K-Means Image Compression Findings

Exercise and K-Means Overview

This exercise uses the K-means algorithm for image compression, where the K-means algorithm is implemented in the kmeans.py file, rather than calling a package. The K-means algorithm follows the below generalized implementation:

- Given m data points, $\{x^1, x^2, .. x^m\} \in R^n$
- Find k cluster centers, $\{c^1, c^2, .. c^k\} \in R^n$
- And assign each data point i to a single cluster, $\pi(i) \in \{1,..,k\}$
- Such that a dissimilarity measure is minimized
    - For k-means, the Euclidean distance is commonly used as the dissimilarity measure.

Each iteration applies the two step expectation-maximization (or in this case, minimization) strategy, where with each pass,

- For each center, a subset of training points are identified, where that center is closer to each respective point than any other center
- For each cluster, the means of each feature for the data points are computed, and this mean vector becomes the new center for that cluster.

These two steps are repeated until convergence, or when the objective function has reached its local optimum.

File Descriptions

Run.py

Run.py starts by reading in an image, where each data point has three values corresponding to the RGB pixel value. The K-means function is imported from Kmeans.py and is used to return the five images with varying *k* number of clusters: [2,4,8,16,32]. The number of clusters corresponds to the number of colors in the resulting image. Kmeans_export.pdf and Result.txt are exported from run.py, which is called with the command, *python run.py [image path]*.

Kmeans.py

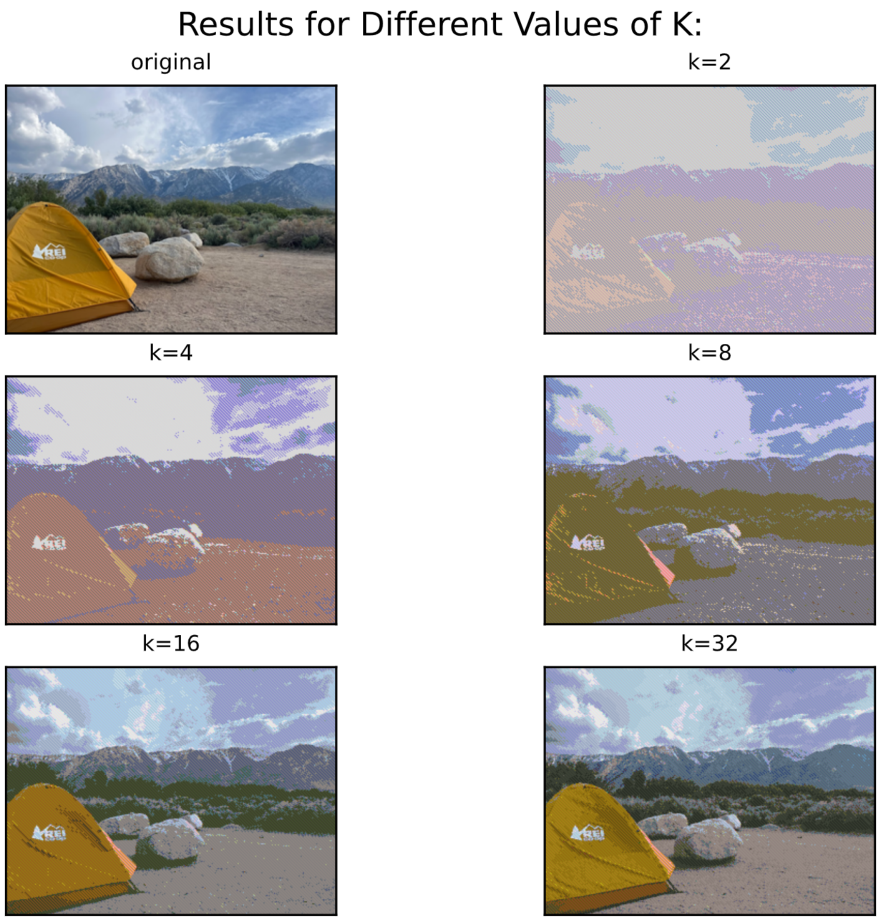
Kmeans.py implements the K-means algorithm, described in *Exercise and K-Means Overview*. The Kmeans function requires the below three parameters:

- Data: The image in a numpy array, where each data point is a subarray with three values corresponding to the RGB pixel value.
- K: The number of cluster centers, or unique RGB values
- Ct_Init: Randomly initialized centers

Kmeans_export.pdf

Kmeans_export.pdf shows the resulting images for each value of *k* cluster centers.

Figure 1



Results.txt

Result.txt gives the run time associated with each cluster, as well as the number of empty clusters.

Table 1

k	**Runtime**	**# of Empty Clusters**
2	0.13s	0
4	0.20s	0
8	0.34s	0
16	0.37s	0
32	1.92s	4

Conclusions

As seen in *Figure 1*,

- As K increases, smaller variations are captured. We can see that with increasing the number of k, the number of colors also increases. The image becomes increasingly similar to the original photo as the granularity is increased with more colors (or K cluster centers).

As seen in *Table 1*,

- As K increases, the time to convergence also increases. Increasing K increases the number of centroids to calculate and update in each iteration, which can lead to higher computational complexity and slower convergence times.
- Different centroid initializations can lead to different results.When 32 randomly assigned cluster assignments were given to the K-means algorithm, some of the initial centers were not the optimal center for any data point. This could be further confirmed by more runs.