ISYE 6740 Spring 2025
Homework 1 (100 points + 10 bonus points)

In this homework, the superscript of a symbol x^i denotes the index of samples (not raising to ith power); this is a convention in this class. Please follow the homework submission instructions in the syllabus.

Provided Data:

- Q3: Image compression using clustering [football.bmp, sea-turtle-400x225.png]
- Q4: Political Blogs Spectral Clustering [nodes.txt, edges.txt]
- Q5: MNIST Clustering [mnist_10digits.mat]

All results that are present only in code/notebooks, but not in your PDF report will not be accepted for points.

1 Concept questions [30 points]

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

1. (5 points) How do supervised and unsupervised learning differ in their approaches? Highlight 3 advantages and two limitations of each method.

> Supervised Learning involves the use of labeled data, where the algorithm learns from input-output pairs to make predictions or classifications. However, Unsupervised Learning involves unlabeled data, where the algorithm identifies patterns or clusters without explicit supervision.
>
> - **Advantages of Supervised Learning**: It produces more accurate and reliable results for well-labeled datasets, works well for predictive tasks, and can generalize well with proper training and enough data.
> - **Advantages of Unsupervised Learning**: It can identify hidden patterns or structures in data without the need for labels, is useful for exploratory data analysis, and requires less preparation and labeling of the dataset, saving time and effort.
> - **Disadvantages of Supervised Learning**: It requires a large amount of labeled data, which is often expensive and time-consuming to initialize and the performance heavily depends on the quality of labels, where noisy data can lead to inaccurate models.
> - **Disadvantages of Unsupervised Learning**: The results are often less interpretable compared to supervised learning, and there's no straightforward way to evaluate the performance of unsupervised learning since there are no labels.

2. (5 points) Consider a dataset with data points each having 3 features, e.g., $x^1 = \{\text{“Atlanta”}, \text{“house”}, 500k\}$, and $x^2 = \{\text{“Dallas”}, \text{“house”}, 300k\}$. 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.)

We can define $d(x^i, x^j)$ as:

$$d(x^i, x^j) = \alpha d_{\text{c}}(f_1^i, f_1^j) + \beta d_{\text{c}}(f_2^i, f_2^j) + \gamma d_{\text{n}}(f_3^i, f_3^j),$$

where:

- d_{c}: similarity measure for categorical features
- d_{n}: similarity measure for numerical features
- α, β, γ: weights for each feature

Convert f_1 and f_2 into one-hot encoded vectors and then compute the Hamming distance:

$$d_{\text{c}}(f_a, f_b) = \begin{cases} 0 & \text{if } f_a = f_b \\ 1 & \text{if } f_a \neq f_b \end{cases}$$

For f_3, compute the scaled Euclidean distance:

$$d_{\text{n}}(f_3^i, f_3^j) = \frac{|f_3^i - f_3^j|}{\sigma},$$

where σ is a scaling factor to normalize the distance.

This similarity function works well because it handles mixed data types appropriately. One-hot encoding with Hamming distance captures differences in categorical features, while normalized Euclidean distance handles numerical features. By combining these with adjustable weights, the method balances each feature and adapts to different data contexts.

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

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

is equivalent to solving

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

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

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

$$\arg \min_{j=1,\dots,k} \|x^i - c^j\|^2 = \arg \min_{j=1,\dots,k} {c^j}^T c^j - 2{c^j}^T x^i + {x^i}^T x^i$$

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

$$\arg \min_{j=1,\dots,k} \|x^i - c^j\|^2 = \arg \min_{j=1,\dots,k} 2{c^j}^T \left(\frac{1}{2} c^j - x^i \right) \quad \text{dropped as it does not contain } c^j$$

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

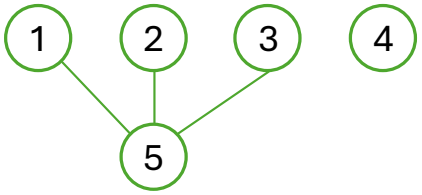
4. (5 points) Why do different initializations for k-means lead to different results? In practice, to finding a better result facing this issue, what would you do?

Different initializations lead to different results due to K-means' sensitivity to starting centroids and its reliance on local optimization. To address this, we can run K-means with different random initializations (e.g., 10 iterations) and select the result with the lowest final distortion. This allows for a better starting point and the ability to save computational resources. Additionally, the K-Means++ algorithm selects initial centroids in a way that spreads them out in the data space, which improves the likelihood of finding a good solution and reduces the issues surrounding initialization.

5. (5 points) Why k-means will not have the issue of running a infinite number of iterations (suppose the stopping criterion is when the cluster assignments after another iteration do not change), in most settings?

K-means will not run for infinite iterations because it runs in a finite solution space, where the number of possible cluster assignments is k^m, where we have m data points and k clusters. The algorithm minimizes the distortion function, which decreases monotonically at each step and is bounded below, ensuring convergence. Additionally, K-means stops when centroids no longer change, meeting the stopping criteria. These factors guarantee that the algorithm terminates after a finite number of iterations.

6. (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.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix} \quad L = D - A = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 3 \end{bmatrix}$$

$$L - \lambda I = \begin{bmatrix} 1-\lambda & 0 & 0 & 0 & -1 \\ 0 & 1-\lambda & 0 & 0 & -1 \\ 0 & 0 & 1-\lambda & 0 & -1 \\ 0 & 0 & 0 & -\lambda & 0 \\ -1 & -1 & -1 & 0 & 3-\lambda \end{bmatrix}$$

$$\det(L - \lambda I) = -\lambda^5 + 6\lambda^4 - 9\lambda^3 + 4\lambda^2 = 0$$
$$\det(L - \lambda I) = \lambda(\lambda^2 - 4\lambda + 3)(\lambda^2 - 2\lambda) = 0$$
$$\lambda = 0, 0, 1, 1, 4$$

Eigenvalue $\lambda = 0$:

$$\mathbf{v}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$\mathbf{v}_1$ corresponds to the isolated node $\{4\}$

$\mathbf{v}_2$ corresponds to the connected component $\{1, 2, 3, 5\}$

2 Math of k-means clustering [20 points]

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

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

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

1. (8 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.

$$\|x^i - \mu^j\|^2 = (x^i - \mu^j)^T (x^i - \mu^j)$$

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

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

$$\frac{\partial J_j}{\partial \mu^j} = \frac{\partial}{\partial \mu^j} \sum_{i=1}^{m} r^{ij} \left[(x^i - \mu^j)^T (x^i - \mu^j)\right]$$

$$\frac{\partial J_j}{\partial \mu^j} = \sum_{i=1}^{m} r^{ij} \cdot \frac{\partial}{\partial \mu^j} \left[(x^i - \mu^j)^T (x^i - \mu^j)\right]$$

$$(x^i - \mu^j)^T (x^i - \mu^j) = (x^i)^T x^i - 2(x^i)^T \mu^j + (\mu^j)^T \mu^j$$

$$\frac{\partial}{\partial \mu^j} \left[(x^i - \mu^j)^T (x^i - \mu^j)\right] = -2x^i + 2\mu^j$$

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

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

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

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

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

2. (7 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.

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

$$\sum_{j=1}^{k} r^{ij} = 1, \quad \forall i$$

The terms $\frac{1}{mk}$ and the summation over i do not depend on the decision for r^{ij}. Thus, to minimize J, for each data point x^i, we can minimize:

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

Since r^{ij} is binary, we only need to worry about selecting the cluster j that minimizes the squared distance $\|x^i - \mu^j\|^2$.

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

This means that we assign each data point x^i to the cluster whose centroid μ^j is closest to x^i in terms of Euclidean distance.

The minimization of J is equivalent to minimizing $\|x^i - \mu^j\|^2$ for each x^i, as the summation over all i and the scaling factor $\frac{1}{mk}$ do not affect the individual assignment.

Thus:

- $r^{ij} = 1$ for the closest cluster j
- $r^{ij} = 0$ otherwise

3. (5 points) For the question above, now suppose we change the similar score to a "quadratic" distance (also known as Mahalanobis distance) for given and fixed positive definite matrix $\Sigma \in \mathbb{R}^{n \times n}$, and the distortion function becomes:

$$J = \frac{1}{mk} \sum_{i=1}^{m} \sum_{j=1}^{k} r^{ij} (\mathrm{x}^i - \mu^j)^T \Sigma (\mathrm{x}^i - \mu^j),$$

Derive what μ^j and r^{ij} becomes in this case.

$$J_j = \sum_{i=1}^{m} r^{ij} (x^i - \mu^j)^T \Sigma (x^i - \mu^j)$$

$$(x^i - \mu^j)^T \Sigma (x^i - \mu^j) = (x^i)^T \Sigma x^i - 2(\mu^j)^T \Sigma x^i + (\mu^j)^T \Sigma \mu^j$$

$$\frac{\partial J_j}{\partial \mu^j} = \frac{\partial}{\partial \mu^j} \sum_{i=1}^{m} r^{ij} \left[(x^i)^T \Sigma x^i - 2(\mu^j)^T \Sigma x^i + (\mu^j)^T \Sigma \mu^j \right]$$

$$\frac{\partial J_j}{\partial \mu^j} = \sum_{i=1}^{m} r^{ij} \left[-2\Sigma x^i + 2\Sigma \mu^j \right]$$

$$\frac{\partial J_j}{\partial \mu^j} = 2\Sigma \left(\sum_{i=1}^{m} r^{ij} \mu^j - \sum_{i=1}^{m} r^{ij} x^i \right)$$

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

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

$$r^{ij} = \begin{cases} 1 & \text{if } j = \arg\min_{j'} (x^i - \mu^{j'})^T \Sigma (x^i - \mu^{j'}) \\ 0 & \text{otherwise} \end{cases}$$

3 Image compression using clustering [25 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, distance calculations, 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 `sea-turtle-400x225.jpeg` and `football.bmp` and also choose a third picture of your own to work on. Your chosen image should meet the following requirements:
   - Full Color (No black and white images)
   - Recommended size between 200x150 and 400x300. Larger images are acceptable assuming they meet runtime requirements, Smaller images are not acceptable.

   Run your k-means implementation with these pictures, with several different $k = 3, 6, 12, 24, 48$.

   Comment: Your 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. Due to the nature of randomness, Please try multiple times and report the only the best seed (in terms of image quality). Please provide the following deliverables:
   - For every K on all 3 images, the reconstructed image
   - The time in seconds it takes to converge for every K on each image
   - The number of iterations to convergence for every K on each image

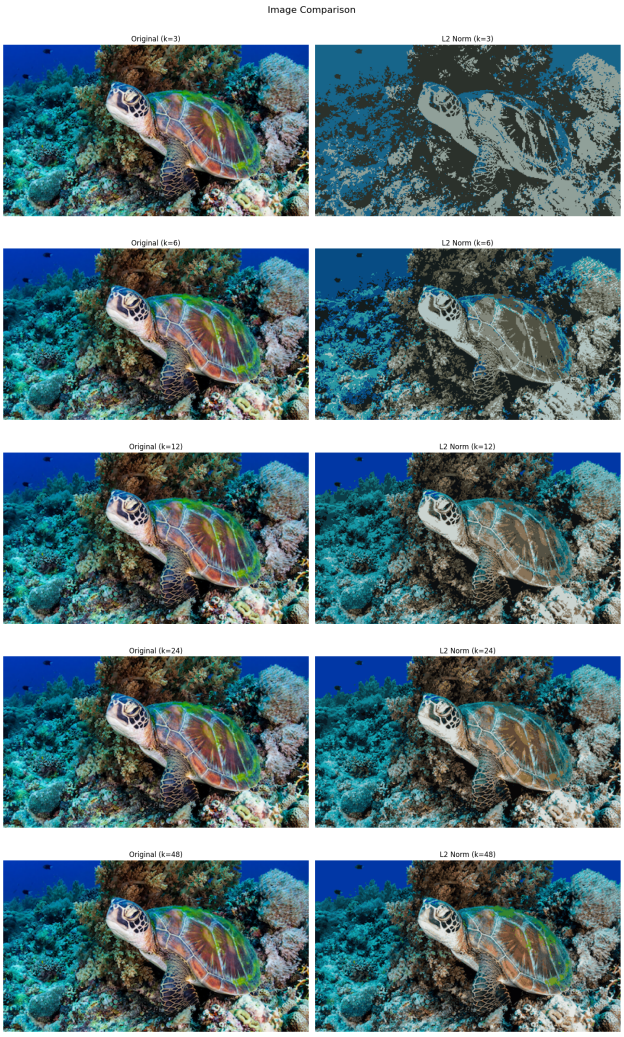


Image Comparison
Original (k=3)
L2 Norm (k=3)
Original (k=6)
L2 Norm (k=6)
Original (k=12)
L2 Norm (k=12)
Original (k=24)
L2 Norm (k=24)
Original (k=48)
L2 Norm (k=48)

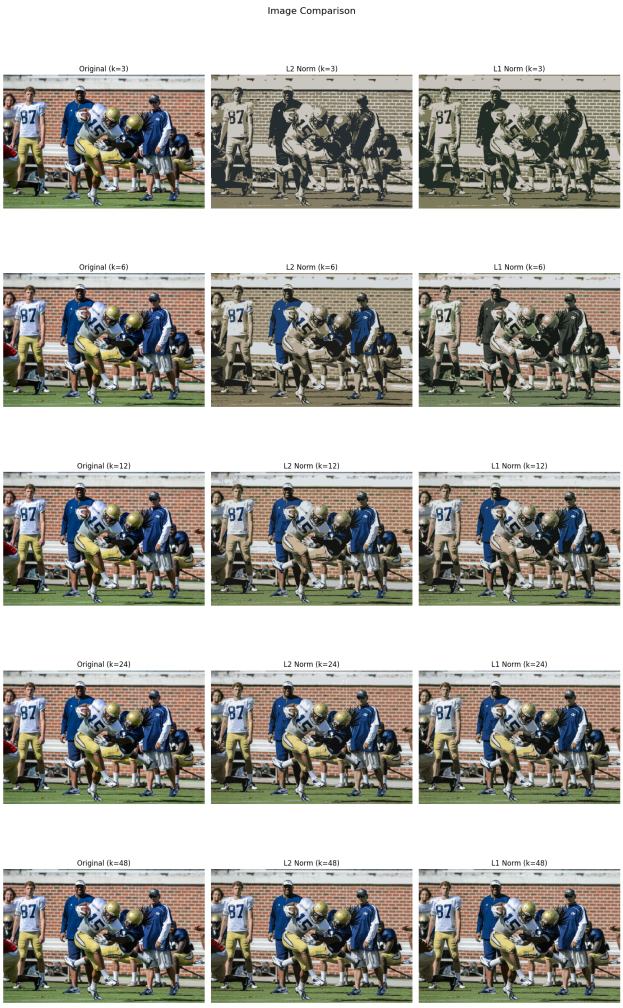
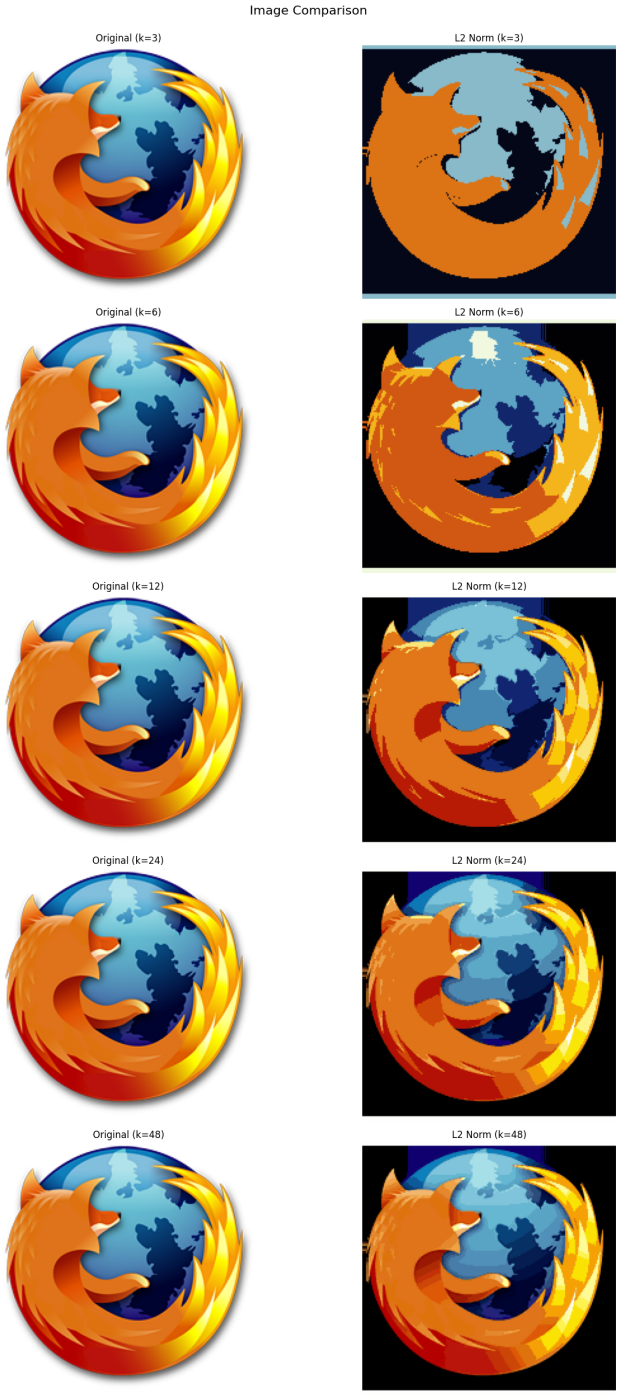


Image Comparison
Original (k=3)
L2 Norm (k=3)
L1 Norm (k=3)
Original (k=6)
L2 Norm (k=6)
L1 Norm (k=6)
Original (k=12)
L2 Norm (k=12)
L1 Norm (k=12)
Original (k=24)
L2 Norm (k=24)
L1 Norm (k=24)
Original (k=48)
L2 Norm (k=48)
L1 Norm (k=48)



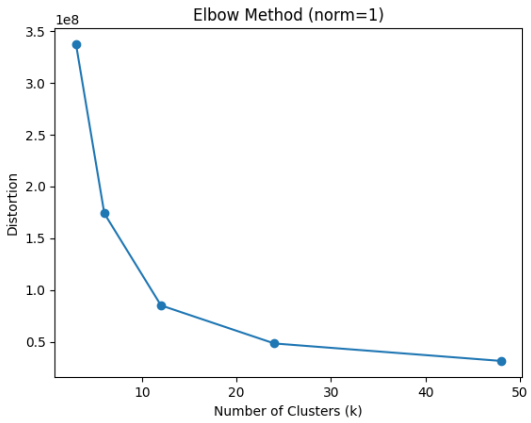
2. (10 points) Now adjust your k-means implementation to use the Manhattan distance (ℓ_1 norm) metric. Please provide all of the same above deliverables for only the football.bmp image. Provide a comment on any differences you see in the results between using the L2 norm and L1 norm metrics.

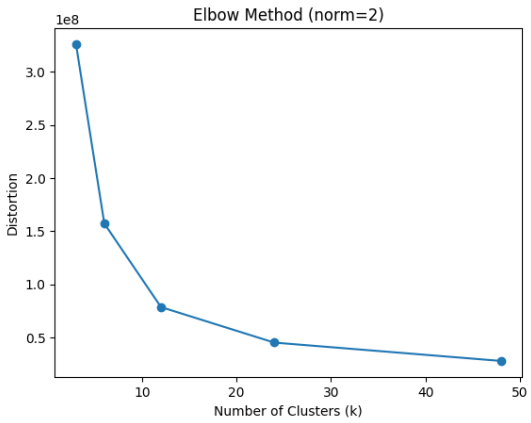
> The ℓ_2-based images typically look more natural and smooth in their segmentation. The ℓ_1-based images may exhibit sharper boundaries between color segments and also seem less refined

or blockier in areas with subtle gradients.

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

To find the best k, we can use the Elbow Method. This method identifies the optimal number of clusters by plotting the distortion against k. The 'elbow' point, where the distortion decreases at a slower rate, indicates the best k. From the graph, around $k = 12$ represents the start of smaller decreases with increases in the number of clusters. The reason for using the Elbow Method is that a higher k reduces distortion but adds complexity and risks overfitting. In simple terms, as k increases, each cluster can represent fewer data points, eventually leading to $k = n$, where every data point becomes its own cluster, and the distortion becomes zero.





Note

- Your code must run across all k values and all images for the L2 norm in less than 10 minutes total. This includes any result generation, but excludes re-iterations due to testing out random seeds. Hint: Vectorization will make this requirement trivial.
- Your code must run and return all results without any form of modification or TA input.
- 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. Think about what this means in the context of the image compression problem, what makes up and image and why might a cluster be empty.
- Setting a random seed will ensure reproducability and is good best practice.
- Ensure that you are using an appropriate stopping criteria in accordance with the k-means algorithm presented in the lecture. Placing a cap on iterations as your criteria is not valid.
- 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 Political blogs dataset [25 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. (15 points) Use spectral clustering to find the $k = 2, 5, 10, 30, 50$ 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 $k = 2, 5, 10, 30, 50$. 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.

k = 2
Mismatch Rate = 0.31425000000000003

k = 5
Mismatch Rate = 0.16234

k = 10
Mismatch Rate = 0.12419

k = 30
Mismatch Rate = 0.08554

k = 50
Mismatch Rate = 0.07663199999999999

2. (10 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.

To tune k, I analyzed the trade-off between the average mismatch rate and the spread of mismatch rates across clusters. The goal was to achieve a reasonably low average mismatch rate while maintaining a balanced spread, avoiding extreme variations between clusters. After evaluating k from 2 to 9, $k = 5$ was selected as the best choice. It had the lowest average mismatch rate (0.07592) and a relatively small spread (0.1136), indicating consistent performance across clusters. Larger k values led to increasing mismatch rates and wider spreads, suggesting over-segmentation, while smaller k values had higher average mismatch rates, implying insufficient segmentation. This result reflects that the network structure is well-represented with five communities, where most nodes within a cluster share the same political orientation.

5 MNIST Dataset clustering [Bonus: 10 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. (5 points) Use the Euclidean distance as a metric for clustering (you may either base it on the code you had for Question 3 and make necessary changes, or use a kmeans package.) Report the *purity* score for each cluster.

Purity scores:

- Cluster 0: 0.5281
- Cluster 1: 0.6448
- Cluster 2: 0.4410
- Cluster 3: 0.8688
- Cluster 4: 0.3193
- Cluster 5: 0.6077
- Cluster 6: 0.9153
- Cluster 7: 0.6625
- Cluster 8: 0.3569
- Cluster 9: 0.9389

2. (5 points) Now we perform a simple pre-processing: threshold the pixels to convert the image into binary-valued-pixel: if the pixel values are above 128, they are assigned as "1" and otherwise (below 128) they are assigned as "0". After thresholding, try your k-means with Hamming distance on the binary-valued-pixel images and repeat the same steps in Part (1). Please note that the assignment of data points should be based on the Hamming distance (for the binary-valued-pixel images), and the cluster centroid will be taken as the per-pixel "majority" image of each cluster. Report the *purity* score for each cluster. Comment on which metric gives the better result?

Purity scores:

- Cluster 0: 0.4261
- Cluster 1: 0.4715
- Cluster 2: 0.4868
- Cluster 3: 0.5854
- Cluster 4: 0.3559
- Cluster 5: 0.9175
- Cluster 6: 0.7835
- Cluster 7: 0.4411
- Cluster 8: 0.4418
- Cluster 9: 0.3950

The Euclidean distance metric gives better results than Hamming distance, as shown by the higher average purity score (0.6283 vs. 0.5305). Euclidean distance captures the continuous nature of pixel intensities in the MNIST dataset, preserving finer details and yielding more accurate clustering. In contrast, Hamming distance oversimplifies the data by converting it to binary, leading to information loss and reduced performance.