CSE 6740: Computational Data Analysis Assignment #1

Due on Thursday, September 26, 2019

Shahrokh Shahi
(GT Account: sshahi3)

Q5. Programming: Image Compression

Answering to questions:

(1) Explain how you designed and implemented details of your K-medoids:

- How you chose representatives of each cluster

  We can pick various representatives. In this problem, to obtain a reasonable and also a [nearly] similar answer to K-means algorithm, in each iteration, I chose a point as the center which is the closest one to the mean of the associated cluster. In this approach, at each step, the mean value of the current cluster will be calculated. Then, we calculate the distance of all cluster's points to the calculated mean. Finally, the closest one will be picked as the representative of the cluster

- What distance measures you tried

  In order to try various distance measure, I added a new function to mykmedoids.m (at the end of this function), which is called d=distanceMeas(x,y,mode) In this function, x and y are any two n-dimensional points and mode is the distance measure that we want to test.

  In this function, three different measures are defined:

  1- Euclidean: Which is the 2-norm of (x-y) and actually the ordinary straight-line distance between the two point

  2- Manhattan: Also known as city block and taxicab metric, which is the sum of the absolute difference of the points, and in linear algebra term is known as 1-norm of (x-y)

  3- Chebyshev: Which is the Inf-norm of (x-y) or the maximum of absolute of (x-y)

  I have tested these three measures on various examples and it seems both Euclidean and Manhattan give the better results (We can say Euclidean is even slightly better)

- When you stopped iteration

  In my codes, the iteration is conducted in a while-loop with a TRUE flag, so we need some stopping criteria. For the stopping criteria, I did the following:

  - After each iteration, the program checks the cost function values (the sum of the distance from centroids). If the difference between two consecutive cost values, |cost(i) – cost(i-1)|, is less than a small value (a tolerance value e.g. 10^{-5}, then we can say it is converged. We can also check the relative change of the cost values |cost(i) – cost(i-1)|/|cost(i)|

- Moreover, at each step, the program checks if there is any updates in cluster assignments. If there is not any update in the class ids, it means that the solution is converged
- In the worst case, if the solution is not converged, to avoid an infinite loop, the program will stop after a maximum number of iterations (set by MAX_ITER)

In the following, I will answer questions (2) and (3) for both K-medoids and K-means together:

(2) Attach a picture of your own

In addition to the given pictures, I have tested my codes with various other pictures, including the following:

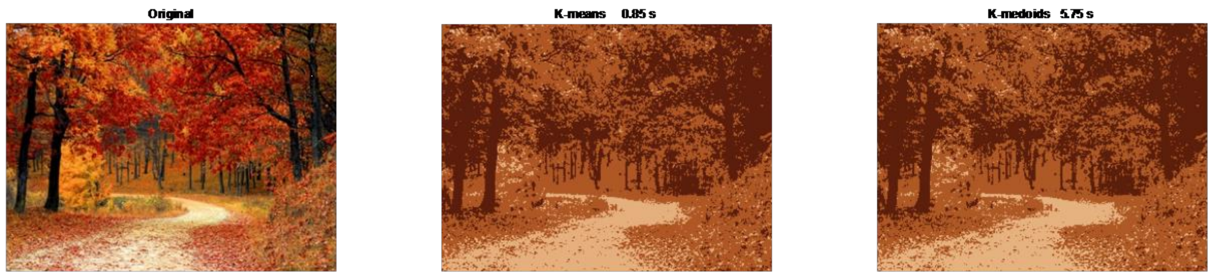
	Name: autumn.jpg Dimension: 320 x 240
	Name: nature.jpg Dimension: 288 x 175
	Name: trees.jpg Dimension: 320 x 240

In the next part, the result of running the program for the first image "autumn.jpg" has been presented.

(3) Running K-means/K-medoids on the pictures with different K:

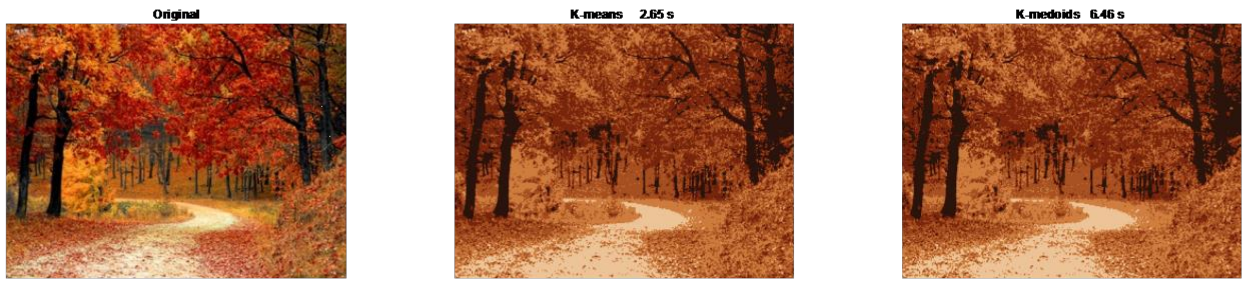
I also modified the main function to output the iterations and running times:

- With small value K = 3



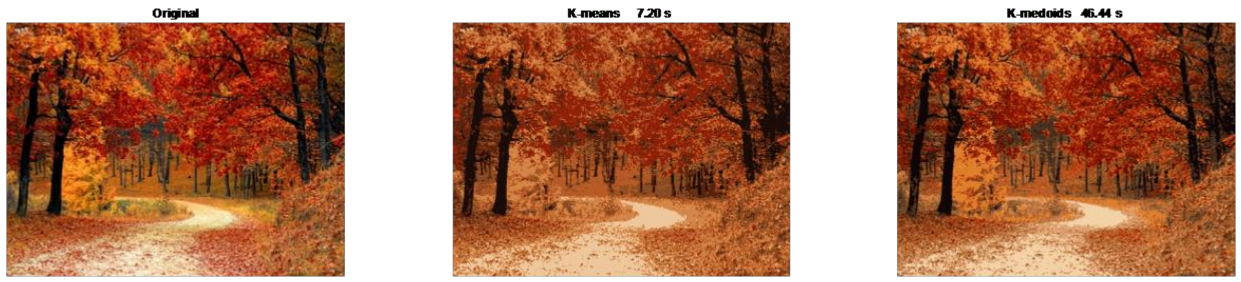
27 iterations 11 iterations

- With K = 5



57 iterations 19 iterations

- With large value K = 16



63 iterations 29 iterations

- What did you observe with different K?

  It is clear that by increasing K (which means having more colors) we get a better quality image, but it needs more effort which means more iterations. With the smaller K we actually obtained a more compressed image which needs smaller space to be saved on disk.

- How long does it take for convergence?

  In the following table the summary of runtimes and iterations for K=3 and K=16 for both methods are presented.

	K-means		**K-medoids**	
	Iterations	Running time (s)	Iterations	Running time (s)
K = 3	27	0.85	11	5.75
K = 16	63	7.20	29	46.44

(4) Running K-medoids with different initial centroids:

- Does it affect the final results? Do you see same or different result for each trial?

  I have tested both methods with different initial centroids (line #41, #42, #43 in the MATLAB code) including random numbers, all ones (the first cluster), and all K (the last cluster). As expected, the results are [almost] the same (of course, the number of iterations and running time is slightly different and will not be a constant number!)

(5) Repeat question 2 and 3 with K-means. (Already did!) Do you see significant difference between these two methods, in terms of:

- Output quality → The output images are not different (at least, for the various example images that I tested, the outputs are pretty similar)
- Robustness → K-medoids seems more robust after testing on various examples. In general, we can expect that K-medoids is comparatively more robust to noise (and also outliers) than K-means because it minimizes a sum of pairwise dissimilarities (absolute distance) between the points and the selected centroid; however, in K-means we minimize a sum of squared Euclidean distance.
- Running time → It is very interesting that the number of iterations that needed for convergence in K-medoids is much less than the number of iterations required in K-means (less than half, in all the examples that I tested); however, the running time of the K-medoids is much more than the K-means. That happens because in K-medoids, the program needs much more computations to find the closest point (center) to the current cluster mean (updating centroid section in my code).