CS 550 -- Machine Learning

Homework #2

Due: 17:30, November 28, 2018

Sevil Çalışkan

21701423

Part 1

In Part 1, we are supposed to Implement the k-means algorithm to cluster the image pixels. My algorithm takes the pixels of an RGB image, group the pixels into given k clusters, and output the cluster centers as well as the labels of the clustered pixels as a map. My knn algorithm assign cluster centers randomly from the image pixels at first. Then calculates the distance of all pixels from those centers, finds the minimum distance center for each and assign them to it. Then recalculates the center this time by taking the mean of the pixels in the same cluster. Then calculates the distances again and same calculations goes on until there is no change in the centers.

To test my algorithm, I used the image (sample.jpg) provided on the course website. Following values of k = {2, 4, 8, 16} has been tested. For each of these k values, the clustering error, the values of the clustering vectors, and the clustered images are provided below.

Clustering error calculated by summing the squares of the distance between each pixel and its cluster centers. As will be seen, clustering error decreases, while k increases since a greater number of clusters means pixels will be closer to the center. In the best case, each pixel will be a cluster and error will be zero. Since this is not out aim, we should decide somehow to choose a k.

Sum of squared errors for k's:

	k = 2	k = 4	k = 8	k = 16
1.0e+08 *	1.1555	0.9042	0.6419	0.4807

Below center vector are provided for each k:

k=2

c1	214.2912	166.8082	196.048
c2	186.0064	82.2852	94.9659

k=4

c1	159.7911	119.9222	47.4135
c2	212.961	38.3653	123.1155
c3	224.1464	199.4823	217.9774
c4	202.0715	133.5111	172.7137

k=8

c1	22.0272	83.805	81.8307
c2	202.7907	96.2971	150.5548
c3	170.6271	166.2452	177.5321
c4	207.8415	114.5381	20.3432
c5	217.4396	186.8912	81.1172

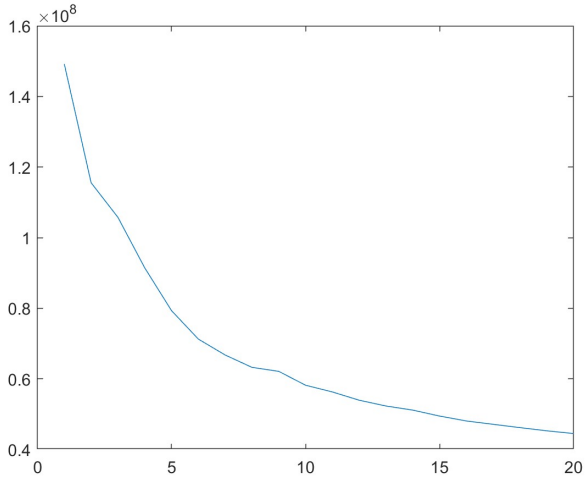
k=16

c1	178.1419	93.1916	119.2858
c2	11.8485	87.0449	89.429
c3	234.6384	81.7296	160.051
c4	241.9903	125.005	194.737
c5	228.8725	101.9691	18.1901
c6	242.1363	172.5855	225.0084
c7	185.2844	17.8086	102.3678
c8	220.9092	199.7706	114.6899
c9	131.4855	78.2317	13.1656
c10	237.8677	24.3978	126.4399
c11	186.2348	205.3853	205.3207
c12	217.4873	164.2585	34.3009
c13	239.459	230.662	236.5803
c14	147.4296	165.888	160.0082
c15	184.2783	119.4624	163.3496
c16	196.076	155.3634	190.1147

c6	230.1308	212.2703	227.1546
c7	211.7417	24.0091	115.5664
c8	237.4139	135.5989	200.4727

Below the clustered images for k={2,4,8,16} can be seen. (first row k=2 and k=4, second row k=8, k=16)

In order to decide an optimum k value, there are some methods recommended like silhouette method or elbow method. Since the image we work on is large and calculating silhouette values take some time, I will go with elbow method. I have clustered the image with k values from 2 to 20 and get the squared errors for each. Then, I plotted these errors, which is given below.

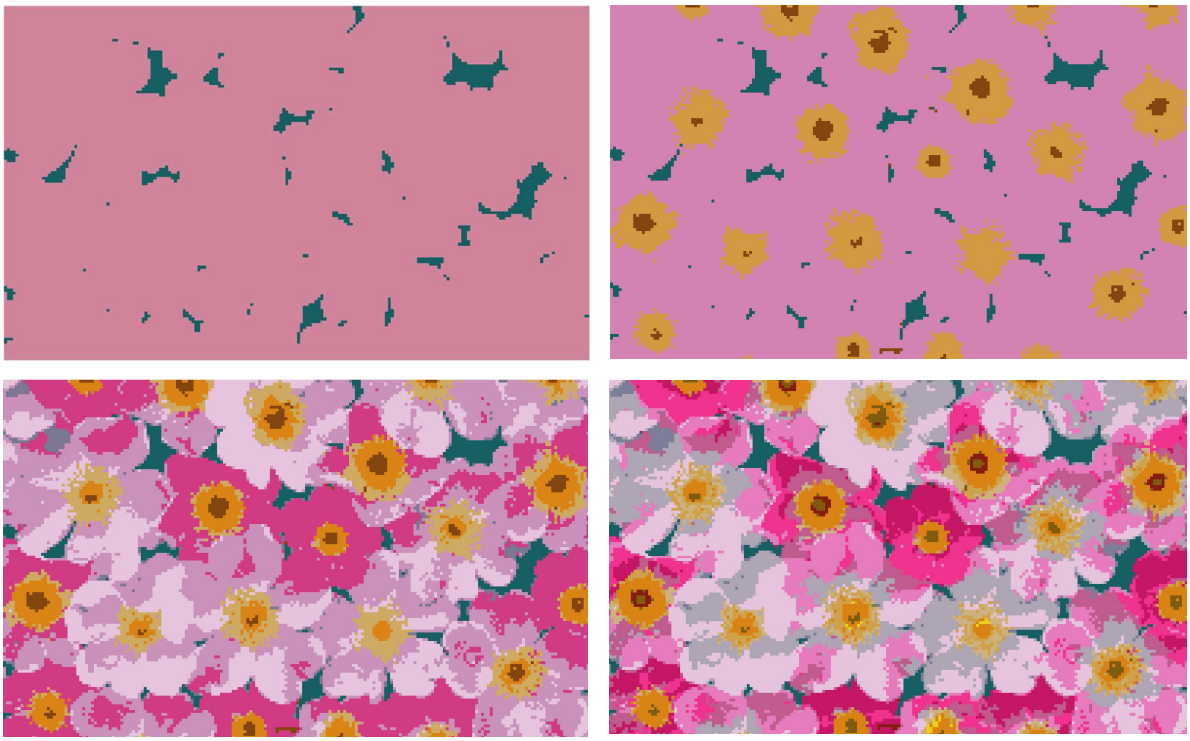


It seems that we can choose a k value between 5 and 10 since the speed of the decrease in error diminishes around those k values. k=6 or k=8 seems to be good values since for those k values, change in the speed is more observable. Since I have already calculated error rate and center vectors for k=8, I will go with it. The clustering error, the values of the clustering vectors, and the clustered image can be seen above.

Part 2

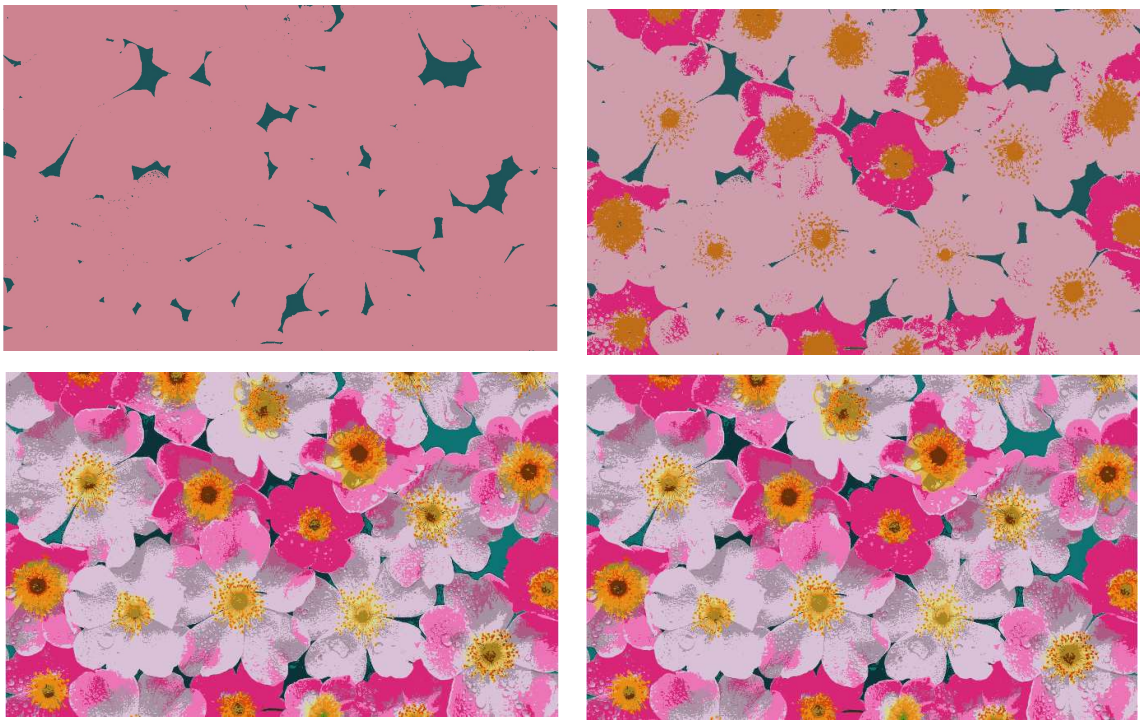
In second part, we are supposed to implement an agglomerative hierarchical clustering algorithm to cluster the image pixels. My algorithm takes the pixels of an RGB image, group the pixels into k clusters using my agglomerative algorithm, and output the clustering vectors as well as the labels of the clustered pixels. It starts with each pixel is being a distinct cluster. Then, distance between each cluster calculated, minimum distance is found. Clusters with this minimum distance are labelled as the minimum label number among them. Similarity of two clusters is decided by the distance of their centroids. So, centroid of the new cluster is calculated, distance between clusters are calculated and so on until the desired number of clusters k is reached.

The computational time of this part is high due to the number of pixels in the image. Thus, I tried to split image to small parts, 8*8 pixels in my case, with the assumption that closer pixels will be similar in terms of colors. Then means of these small parts is used as input to agglomerative algorithm. However, then I realized, my approach was the same as down sampling. Some of the results can be seen below.



The computational time of this part is high due to the number of pixels in the image. Thus, I tried to overcome this issue by splitting image into smaller parts (250*400 pixels in my case) and cluster them using knn algorithm (agglomerative algorithm can be used as well). For the knn part, k=100 seemed to be a good value, since 4*4*100 = 1600 pixels are fed into agglomerative algorithm and it takes a small time. After clustering “clusters” from knn algorithm, each pixel is labelled same as their cluster center label. Results of this approach was similar to knn results. Expectation was clusters from each part of

the picture will be helpful to reflect that part of the picture.. Results of the experiment for k values 2, 4, 8, 16 for agglomerative algorithm can be seen below.



Both knn and agglomerative algorithms gives good results for k = 8. For small values, knn seem to be performing better in terms of output while for larger values both algorithms works well.