Homework 1

Xiaofan Jiao

Q 1.1

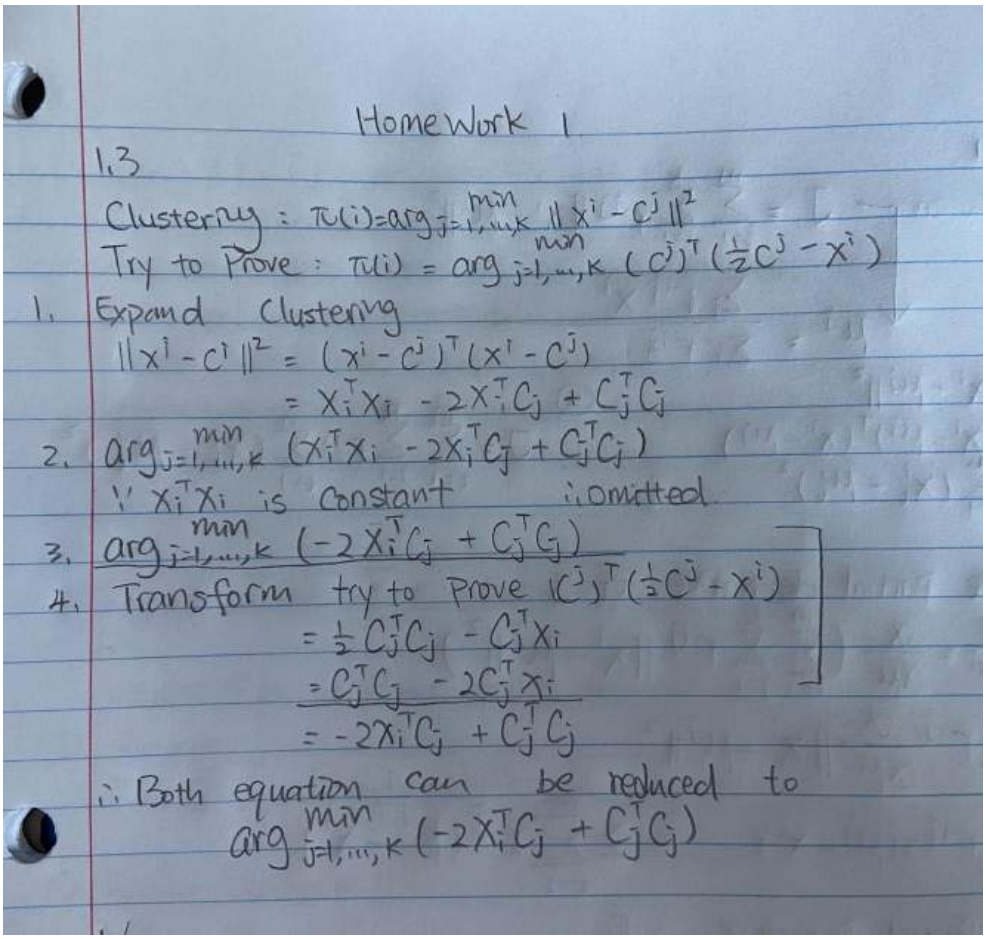
Supervised learning allows models to be trained to predict labels for new data points by using a labeled dataset (X) and matching labels (y). This method is very goal-oriented and makes evaluation easier because it has established labels that predictions can be compared to. Nevertheless, a major disadvantage is the requirement for large volumes of labeled data, which can be expensive and time-consuming to get. On the other hand, unsupervised learning works only with dataset X and does not use any labels; instead, it looks for patterns that are already present in the data. This makes it flexible in processing more abundant, unlabeled data and makes it possible to use datasets without labels, but it also makes evaluating the model's performance more difficult because there are no predetermined labels to confirm the patterns that are found. As such, while unsupervised learning can uncover new information and intricate patterns in data, the absence of precise assessment criteria may cause these discoveries to be unclear and more difficult to understand.

Q 1.2

We may use a mix of Euclidean distance for numerical attributes and Hamming distance for category attributes to build a similarity function d(xi, xj) for a dataset that contains both types of features. In particular, it is best to one-hot encode categorical variables first, including city names and housing types, so that they may be compared using the Hamming distance, which counts the amount of bits that differ in these binary vectors. The Euclidean distance, which calculates the straight-line distance between values, is suitable for numerical features like price.

The overall similarity function can be represented as $d(x_i, x_j) = \alpha \cdot Hamming(x_i^{cat}, x_j^{cat}) + \beta \cdot Euclidean(x_i^{num}, x_j^{num})$, where α and β are weights that balance the influence of categorical versus numerical differences.

Q 1.3



Q 1.4

Because the k-means clustering method might become stuck in local minima, different initializations of the centroids produce different results. Depending on where these centroids are initially put, the k-means algorithm may converge to different local optima as it iteratively modifies them to minimize the within-cluster sum of squares. This occurs due to the fact that the initial positioning of centroids affects which data points are originally deemed to be closer to each centroid, which may result in noticeably different clustering results. As a result, if initial centroids are close to outliers or sparse regions, clusters generated in each iteration may differ in size, shape, and the data points they contain. In order to combat this, k-means is frequently run several times with various initializations in an effort to identify a clustering solution that is more optimal.

Q 1.5

The k-means clustering algorithm is guaranteed to stop after a finite number of iterations because it operates in a finite, discrete space where each point can only be assigned to one of the k clusters, and the total number of possible configurations is finite. Moreover, k-means optimizes a non-increasing objective function, which is the sum of squared distances from each point to its nearest centroid. Each iteration recalculates centroids as

the mean of points assigned to them and reassigns points to the closest centroid, thereby reducing or maintaining the objective function but never increasing it. This methodical reduction and finite state space ensure that the algorithm will converge to a stable set of clusters, where point assignments no longer change, forcing the algorithm to terminate after a limited number of steps.

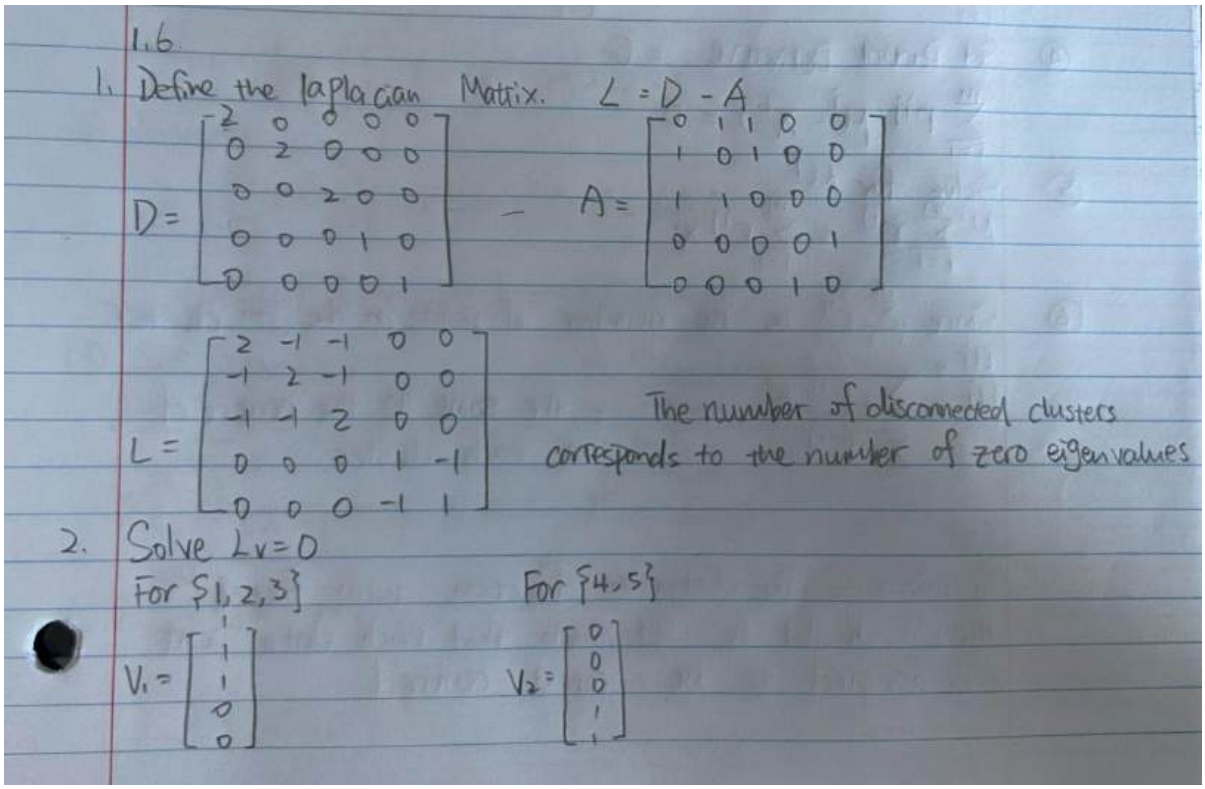
Q 1.5

```python
import numpy as np

A = np.array([
    [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]
])

# Compute the degree matrix D
D = np.diag(np.sum(A, axis=1))

L = D - A

# Perform eigenvalue decomposition on the Laplacian matrix
eigenvalues, eigenvectors = np.linalg.eig(L)

# Find indices of eigenvalues that are approximately zero
zero_eigen_indices = [index for index, value in enumerate(np.around(eigenvalues, 1)) if value == 0]

print("Eigenvalues:", np.around(eigenvalues, 1))
print("Indices of zero eigenvalues:", zero_eigen_indices)
print("Eigenvectors:\n", eigenvectors)
print("Cluster Assignments:\n", eigenvectors[:, zero_eigen_indices])
```

[2] ✓ 0.0s

```
Eigenvalues: [ 3. -0.  3.  2.  0.]
Indices of zero eigenvalues: [1, 4]
Eigenvectors:
 [[ 0.81649658 -0.57735027  0.30959441  0.          0.        ]
 [-0.40824829 -0.57735027 -0.80910101  0.          0.        ]
 [-0.40824829 -0.57735027  0.49950661  0.          0.        ]
 [ 0.          0.          0.          0.70710678  0.70710678]
 [ 0.          0.          0.         -0.70710678  0.70710678]]
Cluster Assignments:
 [[-0.57735027  0.        ]
 [-0.57735027  0.        ]
 [-0.57735027  0.        ]
 [ 0.          0.70710678]
 [ 0.          0.70710678]]
```

Q 1.6


1.6.
1. Define the laplacian Matrix. L = D - A
D = [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]
A = [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]
L = [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]
The number of disconnected clusters corresponds to the number of zero eigenvalues
2. Solve Lv = 0
For {1, 2, 3}
For {4, 5}
V1 = [1; 1; 1; 0; 0]
V2 = [0; 0; 0; 1; 1]

Q 2.1

2.1

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

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

① $\|x^i - \mu^j\|^2$
$= (x^i - \mu^j)^T (x^i - \mu^j)$
$= \sum_{l=1}^{n} (x_l^i - \mu_l^j)^2$

③ $J = \sum_{i=1}^{m}\sum_{j=1}^{k} r^{ij} \sum_{l=1}^{n} (x_l^i - \mu_l^j)^2$

③ take partial derivative of J

$$\frac{dJ}{d\mu^j} = \frac{\alpha}{\alpha\mu^j}\left(\sum_{i=1}^{m}\sum_{j=1}^{k} r^{ij}\|x^i - \mu^j\|^2\right)$$

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

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

④ Set Partial Derivative $= 0$

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

⑤ Solve for μ^j

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

⑥ Since $\sum_{i=1}^{m} r^{ij}$ is the number of points in the j^{th} cluster

$$\mu^j = \frac{\sum_i r^{ij} x^i}{\sum_i r^{ij}}$$ = the same as the centroid of each cluster

Q 2.2

2.2.
To minimize the distortion function with fixed μ^j the r^{ij} should be set such that each data point x^i is assigned to the nearest centroid

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

For each data point x^i we calculate the squared Euclidean distance to all centroids and assign the point to the cluster with the closest centroid. This way we ensure the contribution of the distortion function from each data point is minimized, thus minimize the overall distortion function J

Q 2.3

2.3.

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

where Σ is a fixed positive definite matrix

① Rewrite

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

② Partial Derivative

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

③ Simplify

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

$\frac{\partial J}{\partial \mu^j} ((x^i - \mu^j)^T \Sigma (x^i - \mu^j)) = -2\Sigma (x^i - \mu^j)$

④ Set Derivative = 0

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

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

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

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

⑤ Solve for μ^j

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

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

⑥ Solve for r^{ij}

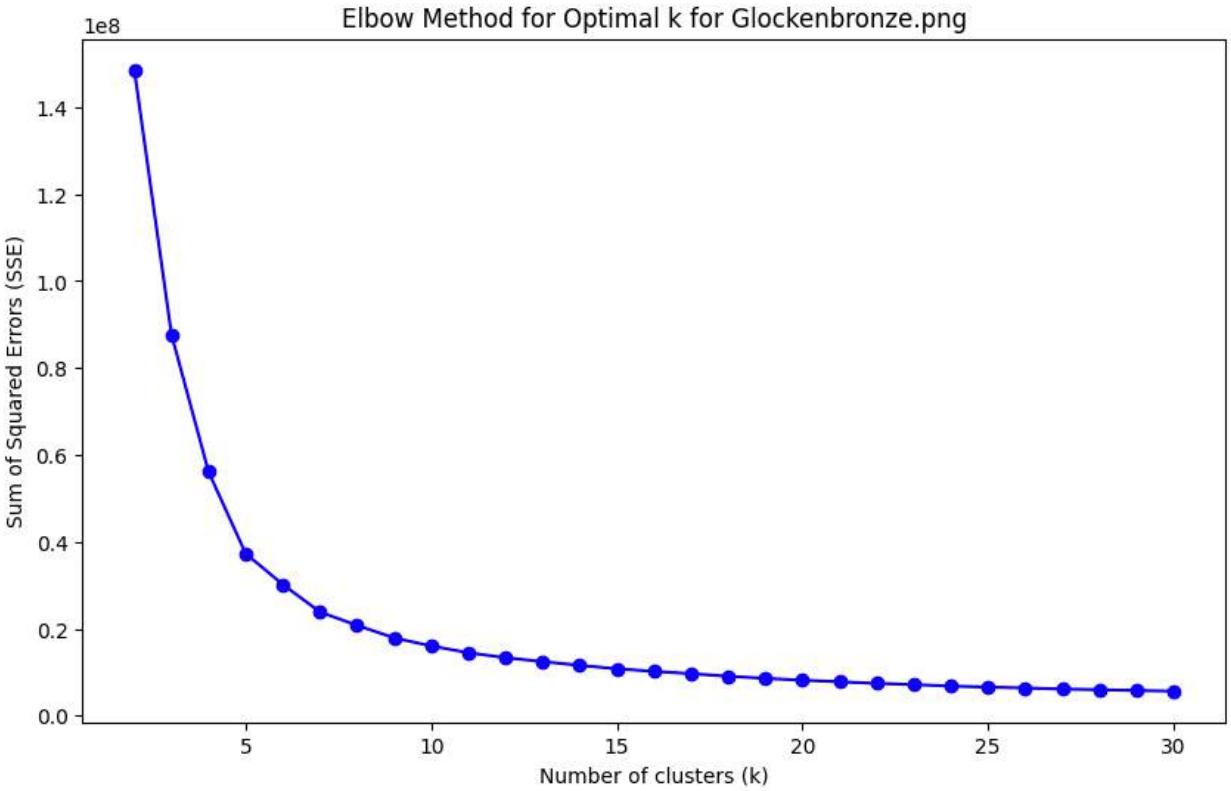
$r^{ij} = \begin{cases} 1 & \text{if } j = \arg\min_j (x^i - \mu^j)^T \Sigma (x^i - \mu^j), \\ 0 & \text{otherwise.} \end{cases}$

Q 3.1 & 3.2


Original Image
k=2
Compressed Image (L2)
k=2
Compressed Image (L1)
k=2
Original Image
k=5
Compressed Image (L2)
k=5
Compressed Image (L1)
k=5
Original Image
k=10
Compressed Image (L2)
k=10
Compressed Image (L1)
k=10
Original Image
k=20
Compressed Image (L2)
k=20
Compressed Image (L1)
k=20
Original Image
k=30
Compressed Image (L2)
k=30
Compressed Image (L1)
k=30

```
Results for Glockenbronze.png:
k     Iterations (L2) Run Time (L2) (s)    Iterations (L1) Run Time (L1) (s)
2     10              0.10                 6               0.05
5     25              0.47                 22              0.41
10    74              2.54                 29              0.96
20    100             6.11                 47              2.77
30    100             8.97                 100             8.73

Cluster Assignments for k=2 (L2 norm):
[1 1 1 ... 0 0 0]

Centroids for k=2 (L2 norm):
[[136.34139541  96.61381691  85.35801376]
 [207.26052707 179.44939732 157.70071815]]

Cluster Assignments for k=2 (L1 norm):
[1 1 1 ... 0 0 0]

Centroids for k=2 (L1 norm):
[[136.3734961   96.74156162  85.46517583]
 [207.45750929 179.54969684 157.79464144]]

Cluster Assignments for k=5 (L2 norm):
[1 4 4 ... 0 0 0]
...
 [229.04724097 205.53473601 182.6605796 ]]
```

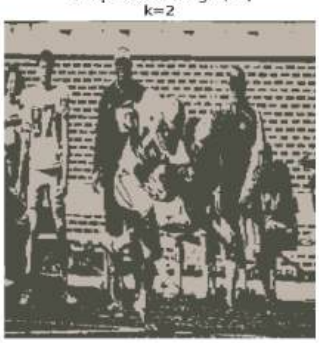
I would use the Elbow method to find the optimal k, for the Glockenbronze.png is 7



Original Image
k=2

Compressed Image (L2)
k=2

Compressed Image (L1)
k=2



Original Image
k=5

Compressed Image (L2)
k=5

Compressed Image (L1)
k=5

Original Image
k=10

Compressed Image (L2)
k=10

Compressed Image (L1)
k=10

Original Image
k=20

Compressed Image (L2)
k=20

Compressed Image (L1)
k=20

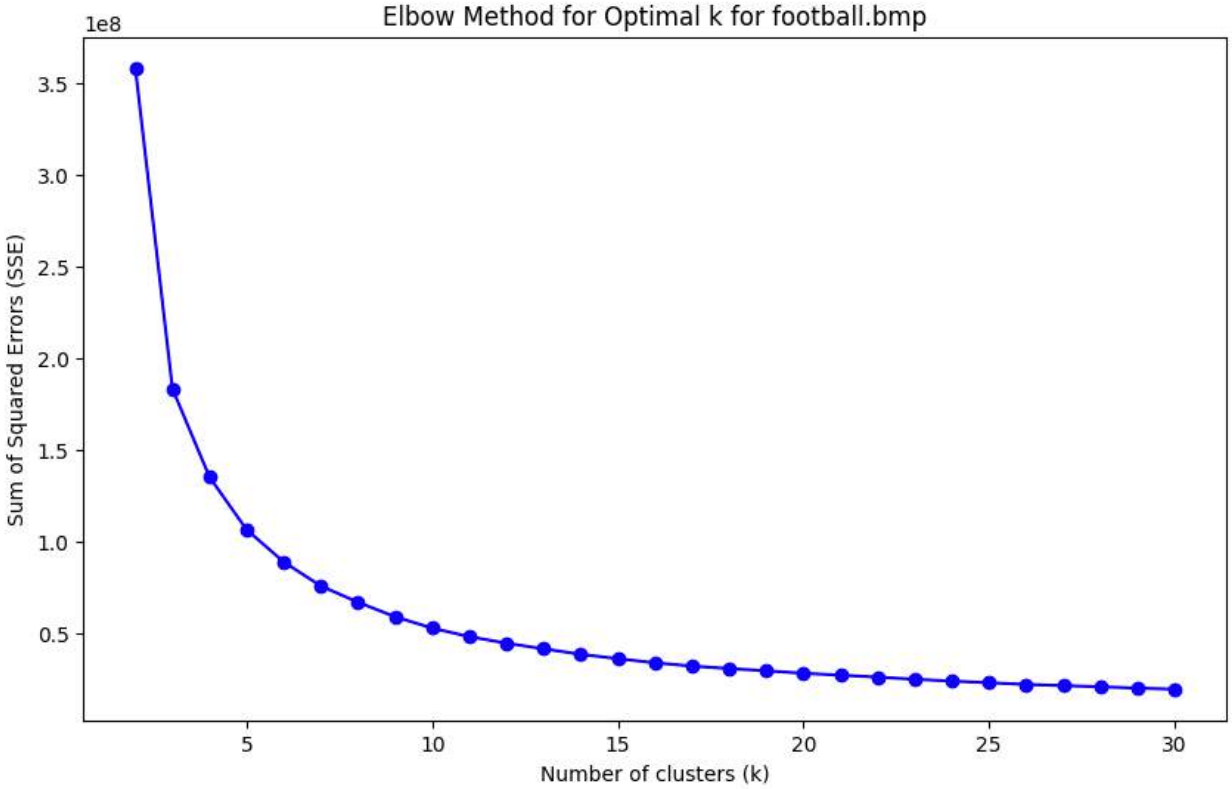
Original Image
k=30

Compressed Image (L2)
k=30

Compressed Image (L1)
k=30

```
Results for football.bmp:
k     Iterations (L2) Run Time (L2) (s)    Iterations (L1) Run Time (L1) (s)
2     21              0.21                 10              0.19
5     32              0.89                 17              0.37
10    65              2.29                 36              1.22
20    100             6.16                 56              3.82
30    100             9.69                 81              7.25

Cluster Assignments for k=2 (L2 norm):
[0 0 0 ... 0 0 0]

Centroids for k=2 (L2 norm):
[[ 76.43129869  78.35079428  68.35474815]
 [188.45300526 180.4567343  170.73523798]]

Cluster Assignments for k=2 (L1 norm):
[0 0 0 ... 0 0 0]

Centroids for k=2 (L1 norm):
[[ 77.14706087  78.68557239  68.58063169]
 [188.69450295 181.08288588 171.50342638]]

Cluster Assignments for k=5 (L2 norm):
[1 1 1 ... 3 3 3]
...
 [192.0716946  180.59683426 107.5679702 ]]
```

I would use the Elbow method to find the optimal k, for the football.bmp is 6



Original Image
k=2

Compressed Image (L2)
k=2

Compressed Image (L1)
k=2

Original Image
k=5

Compressed Image (L2)
k=5

Compressed Image (L1)
k=5

Original Image
k=10

Compressed Image (L2)
k=10

Compressed Image (L1)
k=10

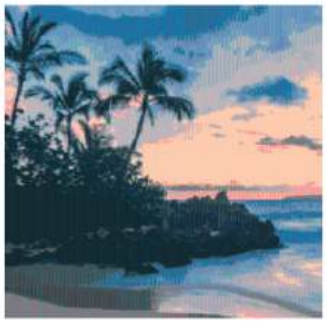
Original Image
k=20

Compressed Image (L2)
k=20

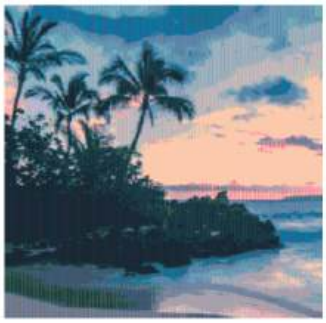
Compressed Image (L1)
k=20

Original Image
k=30

Compressed Image (L2)
k=30

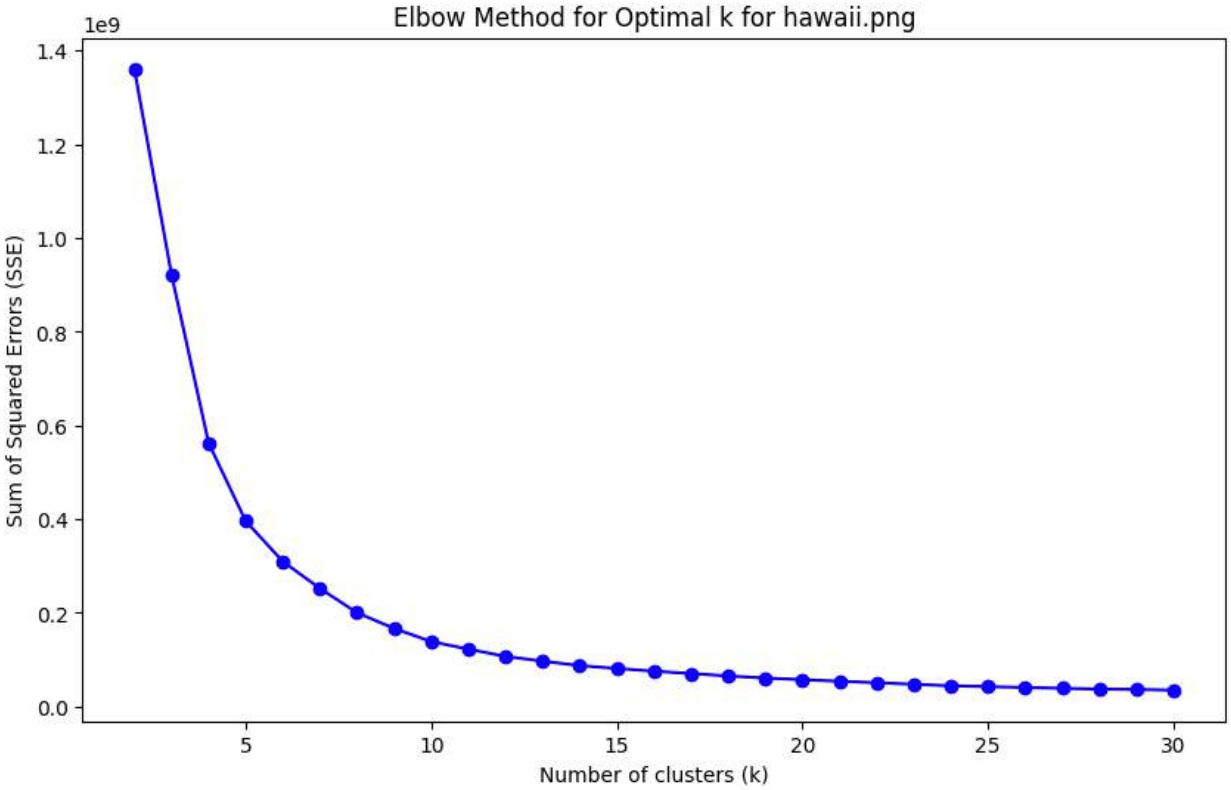


Compressed Image (L1)
k=30



```
Results for hawaii.png:
k     Iterations (L2) Run Time (L2) (s)    Iterations (L1) Run Time (L1) (s)
2     19              0.23                 9               0.11
5     27              0.66                 11              0.59
10    46              2.35                 26              1.17
20    90              8.19                 39              3.52
30    90              13.53                61              7.43

Cluster Assignments for k=2 (L2 norm):
[1 1 1 ... 0 1 1]

Centroids for k=2 (L2 norm):
[[136.0555875  125.20680013  71.07625967]
 [169.04941063 175.51623113 207.86676749]]

Cluster Assignments for k=2 (L1 norm):
[0 0 0 ... 1 0 1]

Centroids for k=2 (L1 norm):
[[170.63274603 212.59586445 171.63598354]
 [135.82930643  72.80658637 134.40148535]]

Cluster Assignments for k=5 (L2 norm):
[3 4 2 ... 4 1 3]
...
 [ 98.09988318 131.91588785 167.54964953]]
```

I would use the Elbow method to find the optimal k, for the hawaii.png is 6



Overall, we can see that as K increases the time to converge will also increase with the number of iterations. We can clearly see that K = 30 has the best image quality for all three images.

Q 4.1 & 4.2

Purity Scores using L2 norm:

Cluster	Purity Score (L2)
1	0.824901
2	0.896464
3	0.801125
4	0.553378
5	0.639281
6	0.932086
7	0.420891
8	0.524377
9	0.337827
10	0.748146

Purity Scores using L1 norm:

Cluster	Purity Score (L1)
1	0.585374
2	0.324989
3	0.935761
4	0.548286
5	0.475508
6	0.766276
7	0.529022
8	0.199584
9	0.690597
10	0.487174

We may draw a number of conclusions from the outcomes of K-means clustering using both the L2 (Euclidean) and L1 (Manhattan) norms. Based on the metric of total dataset purity, the L2 norm seems to perform better throughout the entire dataset. In particular, when comparing clusters with the L2 norm to those with the L1 norm, the latter had higher overall purity scores. This implies that the L2 norm is more successful in forming clusters that closely resemble the dataset's genuine labels.

On the other hand, it is important to note that more highly pure (≥ 80 percent) clusters are produced using the L1 norm (K-medians). This suggests that the L1 norm can handle some similar numbers better, leading to extremely pure clusters. On the other hand, more clusters with lesser purity (≤ 50%) are likely to be produced by the L2 norm. In spite of this, the L1 norm's overall performance is not as good as the L2 norm since the L1 norm's

very pure clusters do not account for as much of the total dataset. Therefore, while taking into account the purity of the entire dataset, the L2 norm performs better for this example.

Q 5.1
K = 2, 5, 10, 30, 50

	k	Cluster Index	Cluster Size	Voted Label	Cluster Mismatch Rate	Overall Mismatch Rate
0	2	1	784	1	0.449	nan
1	2	2	440	0	0.4636	nan
2	k=2					0.4563
3	5	3	400	0	0.0775	nan
4	5	2	206	1	0.1311	nan
5	5	5	187	0	0.0856	nan
6	5	1	138	1	0.1159	nan
7	5	4	293	1	0.0171	nan
8	k=5					0.08544
9	10	1	139	0	0.0576	nan
10	10	10	131	1	0.3817	nan
11	10	3	161	0	0.0683	nan
12	10	6	99	1	0.1313	nan
13	10	8	170	0	0.0647	nan
14	10	5	91	1	0.2637	nan
15	10	7	41	0	0.0244	nan
16	10	2	114	1	0.0263	nan
17	10	4	93	1	0.1183	nan
18	10	9	185	1	0.0378	nan
19	k=10					0.11741

46	30	3	65	1	0.0154	nan
47	30	11	46	1	0.0	nan
48	30	14	39	1	0.0	nan
49	30	5	20	1	0.0	nan
50	k=30					0.0982133
51	50	9	58	0	0.0345	nan
52	50	37	53	0	0.0	nan
53	50	35	10	0	0.2	nan
54	50	41	22	0	0.3636	nan
55	50	18	22	0	0.0	nan
56	50	4	24	0	0.125	nan
57	50	38	29	0	0.1034	nan
58	50	24	19	1	0.1053	nan
59	50	15	21	0	0.0476	nan
60	50	32	34	0	0.0	nan
61	50	23	32	0	0.0938	nan
62	50	50	31	0	0.129	nan
63	50	30	9	1	0.3333	nan
64	50	47	26	0	0.0769	nan
65	50	6	27	0	0.1852	nan
66	50	3	24	0	0.0417	nan
67	50	19	16	0	0.125	nan
68	50	25	25	0	0.12	nan
69	50	16	9	0	0.2222	nan
70	50	43	18	0	0.0	nan
71	50	27	11	0	0.0	nan
72	50	14	21	0	0.0	nan
73	50	1	40	0	0.4	nan

20	30	2	59	0	0.0508	nan
21	30	4	56	0	0.0536	nan
22	30	22	35	1	0.2286	nan
23	30	8	66	0	0.0909	nan
24	30	7	32	0	0.0625	nan
25	30	26	63	0	0.0317	nan
26	30	10	37	0	0.027	nan
27	30	12	54	1	0.1296	nan
28	30	30	21	0	0.0476	nan
29	30	20	31	0	0.0	nan
30	30	17	44	0	0.1364	nan
31	30	13	44	0	0.1136	nan
32	30	27	34	0	0.1176	nan
33	30	29	27	1	0.1111	nan
34	30	23	18	0	0.0	nan
35	30	16	26	0	0.3077	nan
36	30	19	23	0	0.1304	nan
37	30	1	51	0	0.4706	nan
38	30	18	31	1	0.3226	nan
39	30	28	40	1	0.025	nan
40	30	25	28	1	0.0714	nan
41	30	15	47	1	0.1489	nan
42	30	24	61	1	0.082	nan
43	30	21	41	1	0.0244	nan
44	30	9	34	1	0.0882	nan
45	30	6	51	1	0.0588	nan
46	30	3	65	1	0.0154	nan
47	30	11	46	1	0.0	nan

74	50	26	18	0	0.0556	nan
75	50	29	14	0	0.2143	nan
76	50	17	31	1	0.0968	nan
77	50	12	22	1	0.3636	nan
78	50	46	20	0	0.5	nan
79	50	36	43	1	0.0465	nan
80	50	48	13	0	0.0769	nan
81	50	10	26	1	0.0769	nan
82	50	49	17	1	0.0588	nan
83	50	21	10	1	0.3	nan
84	50	45	37	1	0.027	nan
85	50	44	28	1	0.1786	nan
86	50	13	27	1	0.1481	nan
87	50	28	26	1	0.0385	nan
88	50	39	19	1	0.2105	nan
89	50	2	32	1	0.0312	nan
90	50	7	23	1	0.0435	nan
91	50	22	53	1	0.0189	nan
92	50	5	20	1	0.0	nan
93	50	8	25	1	0.0	nan
94	50	42	15	1	0.0	nan
95	50	20	33	1	0.0	nan
96	50	40	24	1	0.0	nan
97	50	34	23	1	0.0	nan
98	50	11	14	1	0.0	nan
99	50	31	18	1	0.0	nan
100	50	33	12	1	0.0	nan
101	k=50					0.103844

Q 5.2

To determine the optimal number of clusters for the network, I experimented with different values of k and analyzed the resulting mismatch rates. My goal was to achieve a reasonably small mismatch rate while maintaining meaningful clusters. I tested k values of 2, 5, 10, 25, 30 and 50.

For $k = 2$, the average mismatch rate was 0.4563 with a spread of 0.0146. Although the mismatch rate was relatively high, the small spread indicated consistency between clusters. Increasing k to 5 reduced the average mismatch rate to 0.08544 with a spread of 0.114, suggesting better cluster separation. When k was set to 10, the average mismatch rate slightly increased to 0.13546 with a larger spread of 0.3389. Higher values of k like 25 and 50 resulted in mismatch rates around 0.0915 with variable spreads.

Based on these results, I would select $k = 5$ as the optimal number of clusters. This choice provided a low average mismatch rate of 0.08544 and a reasonable spread, indicating effective clustering. The small mismatch rate indicates that the majority of nodes within each cluster share the same orientation, highlighting the presence of well-defined communities within the network.