1 Clustering

1. To minimize the distortion function with respect to μ^j, all we need is to take the first derivative of the distortion function and set it to zero. Therefore:

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

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

Setting above equation equal to zero results in:

$$\frac{\partial J}{\partial \mu^j} = 0 \quad \rightarrow \boxed{\mu^j = \frac{\sum_i r^{ij} x^i}{\sum_i r^{ij}}}$$

2. Similar to what we did above, we need to take the first derivative of the distortion function and set it to zero to be able to find the centroids.

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

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

Therefore:

$$\frac{\partial J}{\partial \mu^j} = 0 \quad \rightarrow \quad \mu^j = \frac{\sum_i r^{ij} x^i}{\sum_i r^{ij}}$$

In fact, it is easy to see how the Mahalanobis distance is a generalization of the Euclidean distance. The Euclidean distance D can be written as:

$$D = \left(x^i - \mu^j\right)^T \left(x^i - \mu^j\right) = \left(x^i - \mu^j\right)^T I^{-1} \left(x^i - \mu^j\right)$$

Hence, Euclidean function tacitly assumes that $\Sigma = I^{-1}$. By allowing the covariance matrix to take more general form, the clusters can take a larger variety of shapes.

Intuitively, we can think of the Mahalanobis distance from a point to its respective cluster center as its Euclidean distance divided by the square root of the variance in the direction of the point (i.e. in Mahalanobis distance formulation we have $r\Sigma$ as assignment function).

3. In each step of K-means algorithm, the distortion function J decreases until no change happens anymore. In fact, any change in the assignment of cluster index of the data points or adjusting center of each cluster, should result in a decrease in the loss function. Therefore, at each iteration of K-means algorithm, the loss function decreases.

   Since the number of assignments or center adjustments is limited, there should be a limited decrease in loss function and as a result, limited steps for K-means algorithm. The conclusion is that, K-means algorithm will finally converge to a local minimum value in finite steps.

4. Let's first use appropriate notation to formulate our problem. The cluster centers are located in $\mu^A = (-3,-1)$ and $\mu^B = (2,1)$. We also have five points which are located in the following positions:

   $$x^1 = (2,2), \quad x^2 = (-1,1), \quad x^3 = (3,1), \quad x^4 = (0,-1), \quad x^5 = (-2,-2)$$

   Now, we need to calculate the distance of each point from each cluster center and assign it to the appropriate cluster (in the following, D^{ij} shows the distance between point i and cluster j):

$$point\ 1 \begin{cases} D^{1A} = |2-(-3)| + |2-(-1)| = 8 \\ D^{1B} = |2-2| + |2-1| = 1 \\ \textbf{assign point 1 to cluster B} \end{cases}$$

$$point\ 2 \begin{cases} D^{2A} = |-1-(-3)| + |1-(-1)| = 4 \\ D^{2B} = |-1-2| + |1-1| = 3 \\ \textbf{assign point 2 to cluster B} \end{cases}$$

$$point\ 3 \begin{cases} D^{3A} = |3-(-3)| + |1-(-1)| = 8 \\ D^{3B} = |3-2| + |1-1| = 1 \\ \textbf{assign point 3 to cluster B} \end{cases}$$

$$point\ 4 \begin{cases} D^{4A} = |0-(-3)| + |-1-(-1)| = 3 \\ D^{4B} = |0-2| + |-1-1| = 4 \\ \textbf{assign point 4 to cluster A} \end{cases}$$

$$point\ 5 \begin{cases} D^{5A} = |-2-(-3)| + |-2-(-1)| = 2 \\ D^{5B} = |-2-2| + |-2-1| = 7 \\ \textbf{assign point 4 to cluster A} \end{cases}$$

Therefore, the new cluster centers will be:

$$\mu^A = \frac{(0,-1)+(-2,-2)}{2} = (-1, -{}^{3}/_{2})$$

$$\mu^B = \frac{(2,2)+(-1,1)+(3,1)}{3} = ({}^{4}/_{3}, {}^{4}/_{3})$$

Now, that we know the assignment of points as well as new cluster centers, we can determine whether the algorithm terminates at this step or it needs another step. Let's calculate the distances for point 2.

$$point\ 2 \begin{cases} D^{2A} = \ |-1-(-1)| + \left|1-\left(-{}^{3}/_{2}\right)\right| = \ {}^{5}/_{2} \\ D^{2B} = \left|-1-{}^{4}/_{3}\right| + \left|1-{}^{4}/_{3}\right| = \ 8/3 \\ \textbf{assign point 2 to cluster A} \end{cases}$$

Hence, **the algorithm <u>does not</u> terminate at one step** and it needs one more step to complete the algorithm.

2 Image compression using clustering

1. I tried Euclidean distance, Chebychev distance and Minkovski distance and Mahalanobis distance as my distance measures. Since Euclidean distance had the best performance, I decided to proceed with this measure. Therefore, in following all of the results are shown using Euclidean distance measure.

   According to the document provided by professor, using Euclidean distance gives us the opportunity to simplify the implementation of k-medoids algorithm. In this case, we only need to calculate the mean of the data points corresponding to each cluster and find the closest point in cluster to computed mean. The implementation of centroid computation could be summarized as following:

   I. Compute average of the current cluster.
   II. Find the pixel which has the closest distance to the computed average.
   III. Set the new centroid to this pixel.

   If we use other measures, we need to compute the pairwise distance for all pixels in cluster and determine the new centroid as the one with minimum overall distance from all pixels in that cluster. But again, since we are using Euclidean distance here, these two implementations will be equivalent.

   For stop criteria, I'm checking for the loss difference between current step and previous step. It means, at each step, I calculate the loss based on the dissimilarity function (here Euclidean distance). If the loss between this step and previous step is less than zero, the script keeps running. Otherwise, it means that the centroids and pixel assignments are not changing anymore, and we need to stop the script there. I also defined a maximum iteration for my code, which in case of big datasets and heavy computations could be useful. However, for the images used for the sake of this assignment, the code never satisfied the maximum iteration criterion.

2. The following shows the results of K-medoids algorithm as a function of number of clusters for different images. As can be seen, the quality of the images become better when we increase the number of clusters. However, the resulting image size become

greater. Also, it takes more time to run the algorithm with big number of clusters. Therefore, there should be a trade-off between running time, image size and image quality when compressing an image using K-medoids algorithm.

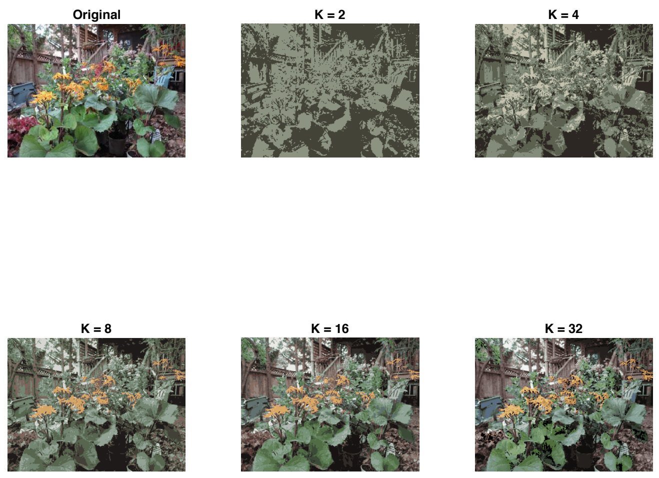

Figure 1 K-medoids algorithm with different number of clusters on Plants image

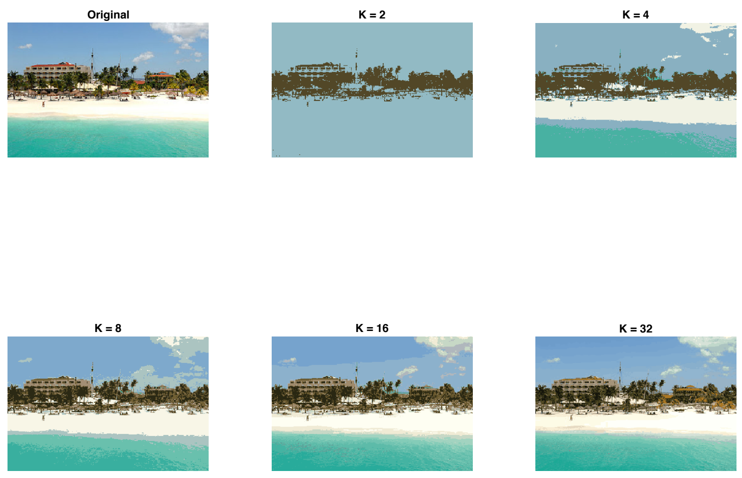

Figure 2 K-medoids algorithm with different number of clusters on Beach image

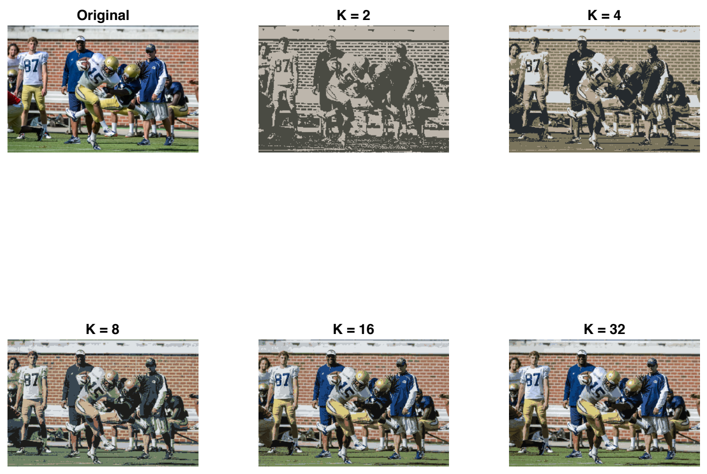

Figure 3 K-medoids algorithm with different number of clusters on Football image

Table 1 Running time (in seconds) associated with each image for different number of clusters used in K-medoids

	K = 2	**K = 4**	**K = 8**	**K = 16**	**K = 32**
Plants	0.023882	0.036879	0.099125	0.102727	0.401160
Beach	0.050090	0.023727	0.050416	0.078051	0.468560
Football	0.145761	0.135463	0.204195	0.617732	1.157132

3. Initial representation causes the algorithm merge into different local minima. That means K-medoids algorithm is not robust to the initialization. Depending on different starting points, the final result could differ. Following show the result of K-medoids algorithm on different setting of initial centroids for plant image.

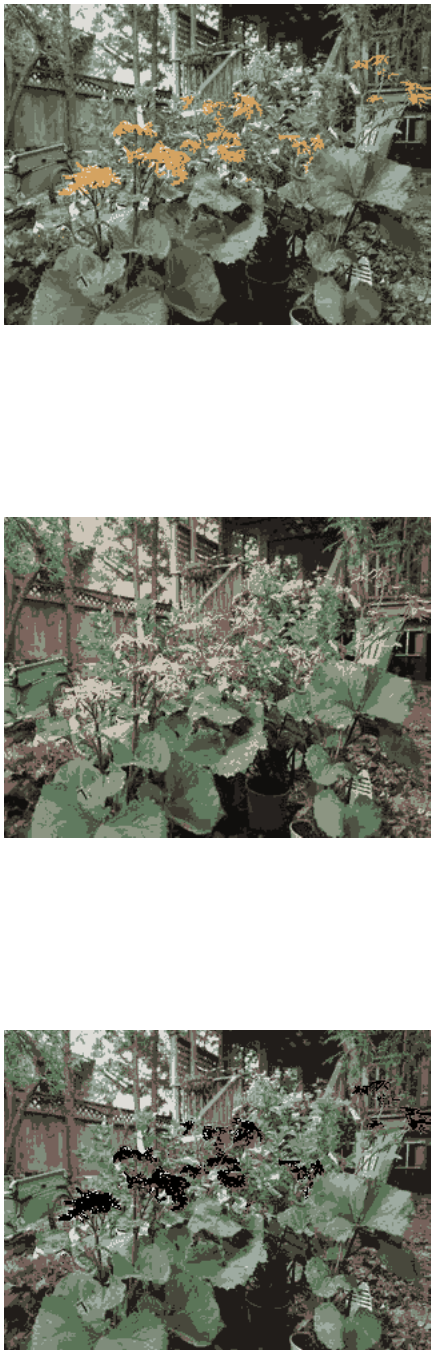

Figure 4 K-medoids algorithm with different setting of initial centroids

4. In terms of quality, both algorithms perform well, and they do not sound different with appropriate number of clusters. However, when small number of clusters are chosen, k-medoids perform slightly better than k-means implementation.

   In terms of robustness, two algorithms perform similarly. Depending on the initial setting of the centroids, the final result becomes different. But when it comes to the running time, k-medoids algorithm outperforms k-means algorithm. This is because k-medoids algorithm converges to the local optima faster than K-means algorithm. This is trivial that at each iteration, K-medoids take more running time with respect to K-means. But this does not necessarily result in smaller running time. Following show the results for Plants image using K-means implementation.

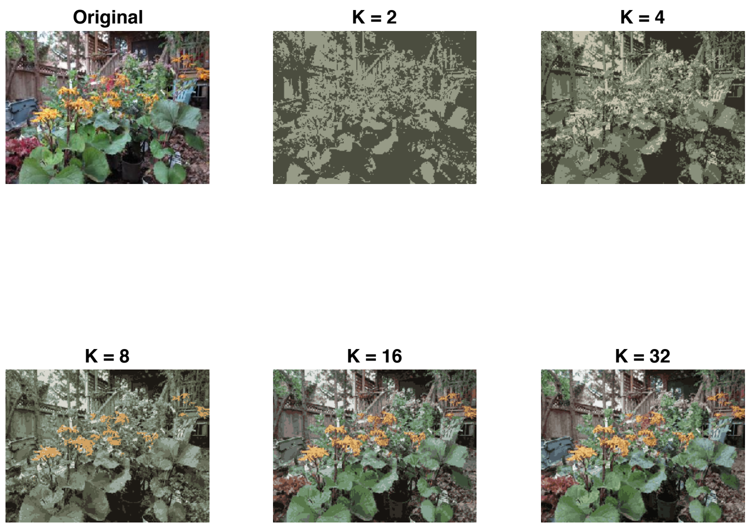

Figure 5 K-means algorithm with different number of clusters on Plants image

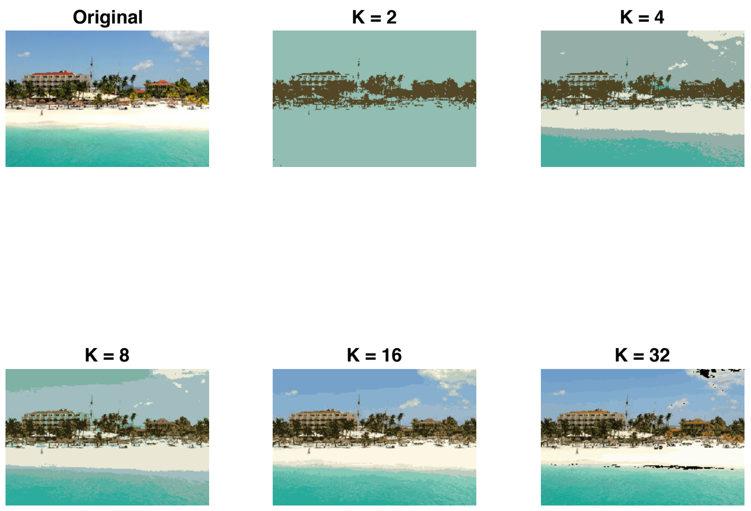

Figure 6 K-means algorithm with different number of clusters on Beach image

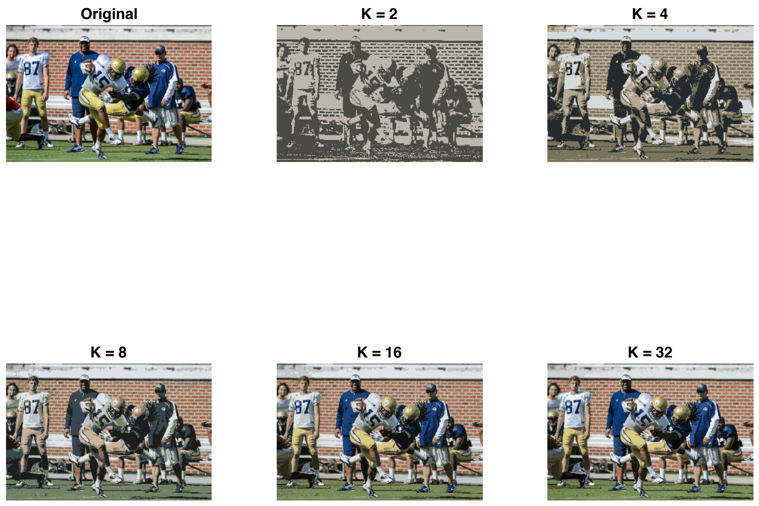

Figure 7 K-means algorithm with different number of clusters on Football image

Table 2 Running time (in seconds) associated with each image for different number of clusters used in K-means

	K = 2	K = 4	K = 8	K = 16	K = 32
Plants	0.383818	0.525053	0.487691	1.036416	0.899984
Beach	0.031887	0.074902	0.171648	0.258063	0.166050
Football	0.996651	0.950004	1.739546	2.891727	5.549763

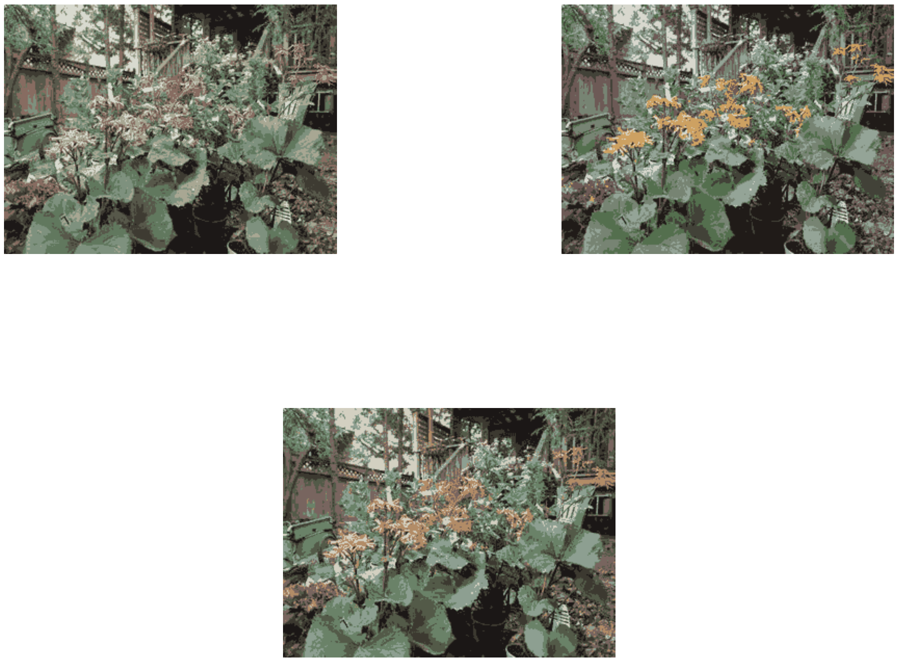

Figure 8 K-means algorithm with different setting of initial centroids