ISYE 6740 Spring 2024
Homework 1 (100 points + 10 bonus points)

Yuxi Chen

1 Concept questions [35 points]

Please provide a brief answer to each question, and provide math arguments if asked.

1. (5 points) What's the main difference between supervised and unsupervised learning? Give one benefit and drawback for supervised and unsupervised learning, respectively.

   The main difference between is the whether the algorithm is provided labeled or unlabeled datasets. In supervised learning, the algorithm is given labeled datasets to train on. This will allow the learning algorithm to assign labels to new datasets provided. The benefit of supervised learning is that once the model is trained it can provide accurate predictions. The drawback however, is that the size of the dataset can create challenges with labeling and run time.
   In unsupervised learning, an unlabeled data set is given to allow the learning algorithm to find patterns within the data. The benefit of unsupervised learning is that it is used for finding patterns and exploratory data analysis. The drawback is that it can be difficult to measure the accuracy and performance of the model

2. (5 points) Consider a dataset with data points each having 3 features, e.g., $x^1 = \{$"Atlanta", "house", $500k\}$, and $x^2 = \{$"Houston", "house", $300k\}$. Define a proper similarity function $d(x^i, x^j)$ for this kind of data, and argue why it is a reasonable choice. (Hint: The feature vector consists of categorial and real-valued features; for categorical variables, it is better to convert them into one-hot-keying binary vectors and use Hamming distance, and for real-valued features, you may use Euclidean distance, for instance. And then you can combine the similarity measure in some way.)

   To calculate Hamming distance, for every categorical feature in x, we can assign it to δ_f:.

   $$D_H(x_i, x_j) = \sum_{i=1}^{f} \delta_f$$

   where f indicates the features for each variable,

   $$\delta_f = \begin{cases} 0 & \text{if } f^i = f^j \\ 1 & \text{otherwise} \end{cases}$$

   To calculate euclidean distance:

   $$D_E(x^i, x^j) = \sqrt{(f^i - f^j)^2}$$

   We can then assign a weight to each distance calculation to determine which features are more significant.

   $$d(x^i, x^j) = w_H(D_H(x^i, x^j)) + w_E(D_E(x^i, x^j))$$

3. (5 points) Show that the clustering assignment problem

$$\pi(i) = \arg \min_{j=1,\ldots,k} \|x^i - c^j\|^2$$

is equivalent to solving

$$\pi(i) = \arg \min_{j=1,\ldots,k} (c^j)^T \left(\frac{1}{2}c^j - x^i\right).$$

Note that the second approach will facilitate "vectorized" operation and implemented in our demo code.

$$\|x^i - c^j\|^2 = (x^i - c^j)^T(x^i - c^j)$$
$$\Rightarrow (x^i)^T x^i - 2(x^i)^T c^j + (c^j)^T c^j$$

We can focus on the part of the equation using j

$$\pi(i) = \arg \min_{j=1,\ldots,k} -2(x^i)^T c^j + (c^j)^T c^j$$

$$\Rightarrow (c^j)^T c^j - 2(c^j)^T x^i = \frac{1}{2}(c^j)^T c^j - (c^j)^T x^i$$

$$\pi(i) = \arg \min_{j=1,\ldots,k} (c^j)^T \left(\frac{1}{2}c^j - x^i\right)$$

4. (5 points) Why different initializations for k-means lead to different results?

The reason why different initializations for k-means lead to different results can be due to various reasons. One reason is because the first run may not converge on the global minima and will require multiple initalizations. Each run may converge to different local minima depending where it starts on the cluster.

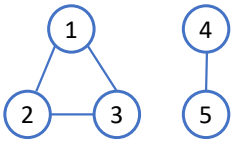
5. (5 points) Why k-means will (be guaranteed to) stop after a finite number of iterations?

The reason why k-means will stop after a finite number of iterations can attributed to the defined number of data point and cluster assignments provided. This also means that after each iteration, the sum of squared distance and error will be reduce as we converge to the centroid. Eventually we will no longer be able to converage any further as each iteration will not improve the distance or error, therefore it will end in finite iterations.

6. (5 points) What is the main difference between k-means and generalized k-means algorithm? Explain how the choice of similarity/dissimilarity/distance will impact the result.

The main difference between k-means and generalized k-means algorithm is the in their method of measuring similarity/dissimilarity/distance. K-means leverages the euclidian distance to find the distance from the cluster. In generalized k-means algorithm, the model has the ability to use other methods to measure the similiarity/dissimiliarity. By using other methods to interpret the data differently, it allows the model to form clusters differently, which in turn changes the performance of the model. Another benefit of using generalized k-means, is being able to interpret non numerical data, however, a downside will be poor performance based on chosen method of measurement.

7. (5 points) Consider the following simple graph



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

Laplace Matrix = Degree Matrix - Adjacency Matrix

$$\text{Adjacency Matrix} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{Degree Matrix} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{Laplacian Matrix} = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Below are the calculations for eigenvalues:

```
eigenval, eigenvec = np.linalg.eig(L)
lambd = np.diag(eigenval)

```

```
print("Eigenval = ",eigenval)
```

```
Eigenval =  [ 3.0000000e+00 -4.4408921e-16  3.0000000e+00  2.0000000e+00
  0.0000000e+00]
```

```
print("Eigenvec = ",eigenvec)
```

```
Eigenvec =  [[ 0.81649658 -0.57735027  0.29329423  0.          0.        ]
 [-0.40824829 -0.57735027 -0.80655913  0.          0.        ]
 [-0.40824829 -0.57735027  0.5132649   0.          0.        ]
 [ 0.          0.          0.          0.70710678  0.70710678]
 [ 0.          0.          0.         -0.70710678  0.70710678]]
```

```
print("lambd = ",lambd)
```

```
lambd =  [[ 3.0000000e+00  0.0000000e+00  0.0000000e+00  0.0000000e+00
   0.0000000e+00]
 [ 0.0000000e+00 -4.4408921e-16  0.0000000e+00  0.0000000e+00
   0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  3.0000000e+00  0.0000000e+00
   0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  0.0000000e+00  2.0000000e+00
   0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  0.0000000e+00  0.0000000e+00
   0.0000000e+00]]
```

From the calculation above, we can see that there are two zero eigenvalues, located at index 1 and 4 of variable eigenval. This indicates that there are two disconnected clusters and using the eigen vector we can determine which cluster belongs to which group in the graph. Referencing the variable eigenvec, we can see that the 1st cluster is (1,2,3) and 2nd cluster is (4,5).

2 Math of k-means clustering [20 points]

Given m data points x^i, $i = 1, \ldots, m$, K-means clustering algorithm groups them into k clusters by minimizing the distortion function over $\{r^{ij}, \mu^j\}$

$$J = \sum_{i=1}^{m} \sum_{j=1}^{k} r^{ij} \|\mathrm{x}^i - \mu^j\|^2, \tag{1}$$

where $r^{ij} = 1$ if x^i belongs to the j-th cluster and $r^{ij} = 0$ otherwise.

1. (10 points) Derive mathematically that using the squared Euclidean distance $\|\mathrm{x}^i - \mu^j\|^2$ as the dissimilarity function, the centroid that minimizes the distortion function J for given assignments r^{ij} are given by

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

   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.

   $$\frac{\partial J}{\partial \mu^j} = \sum_{i=1}^{m} 2r^{ij}(x^i - \mu^j) = 0$$

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

   $$\Rightarrow \sum_{i=1}^{m} r^{ij} x^i = \sum_{i=1}^{m} r^{ij} u^j$$

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

2. (10 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.

   $$J = \sum_{j=1}^{k} r^{ij} \|\mathrm{x}^i - \mu^j\|^2 \tag{2}$$

   When centroids μ^j are fixed, we only need to work with i. In this case given the nature of k-means clustering algorithm, for each data point i, r^{ij} can only be 1 or 0 and $\sum_{j=1} r^{ij} = 1$

   $$r^{ij} = \begin{cases} 1 & \text{if } j = \arg\min_{j=1,\ldots,k} \|x^i - \mu^j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

3 Image compression using clustering [20 points]

In this programming assignment, you are going to apply clustering algorithms for image compression. This can also be viewed as an example of segmenting colors in an automated fashion using K-means clustering.

Your task is to implement *K-means* for this purpose. **It is required you implement the algorithms yourself rather than calling k-means from a package. However, it is ok to use standard packages for supplementary tasks, e.g., file i/o, linear algebra, and visualization.**

Formatting instruction

As a starting point, we suggest the following input/output signature for your k-means algorithm.

Input

- `pixels`: the input image representation. Each row contains one data point (pixel). For image dataset, it contains 3 columns, each column corresponding to Red, Green, and Blue components. Each component has an integer value between 0 and 255.
- `k`: the number of desired clusters.

Output

- `class`: cluster assignment of each data point in pixels. The assignment should be 1, 2, 3, etc. For $k = 5$, for example, each cell of the class should be either 1, 2, 3, 4, or 5. The output should be a column vector with `size(pixels, 1)` elements.
- `centroid`: location of k centroids (or representatives) in your result. With images, each centroid corresponds to the representative color of each cluster. The output should be a matrix with K rows and 3 columns. The range of values should be [0, 255], possibly floating point numbers.

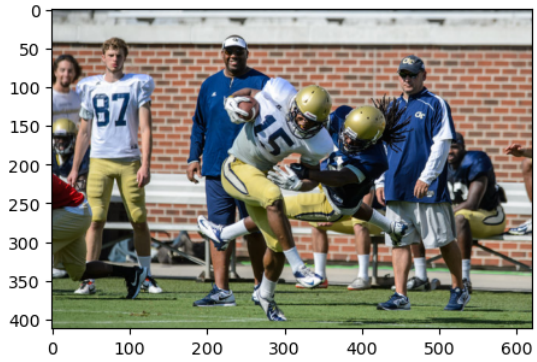
Hand-in

Both of your code and report will be evaluated. Upload the code as a zip file, and the report as a pdf, separately from the zip file. In your report, answer the following questions:

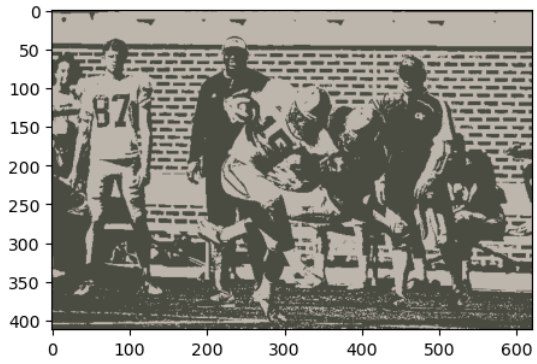
1. (10 points) Use k-means with squared-ℓ_2 norm as a metric for `hestain.bmp` and `football.bmp` and also choose a third picture of your own to work on. We recommend the size of 320×240 or smaller. Run your k-means implementation with these pictures, with several different $k = 2, 3, 4, 5, 6$.

   Comment: `hestain.png` is an image of tissue stained with hematoxylin and eosin (H&E). This staining method helps pathologists distinguish between tissue types that are stained blue-purple and pink. Then the algorithm will segment the image into k regions in the RGB color space. For each pixel in the input image, the algorithm returns a label corresponding to a cluster.

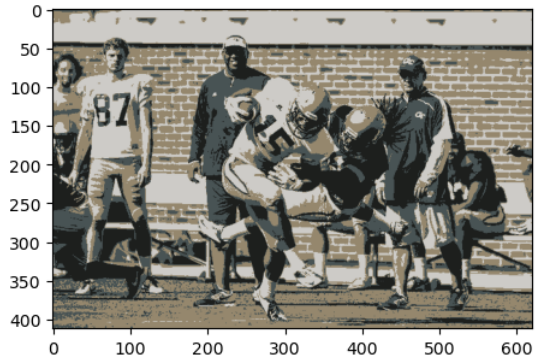
   Run your k-means implementation (with squared-ℓ_2 norm) with random initialization centroids. Please try multiple times and report the best one for each k (in terms of image quality).



Starting Image

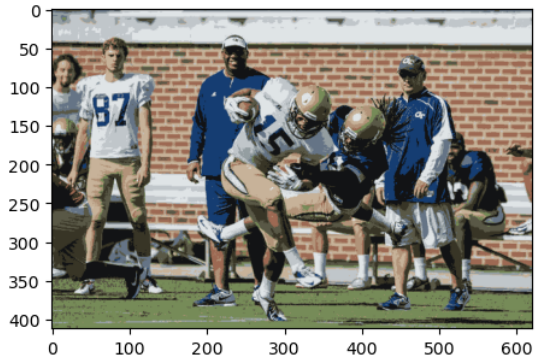


Run time: 0.9526278972625732
k = 2
Iterations = 16

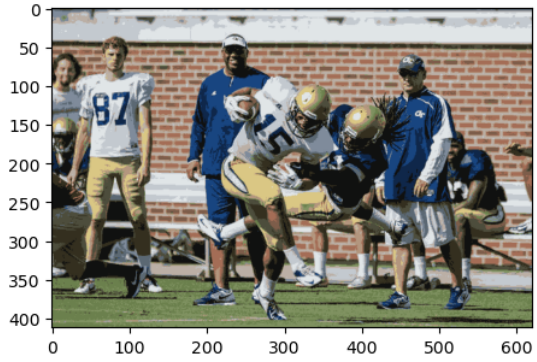


Run time: 3.4245219230651855
k = 4
Iterations = 52

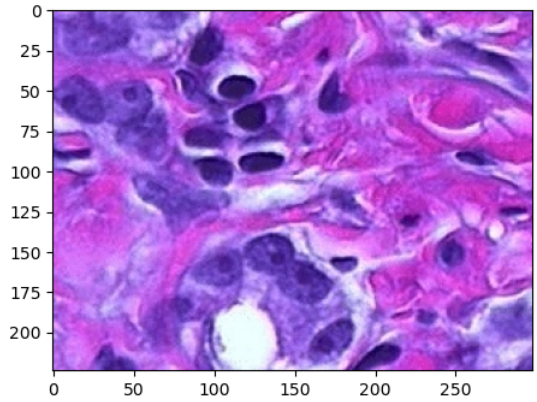
Run time: 5.927335977554321
k = 8
Iterations = 55



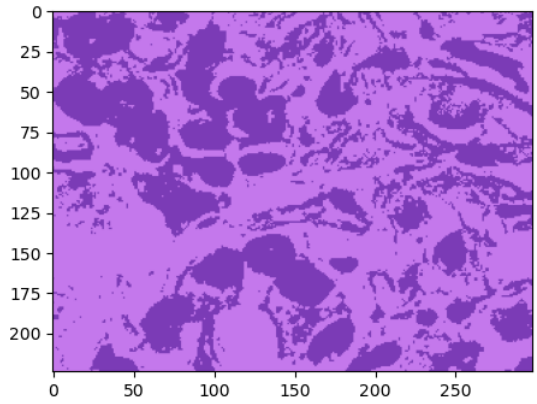
Run time: 24.01438593864441
k = 12
Iterations = 159



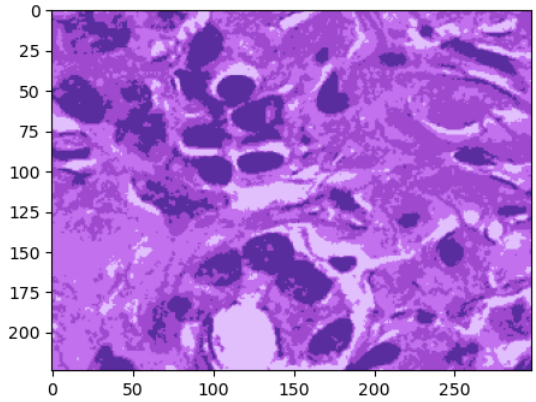
Run time: 24.701567888259888
k = 16
Iterations = 134



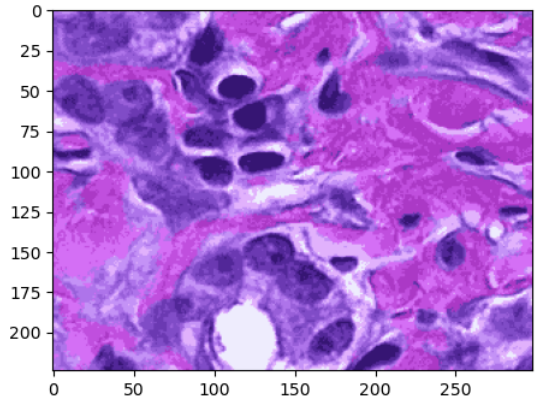
Starting Image



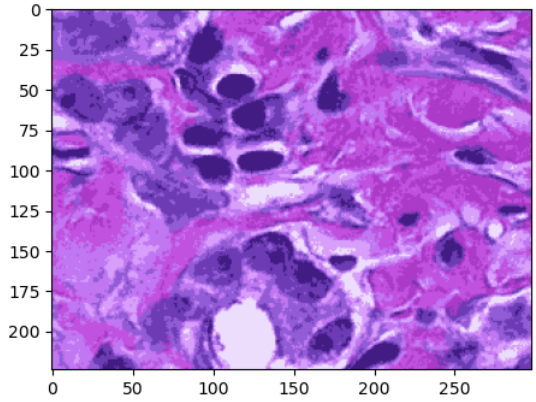
Run time: 0.3773519992828369
k = 2
Iterations = 26



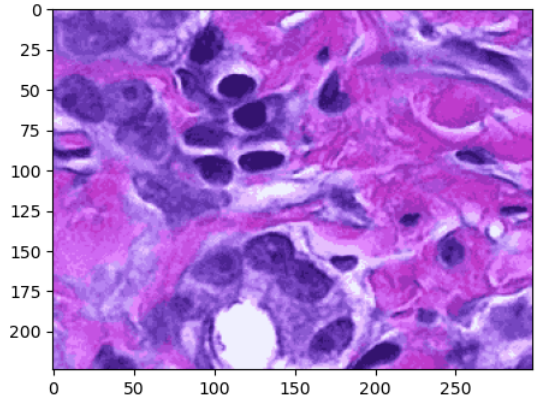
Run time: 0.7380750179290771
k = 4
Iterations = 42



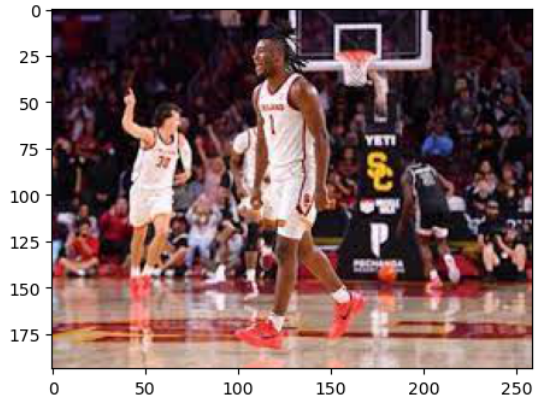
Run time: 2.056425094604492
k = 12
Iterations = 51



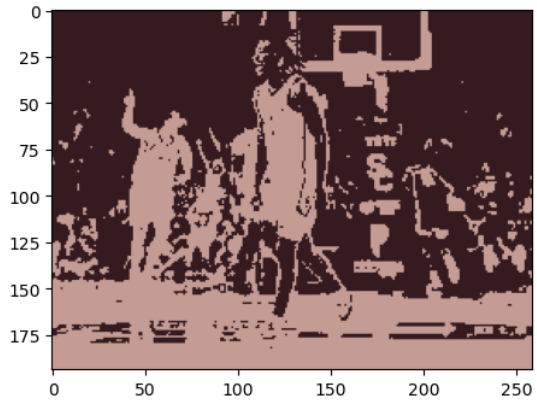
Run time: 1.607118844985962
k = 8
Iterations = 57



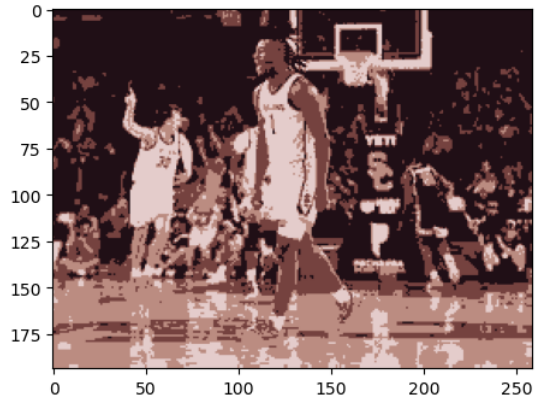
Run time: 5.831842660903931
k = 16
Iterations = 119



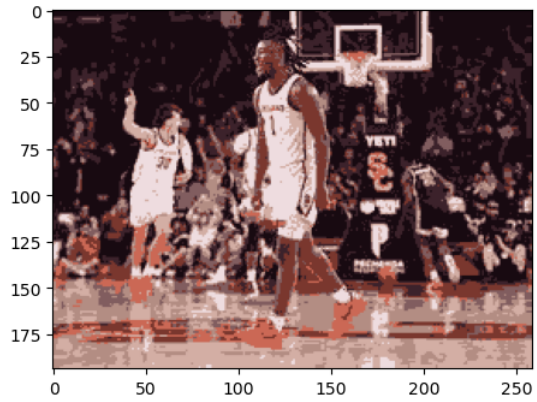
Starting Image



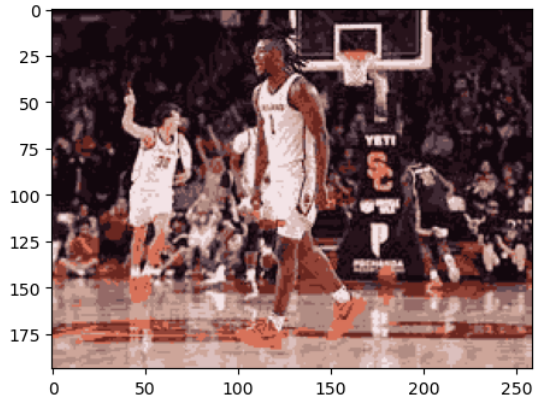
Run time: 0.29012489318847656
k = 2
Iterations = 16



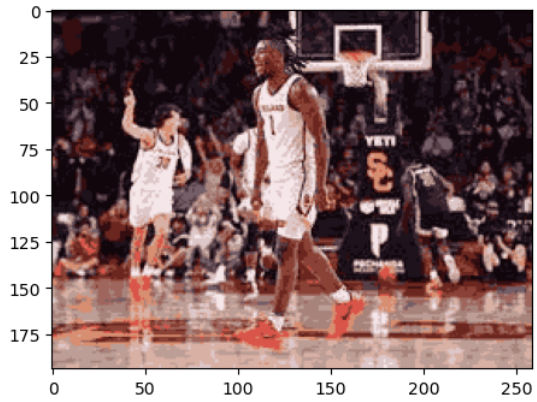
Run time: 0.4723498821258545
k = 4
Iterations = 31



Run time: 1.874823808670044
k = 8
Iterations = 94




```
Run time: 4.902047872543335
k = 12
Iterations = 175
```




```
Run time: 4.844647169113159
k = 16
Iterations = 123
```

2. (5 points) Please report how long it takes to converge for each k (report the number of iterations and elapsed time in seconds).

   Please see above for time taken to converge at each k and number of iterations.

3. (5 points) Describe a method to find the best k. What is your best k?

   Visually k=16, provides the closest image to the original. However, to measure the best k value we would need to use either the elbow curve or silhouette score method. For the elbow curve, you need to calculate the within cluster sum squared error. This method can be visually used to determine the best k value best inspecting with k value creates an elbow curve. The idea is that as we increase the clusters the errors will decrease as we reach the center.

Note

- You may see errors caused by empty clusters when you use too large k. Your implementation should treat this exception as well. That is, do not terminate even if you have an empty cluster, but automatically decrement to a smaller number of clusters.
- We recommend you test your code with several different pictures so that you can detect some problems that might happen occasionally.
- If we detect plagiarism from any other student's code or from the web, you will not be eligible for any credit for the entire homework, not just for this problem.

4 MNIST Dataset clustering [25 points]

This question is to compare different classifiers and their performance for multi-class classifications on the complete MNIST dataset at `http://yann.lecun.com/exdb/mnist/`. You can find the data file **mnist_10digits.mat** in the homework folder. The MNIST database of handwritten digits has a training set of 60,000 examples and a test set of 10,000 examples. Use the number of clusters $K = 10$.

We suggest you "standardize" the features (pixels in this case) by dividing the values of the features by 255 (thus mapping the range of the features from [0, 255] to [0, 1]).

We are going to use *purity* score as a performance metric: each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by the number of correlated assigned samples and divided by the size of the cluster:

$$\text{purity}_i = \frac{\text{corrected assigned samples}_i}{\text{size of cluster}_i}$$

for the cluster i.

1. (15 points) Use the squared-ℓ_2 norm as a metric for clustering (you may base it on the code you had for Question 2 and make necessary changes.) Report the *purity* score for each cluster.

2. (10 points) Now try your k-means with the Manhattan distance (or ℓ_1 distance) and repeat the same steps in Part (1). Please note that the assignment of data points 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. Report the *purity* score for each cluster. Comment on which metric gives the better result?

5 Political blogs dataset [bonus, 10 points]

We will study a political blog dataset first compiled for the paper Lada A. Adamic and Natalie Glance, "The political blogosphere and the 2004 US Election", in Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem (2005). It is assumed that blog-site with the same political orientation are more likely to link to each other, thus, forming a "community" or "cluster" in a graph. In this question, we will see whether or not this hypothesis is likely to be true based on the data.

- The dataset nodes.txt contains a graph with $n = 1490$ vertices ("nodes") corresponding to political blogs.
- The dataset edges.txt contains edges between the vertices. You may remove isolated nodes (nodes that are not connected to any other nodes) in the pre-processing.

We will treat the network as an undirected graph; thus, when constructing the adjacency matrix, make it symmetrical by, e.g., set the entry in the adjacency matrix to be one whether there is an edge between the two nodes (in either direction).

In addition, each vertex has a 0-1 label (in the 3rd column of the data file) corresponding to the true political orientation of that blog. We will consider this as the true label and check whether spectral clustering will cluster nodes with the same political orientation as possible.

1. (5 points) 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 $1/3$.

2. (5 points) 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.