, Minkowski distance).
    a. Changing the similarity function will induce different geometry. The clusters will contain different points and converge differently based on the similarity function chosen. The similarity function assigns points to clusters based on distance, and different distances will impact this. It will also move the clusters centers based on distance.
5. Graph



    a.
    b. Write down graph laplacian matrix.(see screenshot)
    c. Find eigenvectors associated with the zero eigenvalue. (see screenshot)



d. Explain how do you find out the number of disconnected clusters in graph and identity these disconnected clusters using these eigenvectors
    i. Property II (according to the lectures) states that the eigenvectors contain cluster assignment information. In the attached picture - we can see that

nodes 1-2-3 are connected, and separate from the connected cluster of 4 and 5.

Question 2

Write-up:
Included in my homework submission is a jupyter notebook file named HW1 - Q2. This jupyter notebook contains my source code for this question. Additionally, I have included a .html version of this.

I ran several instances of the k-means algorithm on 3 images, the two provided with the assignment and 1 I provided (The rock band AC/DC's Highway to Hell Album cover). I ran the algorithm for k clusters of 4,6,8,16 and 24 for each image and outputted the results within the notebook.

As expected the larger the clusters, the better the image quality because we are representing more of the original color scale in the compressed image. In a few runs of my code, the original file did look very poor. In these instances, empty clusters were initialized and left as blank, reducing the color scale further for the compressed image.

As expected,, the iterations and time seemed to increase the more clusters were initialized. However, there is still some variation here where more clusters resulted in less iterations, it tends to be dependent on the random initialization of initial clusters as well.

Also, of note, During repeated tests, the Manhattan distance('city block') k-medoid tended to finish slightly faster with equal/comparable compression images. Giving it a clear advantage over the Euclidean distance in calculating distance measurements for assigning points to clusters and recomputing centroids. An example of this is for my 3rd image of AC/DC's album cover. 64 clusters under Manhattan distance, converged in 153 iterations in approx 50 seconds. For 24 iterations under Euclidean distance, 61 seconds were needed.

The one image with the least image compression noticeable to the human eye is image 2. The color scale tends to be continuous with very few sharp contrasts, making it an ideal candidate for this algorithm.

Overall, this exercise illustrated to me how applicable kmeans is to many problems, but also how quickly it can become computationally impractical if the dataset is too large.

Samples of the images are pasted below to show the difference between the smallest clusters and largest cluster count run. Note that the actual clusters may have been less in these runs. In instances where a cluster was assigned no data points, it was dropped to prevent technical glitches. So the actual k-values (esp in higher initializations) is slightly lower.

Image 1 = k =4 and k =24

Image 2 k = 4 and k =24

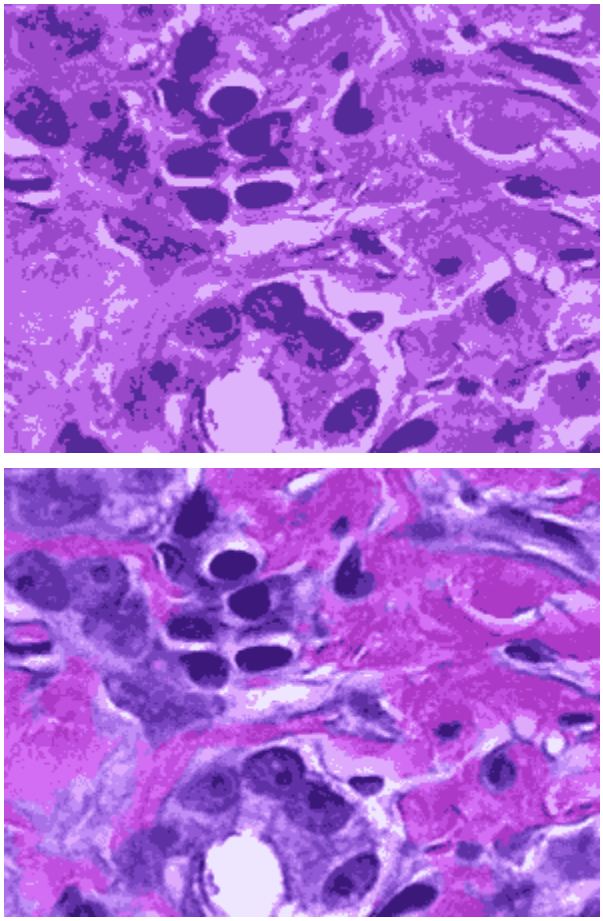


Image 3 k = 4 and k = 64

Question 3

Main source: Demo Code
Write up

Attached with this write up is a jupyter notebook for my spectral graph results for the political blog dataset of nodes and edges. Reported below are the results from running the spectral clustering at 2,5,10,25 clusters. The mismatch rates are prone to some oscillation given the randomness, however the rates seem to hover around 5-6%.

Majority Labels:
2: [1, 0],
5: [0, 1, 0, 1, 1],
10: [1, 1, 1, 0, 0, 0, 0, 1, 0, 0],
25: [1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0]

Mismatch Rates:
2: [0.4795417348608838, 0.0],

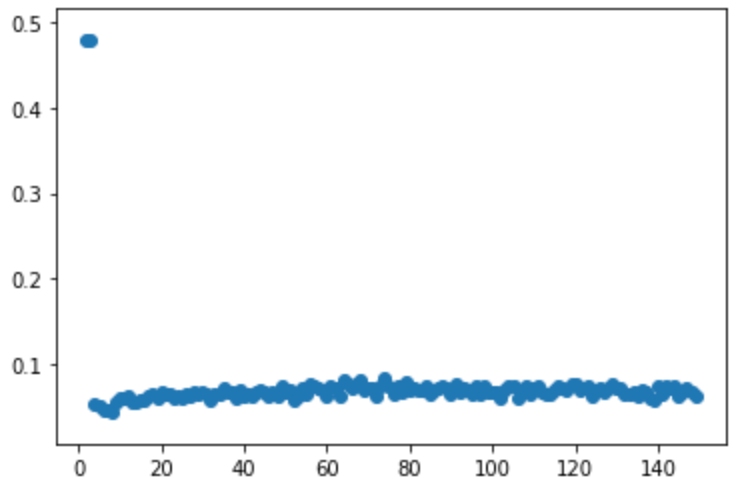
5: [0.023809523809523808, 0.021321961620469083, 0.4342105263157895, 0.3, 0.024691358024691357],

10: [0.02912621359223301, 0.02843601895734597, 0.13043478260869565, 0.4166666666666667, 0.41935483870967744, 0.02197802197802198, 0.03333333333333333, 0.2, 0.019178082191780823, 0.42105263157894735],

25: [0.010869565217391304, 0.03125, 0.02, 0.0, 0.016, 0.32142857142857145, 0.10714285714285714, 0.03773584905660377, 0.03333333333333333, 0.46875, 0.07894736842105263, 0.044444444444444446, 0.05263157894736842, 0.034482758620689655, 0.025974025974025976, 0.027777777777777776, 0.0, 0.23076923076923078, 0.030303030303030304, 0.05555555555555555, 0.0196078431372549, 0.0, 0.25, 0.1643835616438356, 0.22580645161290322]

Total mismatch
[0.47875816993464054, 0.05065359477124183, 0.06045751633986927, 0.06209150326797384]

In terms of fine-tuning the spectral clustering, to achieve a low mismatch rates. I ran the algorithm from k=2 to 150. The results are below



Generally, the intuition and math dictates that a higher clustering will lead to better results, at the expense of possibly overfitting our data. However here, the results tend to oscillate between .1 and .05. It could be that we have many ‘low-degree’ nodes (1 or 2) that are impacting our performance, but were are not seeing major variation here, except at very low numbers, which we would assume. k=2 has an almost 50% mismatch rate.