ISYE 6740 Homework #1 – Richard Moss

Part 1: Concept Questions

1) What's the main difference between supervised and unsupervised learning? Give one benefit and drawback for supervised and unsupervised learning, respectively.

Supervised learning uses labeled data as an input. For supervised learning, the response or class of the data point is known and fed into the algorithm. The response can be used in tandem with its associated features to train and improve the model. An example of this would be regression, where a set of predictors are associated with a response variable, we are interested in predicting or explaining, and a linear or non-linear model is fit to the response in order to minimize some measure of error. One advantage of supervised learning is that the output classes in the case of classification or response in the case of regression are often more accurate given sufficient data is used to train the model, since the correct response is known. This means that the model only needs to fit the predictors to a known outcome, rather than having to make assumptions about the output or interpret the input as with unsupervised learning. One disadvantage with supervised learning a lot of data is required to train the model. This results from the fact that in order for the model to make predictions based on new data points, it must have been exposed to a lot of data in order to properly understand the relationship between the various predictors and their effect on the response. In addition, the model will be biased to some extent towards this training set, so if certain predictors are not considered or external factors affecting the model change, the model will be inaccurate in its prediction of new points not in the original training set, making it somewhat inflexible.

Unsupervised learning uses unlabeled data as an input. Each data point is comprised of a feature set, but the correct class or outcome is not given to the model, and it must derive how the data is correctly grouped or find patterns based only on these feature sets. An example of this is k-means clustering, which attempts to group the data into k groups based on the initialized centroids, and then adjusts the cluster centers iteratively. One advantage of unsupervised learning is that since the data does not need to be labelled, preprocessing is reduced significantly, as unlabelled data is fairly common and requires less cleaning compared to supervised learning where the response must be either given or labelled in the preprocessing stage. One drawback of unsupervised learning is that the output may be less accurate, since the correct number of classes may not be derived by the algorithm, or if the wrong algorithm is used it may cluster the data incorrectly. As an example, if one class exists within another, such is in the ring example in the spectral clustering lecture, k-means can not correctly distinguish between those classes and will cluster the data points in a fashion which we may know is incorrect visually, but the machine deems the best solution to the problem.

2) Will different initializations for k-means lead to different results?

Yes, different initializations for k-means leads to different results. Since the cluster centers are randomly assigned, the starting location of the clusters will determine the outcome of the algorithm. This is because k-means is a non-convex polynomial algorithm, and as a result, there are a number of local minimum values which may be similar to the global minimum but are not the optimal solution. Since k-means will only reassign points if they are closer to a cluster center to which they are not assigned, it may approach

and inevitably converge at one of these local minima, and since no points are reassigned, it would deem this to be the final output. For this reason, when using k-means it may be beneficial to randomly assign a variety of starting locations and run the algorithm many times in order to find the best number of clusters and cluster locations.

3) Give a short proof (can be in words but using correct logic) why k-means algorithm will converge in finite number of iterations.

K-means does not oscillate, but rather it will decrease monotonically. Using the objective function based on the L-2 norm, shown below, the algorithm will only reassign a point to a different cluster if it will improve the objective function. In this function, the goal is to minimize the sum of square distance from each point to the centroid, and a centroid will not be moved (by way of reassigning a data point) if it does not reduce this distance. As a result, we will eventually reach and converge at a local minimum where the algorithm can not improve the objective function, and the process terminates. In addition, since our objective function utilizes the sum of **square** numbers, the sums must always be greater than or equal to zero, this since they can not be negative the algorithm should approach zero and converge in at most k^m steps.

$$\frac{1}{m}\sum_{i=1}^{m}\left\|x^i - c^{\pi(i)}\right\|$$

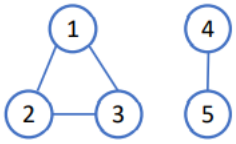
4) What is the main difference between k-means and generalized k-means algorithm? Explain how the choice of the similarity/dissimilarity/distance will impact the result.

The main difference between k-means and generalized k-means is that rather than the objective function shown above, the objective function for generalized k-means will become:

$$\min_{c,\pi}\sum_{i=1}^{m} d\left(x^i, c^{\pi(i)}\right)^2$$

In this equation, we have replaced the L2 distance with a general definition of dissimilarity. This measure of dissimilarity allows us to define in the algorithm what quantifies similarity or dissimilarity between points. As an example, we may choose to use the L2 distance, or it may make more sense to use the Manhattan distance which is the sum of absolute differences between the data points, or even the infinity-distance, which is the maximum distance between the pair of features across two data points. This is key, since how we define similarity affects the output of the algorithm. As an example, if I take all the clothes in my closet, and ask you to group them by similarity, it is entirely subjective as to how you group them. You may choose to group them by color, by season, by brand, or by type. This subjectivity means that the problem is not fully defined, so by defining what it means for two points to be similar/dissimilar, I am changing the method in which I want you to group the clothes, and consequentially, the resultant groups will be different.

5) Consider the following simple graph



Write down the graph Laplacian matrix and find the eigenvectors associated with the zero eigenvalue. Explain how you find out the number of disconnected clusters in graph and identify these disconnected clusters using these eigenvectors.

To derive the Laplacian matrix, we first need to construct the adjacency matrix. Assuming that the graph is undirected and unweighted, this matrix will have 0 if two nodes are not connected, and one if they are, and is shown below.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Next, we construct the degree matrix, which is a diagonal matrix dictating the number of nodes each node is connected to, shown below.

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, the Laplacian Matrix is a positive semi-definite matrix calculated as L = D – A.

$$L = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

I then used Matlab to derive that the first two eigenvalues are 0, and the associated eigenvectors are shown in the code snip below, where columns 1 and 2 are associated with the 0 eigenvalues.

```
>> [V,D] = eig(L)
V =

   0.5774         0         0   -0.7594   -0.3000
   0.5774         0         0    0.1199    0.8076
   0.5774         0         0    0.6395   -0.5077
        0   -0.7071   -0.7071         0         0
        0   -0.7071    0.7071         0         0
```

The number of disconnected clusters can be found by identifying the number of 0 eigenvalues derived from the Laplacian matrix. Since our Laplacian matrix has two 0 eigenvalues, we can conclude it has two disconnected nodes. To identify these nodes, we can bind the eigenvectors of our Laplacian Matrix, and

nodes with the same feature set, or row in the matrix V = [v1 v2 v3 v4 v5] where each v is an eigenvector belong to the same cluster. The V matrix for our Laplacian Matrix is shown above, and it can be seen that rows one to three are the same, and rows four and five are the same, with the eigenvector being constant within the block of connected nodes, and 0 in the other block to which the node is not connected. Thus, nodes 1-3 are shown to be connected to each other, and disconnected from the cluster of nodes 4 and 5.

Part 2: Image Compression Using Clustering

1) Use k-means with squared-ℓ2 norm as a metric, for hestain.bmp and football.bmp and also choose a third picture of your own to work on. We recommend size of 320 × 240 or smaller. Run your k-means implementation with these pictures, with several different k = 2, 4, 8.

The first image compressed using the k-means clustering algorithm was the football image. The algorithm was performed several times with random initial centroids. The images associated with the random centroids which resulted in the best outcome are shown below.

Figure 1. Original Football Image

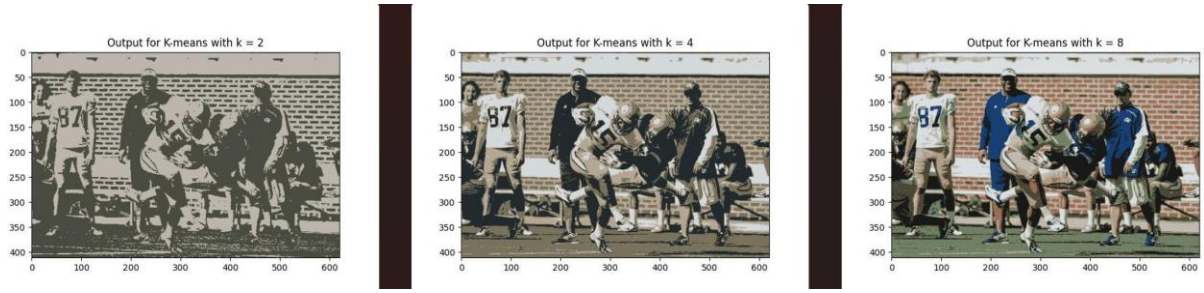


Figure 2. Output for k-means Clustering with Euclidean Distance

The time and iterations to convergence and the centroid values are shown in the table below.

Table 1. Convergence for k-means For Football.bmp

k	Iteration Time	Iterations to Convergence	Centroids		
			R	G	B
2	0.63669	20	74.59407	76.0487	66.44037
			189.9144	182.6483	172.4558
4	1.03501	32	182.2837	166.6998	148.6713
			121.1176	115.9644	93.07748
			35.63657	42.2084	45.50039
			212.766	217.4282	220.334
8	2.05011	47	24.69133	64.33026	117.1546
			75.075	72.71014	55.35965
			24.38335	25.2949	24.27693
			176.9205	157.4748	135.5316
			192.0062	186.3792	177.423
			112.848	129.7504	101.7111
			218.0941	223.2326	226.8381
			157.7524	108.297	84.44534

Interestingly, there were few iterations which resulted in empty clusters for this image, perhaps indicating it had a wide spectrum of colors. Another outcome with this algorithm was that the edges of the shapes tended to be blurred and slightly under-defined, which is significant when we compare to the outcomes of using Manhattan distance in part two. This blurring also becomes significant when cluster centers are initialized poorly resulting in empty clusters. In the image below, two cluster centers were dropped, and

as a result the image is much more monochromatic versus k=8 above and has significant blurring.

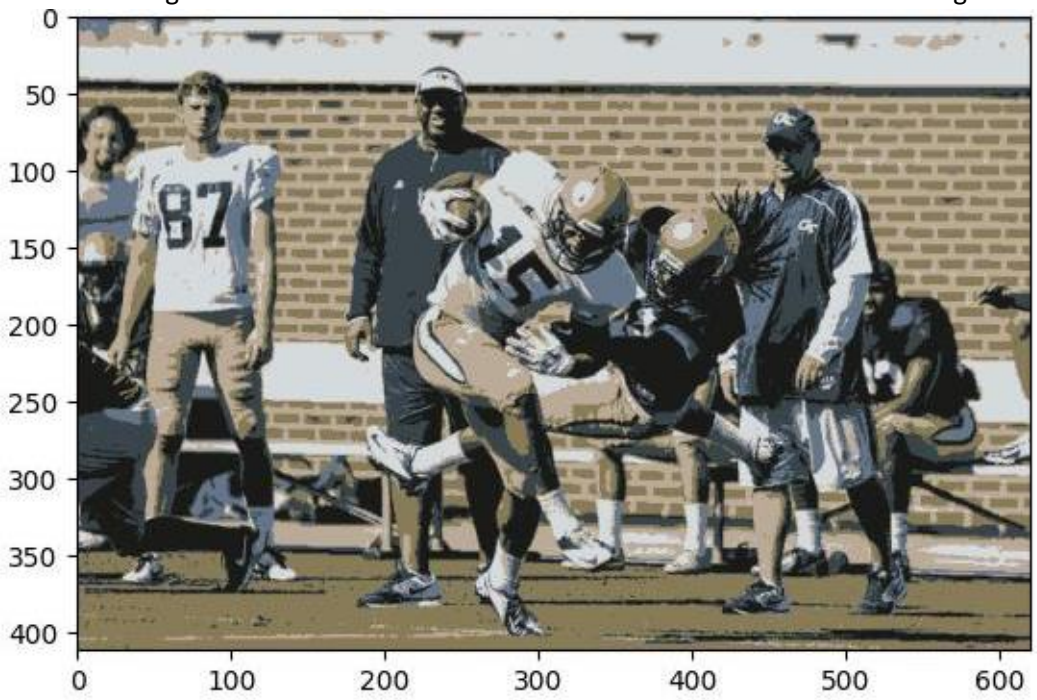


Figure 3. K = 8 decremented to k = 6 due to empty clusters

Next, the outcomes of the algorithm for the hestain image are shown below. Interestingly in the hestain case, cluster centers were almost always dropped, perhaps due to the more monochromatic nature of the image. This indicates that correctly choosing cluster centers for image processing or recognition of hestain samples would be very important as poor centers can result in images which are heavily blurred and lacking definition. Of note, in this best-case scenario, k = 4 was decremented to k =3 and k = 8 was decremented to k = 6 due to empty cluster centers, perhaps indicating with the correct centroids less clusters are necessary for compressing this image.

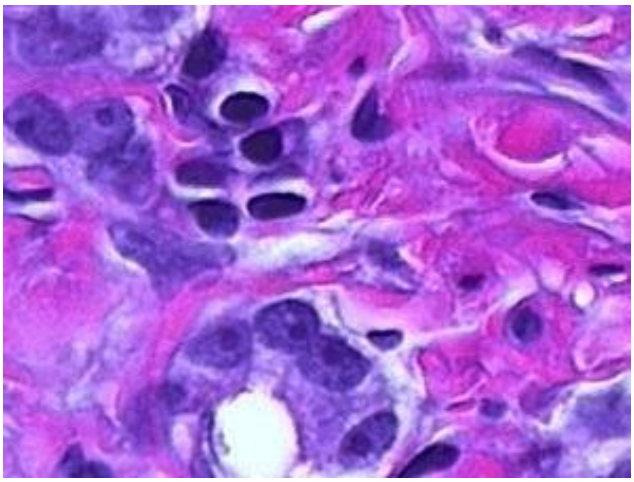


Figure 4. Original hestain.bmp image

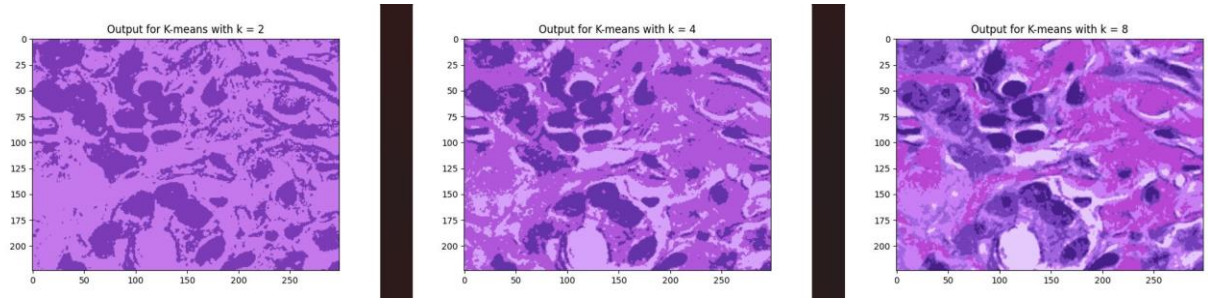


Figure 5. Outcome of k-means clustering for hestain image with Euclidean distance

The convergence values for this scenario are shown in the table below. Of note, convergence took more iterations despite less clusters, perhaps due to the fact that the colors are more similar, resulting in more condensed cluster centers which would cause more reassignment of points as the algorithm converges. This point is further reinforced by the similarity in the RGB centroid values versus the spread in the football image.

Table 2. Convergence based on Euclidean distance for hestain.bmp

k	Iteration Time	Iterations to Convergence	Centroids		
			R	G	B
2	0.204148	24	196.2455	120.338	236.2651
			123.5961	59.80035	182.7479
4	0.4303367	48	175.6085	86.80419	219.8584
			99.27629	51.3107	167.3093
			214.5232	161.908	250.0429
			empty	empty	empty
8	0.5948095	54	70.54263	31.68314	136.4515
			118.8505	63.83286	186.5419
			185.0415	71.22374	213.4936
			202.7688	126.0234	245.3253
			229.2766	201.8445	252.8837
			158.0415	101.2828	222.4093
			empty	empty	empty
			empty	empty	empty

Discussed above the importance of good cluster center initialization was mentioned. This is shown in the image below for k = 2, where the image is entirely lost due to all points being closer to a single cluster center versus the other.

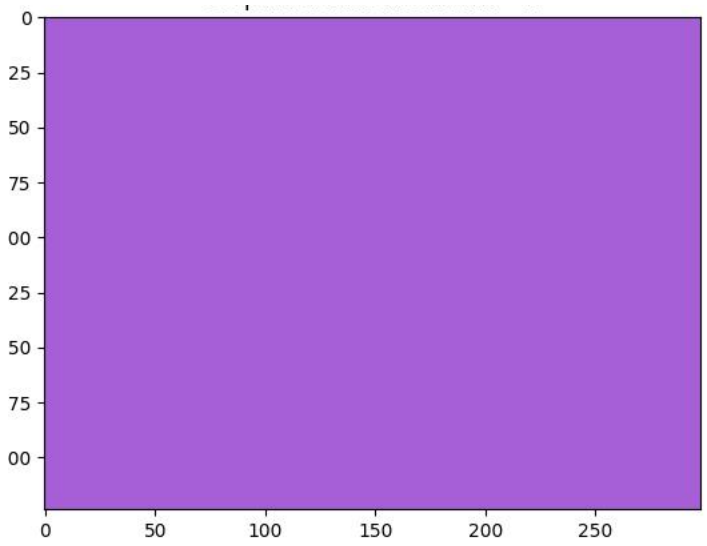


Figure 6. K = 2 decremented to k = 1 due to poor centroid initialization

Finally, in order to test the algorithm further, test subject Ruby was used as input. The original image was chosen due to sharp background colors to see how this impacted the performance of the algorithm with respect to the intended color of the main subject. The best outcome is shown below.

Figure 7. Original Image of ruby.bmp

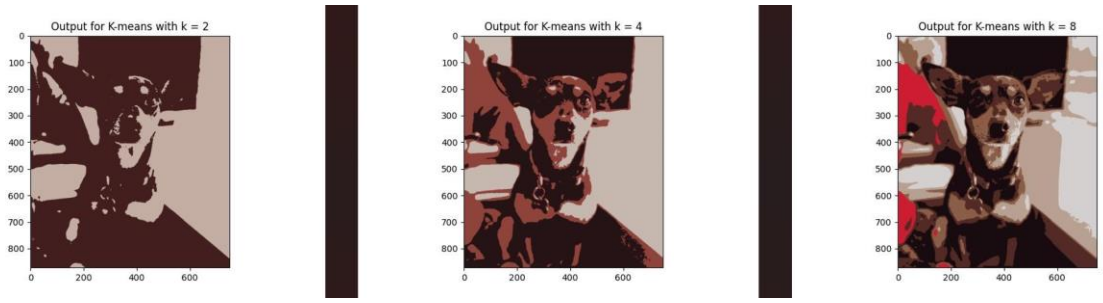


Figure 8. Best Outcome for k-means clustering of ruby.bmp

The convergence values of this iteration are shown below.

Table 3. Best-case convergence values for ruby.bmp

k	Iteration Time	Iterations to Convergence	Centroids		
			R	G	B
2	0.872667	10	196.0486	174.0696	164.4799
			66.38325	30.53946	31.29746
4	2.5138	29	198.166	183.6541	175.7776
			37.79288	19.30293	22.15486
			143.6111	67.30609	59.58227
			empty	empty	empty
8	3.95369	39	139.6073	94.69315	72.91212
			83.16078	42.4605	38.22655
			184.8349	161.8827	146.9171
			24.65582	12.36451	17.2777
			213.857	208.398	208.3282
			206.0895	28.1193	50.35642
			empty	empty	empty
			empty	empty	empty

The examples below show the importance of good centroid initialization in more complex images. In Figure 9, the loss of empty centroids results in the background colors dominating the compression algorithm and the color of the subject is lost. In Figure 10, the cluster centers are fairly monochromatic resulting in a poor image even at k = 8.

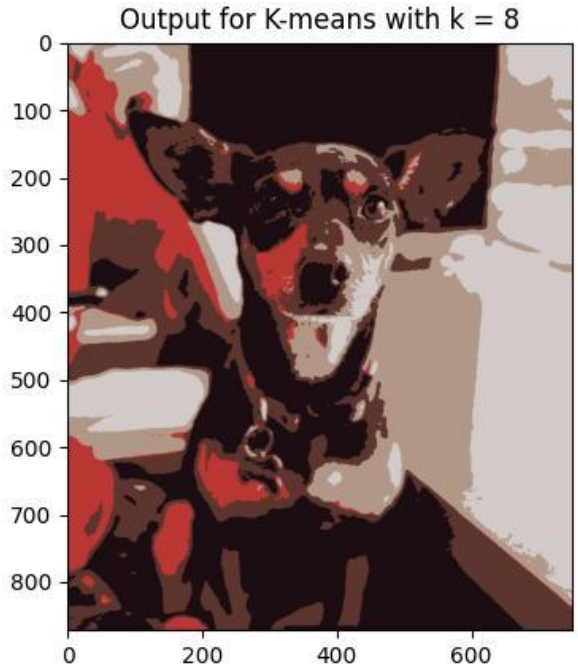


Figure 9. Loss of correct subject color

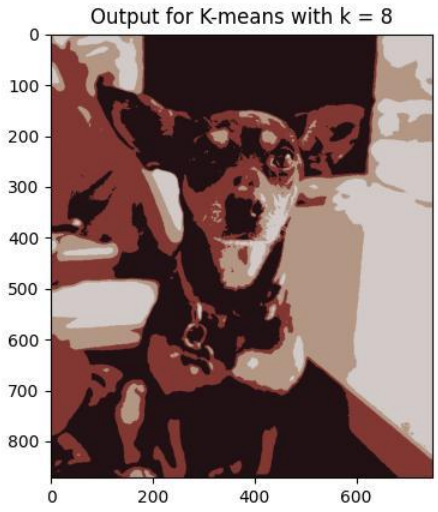


Figure 10. Monochromatic outcome for k = 8 as a result of 4 empty clusters

2) Now try your k-means with the Manhattan distance (or ℓ1 distance) and repeat the same steps in Part (1). Please note that the assignment of data point should be based on the Manhattan distance, and the cluster centroid (by minimizing the sum of deviance – as a result of using the Manhattan distance) will be taken as the "median" of each cluster. Comment on the difference of image compression results using the two methods.

In this section the best outcomes for the Manhattan distance k-means algorithm are shown. Of note, even when cluster centers were dropped, the images were less monochromatic as compared to the use of Euclidean distance. In addition, there was less blur and overlap between clusters, with edges being more clearly defined. First, the best outcome for the football image is shown below.

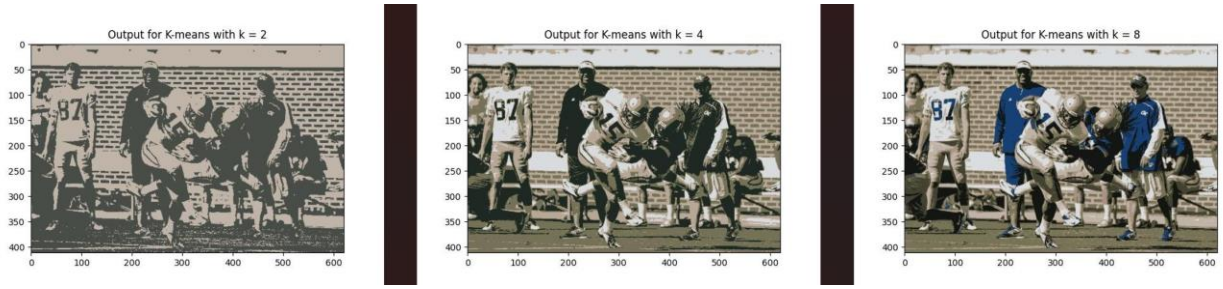


Figure 11. Outcome of k-means clustering based on Manhattan distance for football.bmp

As explained above, we can see that the edges are much sharper, and the image is well defined. The convergence values for this outcome are shown in table. We can see that the cluster centers are always integers, which is expected since they are chosen as the median of the cluster rather than the mean, and the algorithm for this image appears to converge much faster.

Table 4. Convergence values for football.bmp based on Manhattan distance

k	Iteration Time	Iterations to Convergence	Centroids		
			R	G	B
2	0.23214	7	192	181	168
			73	77	71
4	0.28965	8	217	220	222
			185	167	151
			117	114	87
			31	41	35
8	1.19731688	31	23	64	116
			20	22	20
			219	222	224
			192	180	167
			120	116	87
			171	148	130
			68	67	50
			empty	empty	empty

For the hestain image, the monochromatic nature caused blurring in the Euclidean case, but using Manhattan distance the image seemed to match the original to a greater degree. Of note, despite running the algorithm many times, the issue of k=2 decrementing to k = 1 for this image never arose when using Manhattan distance. The one color which did often get lost with this method versus Euclidean was the white values. Perhaps related to this is the fact that at k=4 and k = 8 the algorithm almost always encountered 1 and 3-4 empty cluster centers respectively.

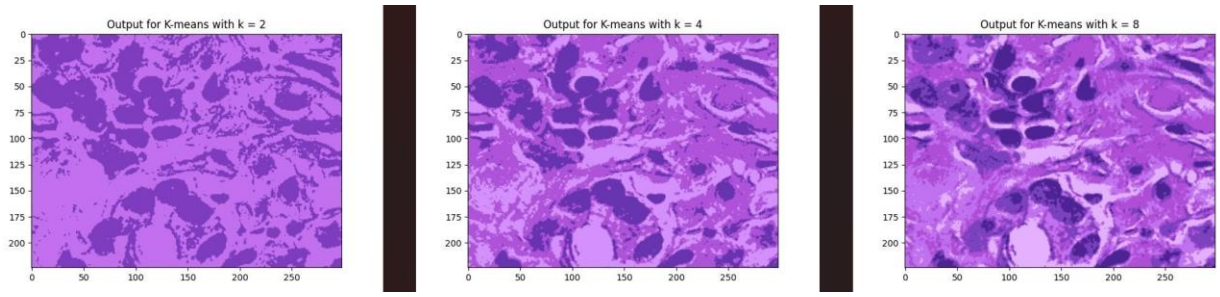


Figure 12. Outcome of hestain.bmp using Manhattan distance

In the case of test subject Ruby, the Manhattan distance performed rather poorly. Even at high k values, the background color was often lost without improvement to the color definition of the subject. If the background color was detected, it almost always appeared on top of the main subject. In addition, using Euclidian distance worked well since there are many similar colors and shades in this image and using Manhattan distance it appears that these subtle shades are lost. Two outcomes which show these issues are shown below, despite them being amongst the best-case scenarios.

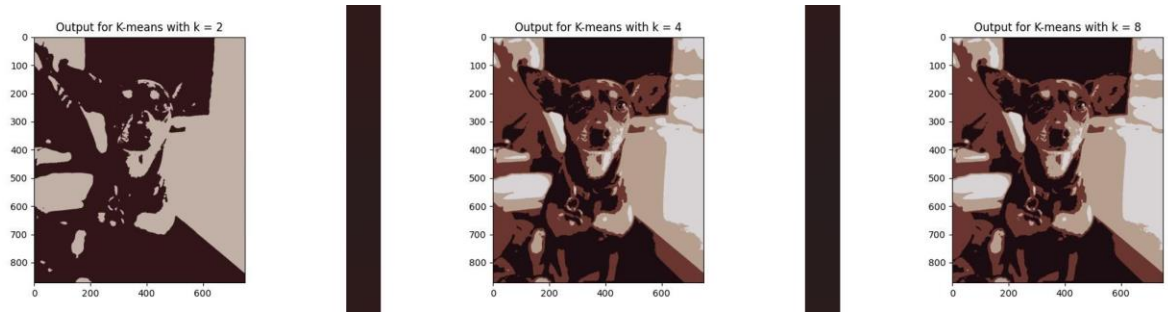


Figure 13. Ruby example 1 using Manhattan distance

Table 5. Convergence Values for Ruby example 1 with Manhattan distance

k	Iteration Time	Iterations to Convergence	Centroids		
			R	G	B
2	0.44800353	5	194	177	167
			49	21	25
4	0.5937342	6	183	159	143
			216	212	214
			28	13	18
			107	54	47
8	1.10170078	12	216	212	214
			107	54	48
			183	159	143
			28	13	18
			empty	empty	empty
			empty	empty	empty
			empty	empty	empty
			empty	empty	empty

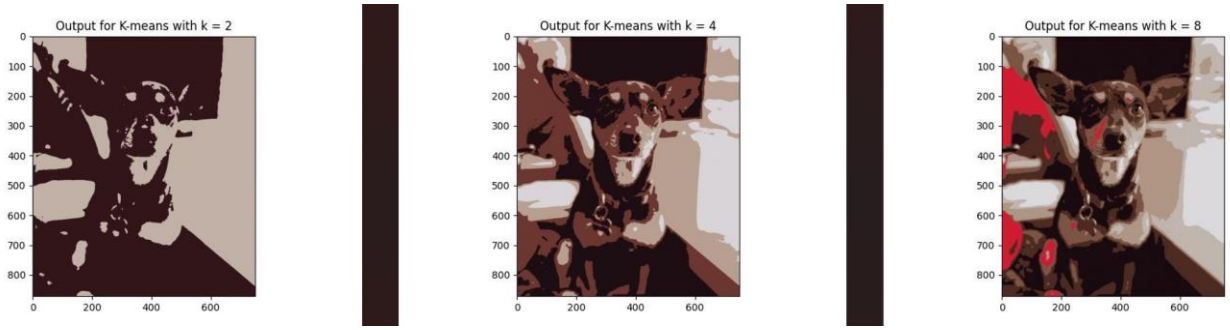


Figure 14. Ruby Example 2 using Manhattan distance

Table 6. Convergence Values for Ruby example 2 using Manhattan distance

k	Iteration Time	Iterations to Convergence	Centroids		
			R	G	B
2	0.379229	4	49	21	25
			194	177	167
4	1.361508	16	108	55	48
			217	212	215
			28	14	18
			184	160	145
8	1.8845577	19	176	150	133
			77	43	34
			192	182	175
			124	85	69
			25	11	16
			220	217	221
			209	28	50
			empty	empty	empty

Part 3: Political Blogs Dataset

1) Use spectral clustering to find the k = 2, 5, 10, 25 clusters in the network of political blogs (each node is a blog, and their edges are defined in the file edges.txt). Find majority labels in each cluster, for different k values, respectively. For example, if there are k = 2 clusters, and their labels are {0, 1, 1, 1} and {0, 0, 1} then the majority label for the first cluster is 1 and for the second cluster is 0. It is required you implement the algorithms yourself rather than calling from a package. Now compare the majority label with the individual labels in each cluster, and report the mismatch rate for each cluster, when k = 2, 5, 10, 25. For instance, in the example above, the mismatch rate for the first cluster is 1/4 (only the first node differs from the majority) and the second cluster is 1/3.

To answer parts 1 and 2 of this question, the code spectral_clustering.py was developed. For part 1, k values of [2, 5, 10, 25] were used, and the output is as follows. The first output for this program is the

mismatch rate in each cluster (i.e. for k=2, there are two mismatch rates, one for the percentage of mismatched items in each cluster), as well as the total number of items in the cluster. Finally, a list is printed showing the total mismatch rate for each spectral clustering repetition for the value of k in the list above and a plot is created to visualize the results.

Table 7. Mismatch Rates for k = 2

Cluster	Mismatch rate	Items in cluster	Overall Mismatch Rate
1	47.95%	1222	47.86%
2	0.00%	2	

Table 8. Mismatch Rates for k = 5

Cluster	Mismatch rate	Items in cluster	Overall Mismatch Rate
1	2.18%	551	4.82%
2	0.00%	2	
3	0.00%	2	
4	7.07%	665	
5	0.00%	2	

Table 9. Mismatch Rates for k = 10

Cluster	Mismatch rate	Items in cluster	Overall Mismatch Rate
1	2.26%	530	4.82%
2	0.00%	2	
3	0.00%	2	
4	0.00%	4	
5	0.00%	2	
6	0.00%	13	
7	0.00%	2	
8	0.00%	2	
9	0.00%	2	
10	7.07%	665	

Table 10. Mismatch Rates for k = 25

Cluster	Mismatch rate	Items in cluster	Overall Mismatch Rate
1	4.88%	41	25.08%
2	0.00%	148	
3	0.00%	2	
4	3.60%	222	
5	0.00%	2	
6	0.00%	2	
7	0.00%	2	
8	0.00%	2	
9	0.00%	2	
10	0.00%	2	
11	0.00%	3	
12	0.00%	4	
13	0.00%	3	
14	0.00%	2	
15	0.00%	2	
16	0.00%	2	
17	0.00%	163	
18	25.00%	4	
19	0.00%	11	
20	0.00%	4	
21	14.29%	7	
22	0.00%	4	
23	0.00%	3	
24	0.00%	7	
25	44.14%	580	

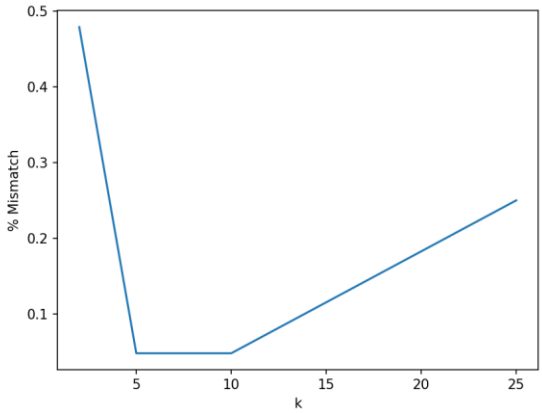


Figure 15. Changing Mismatch Rate with k in [2,5,10,25]

2) Tune your k and find the number of clusters to achieve a reasonably small mismatch rate. Please explain how you tune k and what is the achieved mismatch rate. Please explain intuitively what this result tells about the network community structure.

To tune k, we can plot the mismatch rate for all values of k and try to identify the value of k which produces a low mismatch rate, and beyond which the algorithm shows little improvement which would mean increasing k beyond this elbow value has little benefit. From the plot below we can see that around k = 10 the best mismatch rate is found beyond which the mismatch rate oscillates up and down so this value could be chosen as the best k value. **Of most importance**, when we look at the number of items in each cluster for k = 10 in table 9, we can see that the overwhelming majority of items still fall in two groups, indicating that on the whole, most blogs fall in one of two communities depending on their political views. Our k = 2 does not seem to separate these two communities correctly, however, if we were able to tune the algorithm such that it correctly splits the two groups it could be expected that k = 2 is the optimal value.

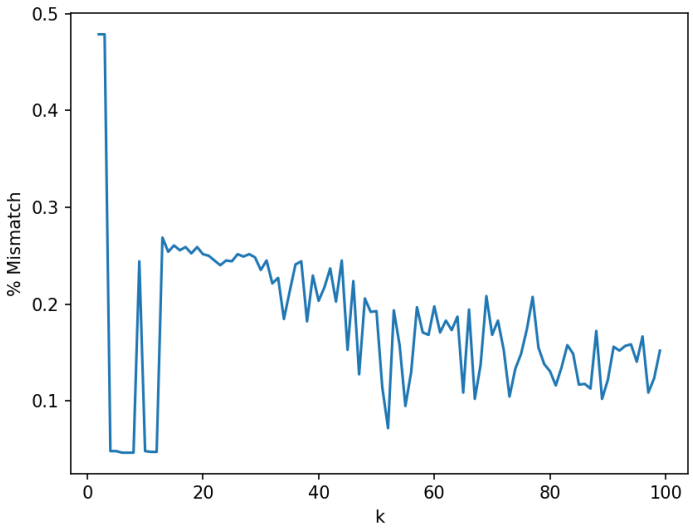


Figure 16. Mismatch Rates for Varied k Values

Part 4: Math of k-Means Clustering

1) (3 points) Derive mathematically that using the squared Euclidean distance $\|x^i - \mu^j\|^2$ as the dissimilarity function, the centroid that minimizes the distortion function J for given assignments r^{ij} are given by

$$\mu^j = \frac{\sum_i r^{ij} x^i}{\sum_i r^{ij}}.$$

That is, μ^j is the center of j-th cluster.
Hint: You may start by taking the partial derivative of J with respect to μ^j, with r^{ij} fixed.

First, taking the partial derivative of J with respect to μ^j for fixed r^{ij} gives the equation below.

$$\frac{\partial J}{\partial u^j} = -\sum_{i=1}^{m} 2r^{ij}(x^i - \mu^j)$$

Setting equal to 0, we can separate the equation by distributing over $(x^i - \mu^j)$ to get:

$$0 = -2\sum_{i=1}^{m} r^{ij}x^i + 2\sum_{i=1}^{m} r^{ij}\mu^j$$

Finally, we can isolate for μ^j:

$$\mu^j = \frac{\sum_{i=1}^{m} r^{ij}x^i}{\sum_{i=1}^{m} r^{ij}}$$

2) (2 points) Derive mathematically what should be the assignment variables r^{ij} be to minimize the distortion function J, when the centroids μ^j are fixed.

Since r^{ij} can only be 1, or 0 if the data point belongs to the cluster or not when the centroids are fixed, r^{ij} can only be 1 if the distance from x^i to μ^j where j = 1 for example is less than for any other value of j. As such, r^{ij} can be written as follows:

$$r^{ij} = \begin{cases} 1 \; if \; j = argmin_{j=1\rightarrow k}\left\|x^i - \mu^j\right\|^2 \\ 0 \; otherwise \end{cases}$$