“Pattern Recognition”

Shervin Halat

98131018

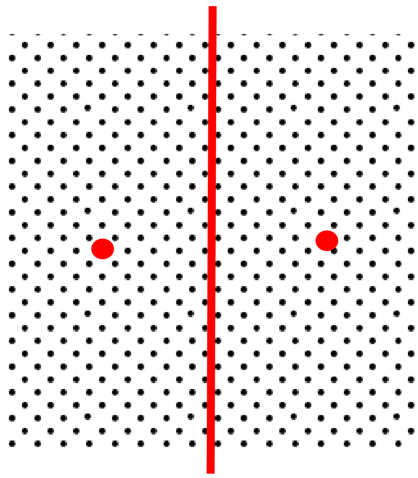
Homework 5

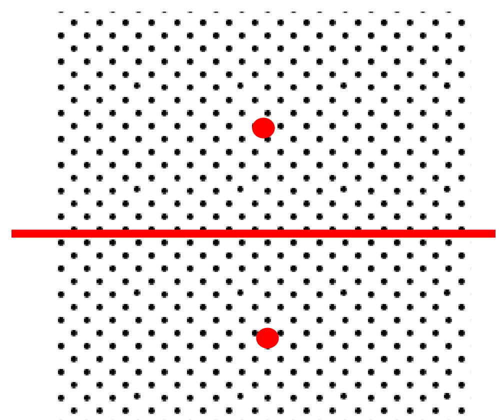
1.

Considering K-Means algorithm converges to local minimum, the final clusters depend on initial centroids. Thus, for some parts more than one probable clustering is shown (clusters' separating lines and centroids are shown by red lines and red dots respectively):

a.

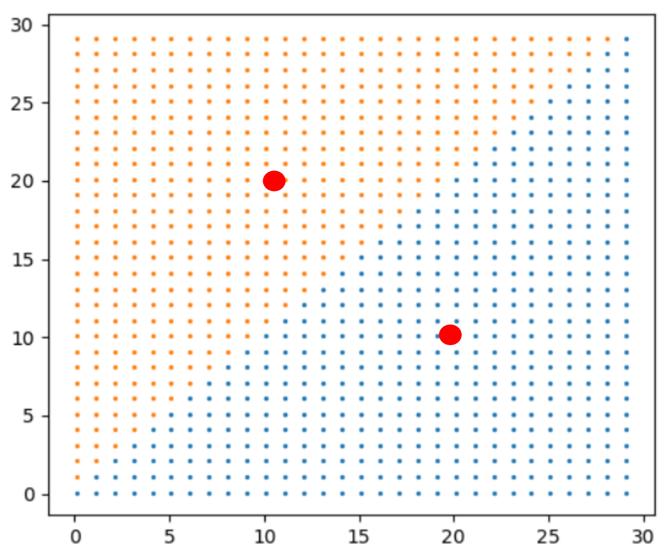
Probable answers based on initial centroids: (global minimum)





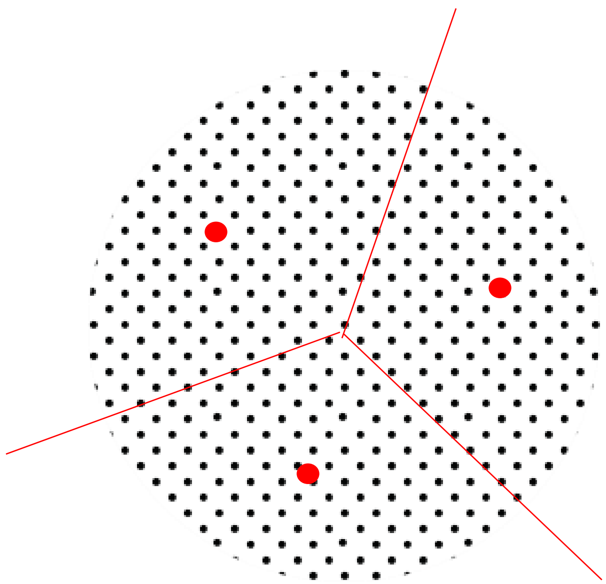
Or even diagonal: **(based on p1.a script output)**

(local minimum)

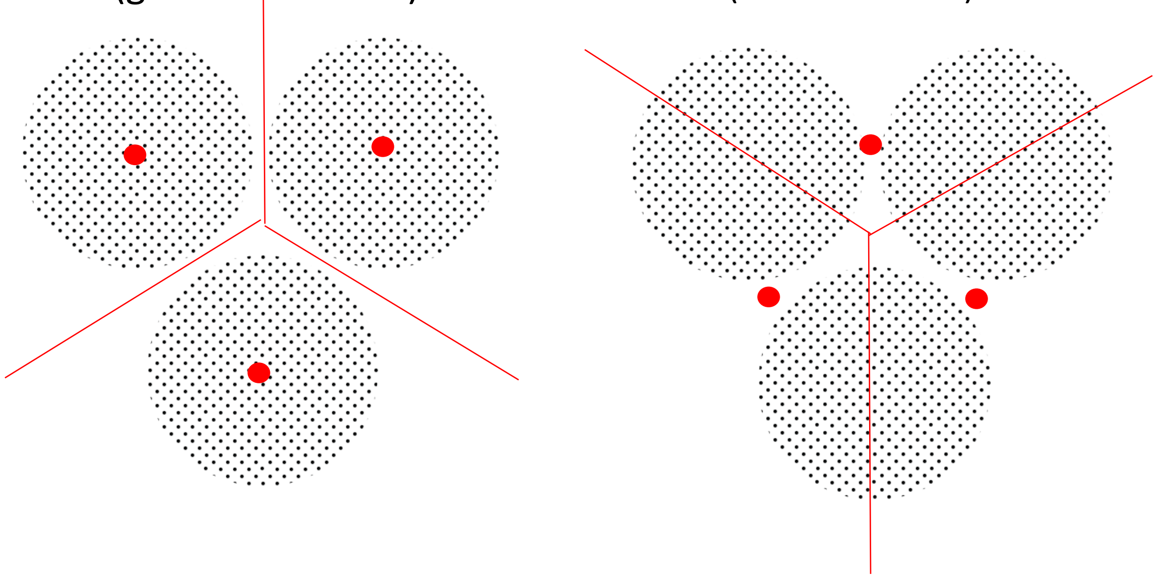


b.

For a single cyclic shaped datapoints, any three equal clusters (k=3) obtained by each triple of diameters is probable. That is, angle between diameters should be 120^o degree.

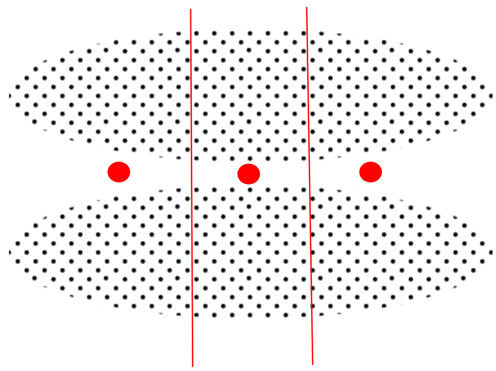


c. (global minimum) (local minimum)

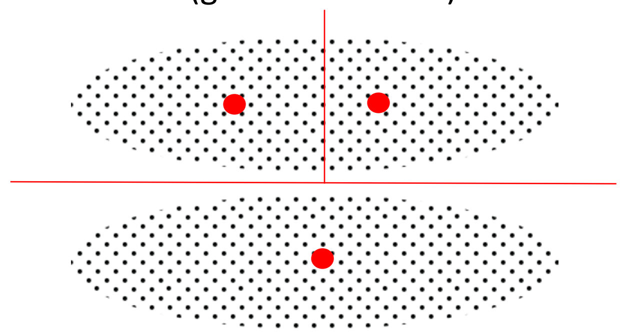


d.

(local minimum)

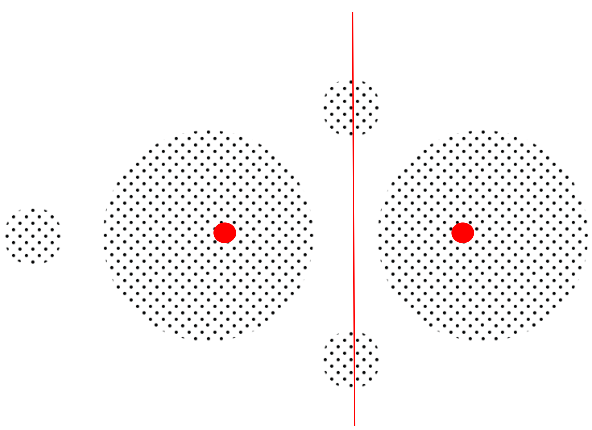


(global minimum)

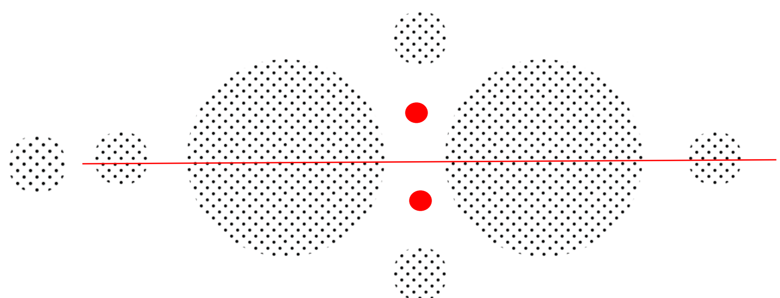


e.

(global minimum)



(local minimum)



f.

???

g.

???

h.

Result (2). Since K-means tends to cluster data into equally shaped and sized groups.

i.

Result (1)

j.

Result (2)

k.

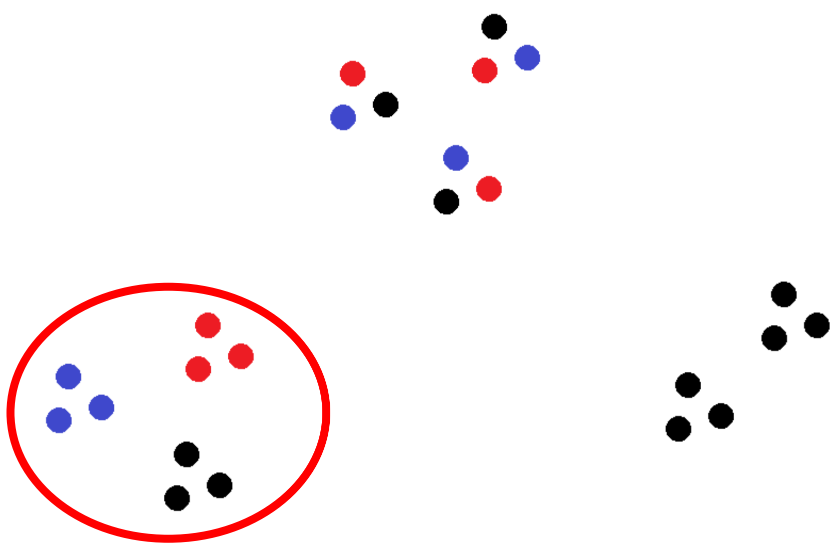
Result (2)

l.

Result (1)

m.

The only group consisting well seperated clusters is shown with red circle:



2.

Note:

“c1” denotes: center of cluster 1

“mean 1” denotes: mean of cluster 1 datapoints

a.

Step1: c1 = [1,3] , c2 = [5,3]

Considering euclidean distances:

→ Cluster1 includes: (1,2,3) → mean 1: [5/3 , 3]

Cluster2 includes: (4,5,6) → mean 2: [13/3 , 3]

Step2: c1 = [5/3 , 3] , c2 = [13/3 , 3]

Considering euclidean distances:

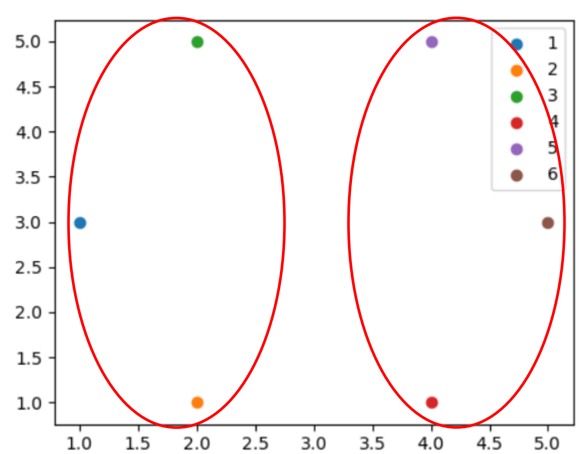
→ Cluster1 includes: (1,2,3) → mean 1: [5/3 , 3]

Cluster2 includes: (4,5,6) → mean 2: [13/3 , 3]

“Non of datapoints’ clusters changed in step 2”

With only two steps we can say kmeans converged with the centroids of [5/3 , 3] and [13/3 , 3].

Clusters:



b.

Step1: c1 = [2,5] , c2 = [4,1]

Considering euclidean distances:

→ Cluster1 includes: (1,3,5) → mean 1: [7/3 , 13/3]

Cluster2 includes: (2,4,6) → mean 2: [11/3 , 5/3]

Step2: c1 = [7/3 , 13/3] , c2 = [11/3 , 5/3]

Considering euclidean distances:

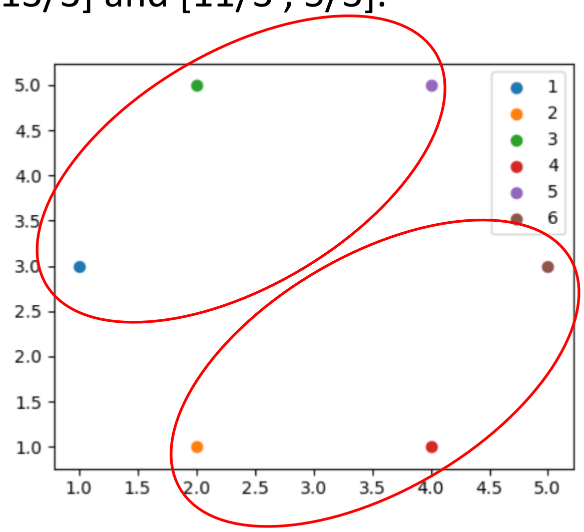
→ Cluster1 includes: (1,2,3) → mean 1: [7/3 , 13/3]

Cluster2 includes: (4,5,6) → mean 2: [11/3 , 5/3]

"Non of datapoints' clusters changed in step 2"

With only two steps we can say kmeans converged with the centroids of [7/3 , 13/3] and [11/3 , 5/3].

Clusters:



c.

Note:

“dis(#1 , C1)” denotes: euclidean distance between datapoint ‘1’ and centroid ‘C1’.

Cosidering equation below:

$$SSE = \sum_{k=1}^{K} \sum_{\forall x_i \in C_k} \|x_i - \mu_k\|^2$$

Clustering ‘a’:

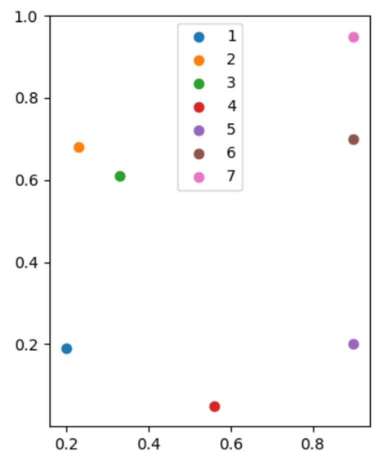
SSE(a) = dis(1 , c1) + dis(2 , c1) + dis(3 , c1) + dis(4 , c2) + dis(5 , c2) + dis(6 , c2) = **17.3 = SSE(a)**

Clustering ‘b’:

SSE(b) = dis(1 , c1) + dis(3 , c1) + dis(5 , c1) + dis(2 , c2) + dis(4 , c2) + dis(6 , c2) = **15 = SSE(b)**

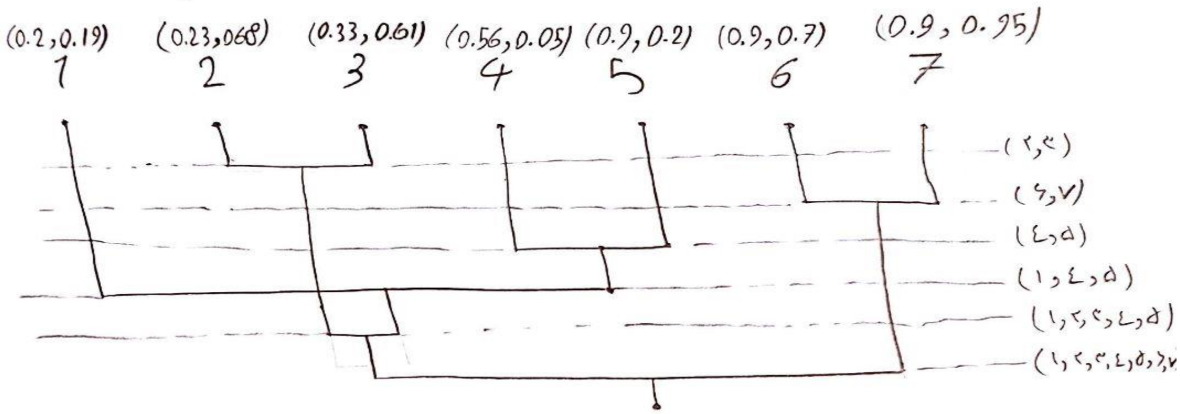
Therefore, comparing two SSE values, **‘b’ clustering is better.**

d & e.

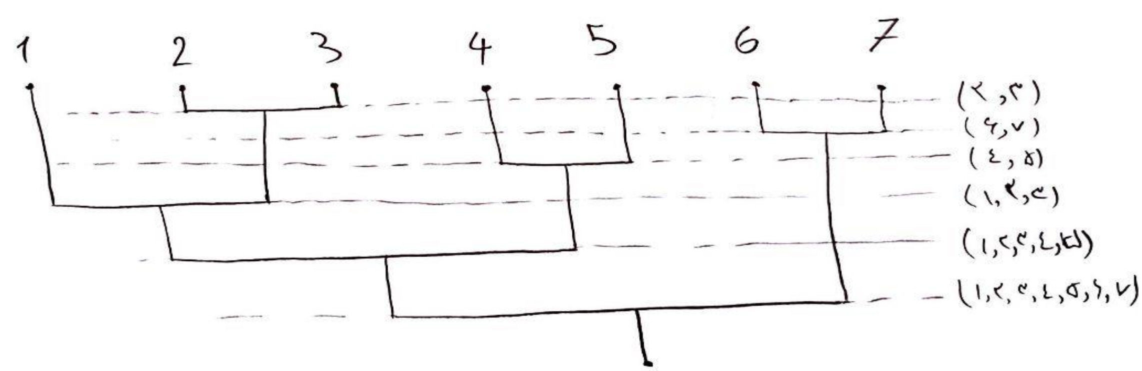


Hierarchical Clustering:

Single Linkage, Euclidean Distance



Complete Linkage, Euclidean Distance



3.

a.

Probably the idea behind this problem is to apply K-means clustering on the dataset of RGB values in photo, as if these RGBs are sample points in three dimensional space. And lasly, introduce the mean of each cluster as one the main colors of the photo! (The ‘k’ just refers to the number of clusters desired in the final output)

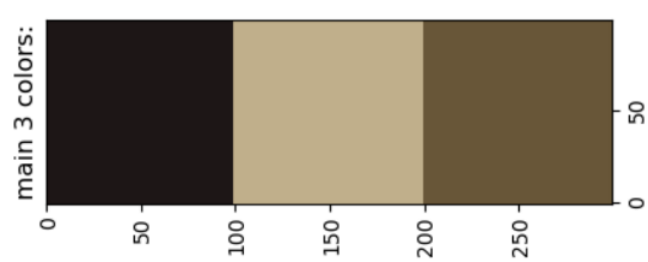
(Related script is p3.a)

Note: Due to high runtime of implemented K-means code, for the image of shape (495,600), the Scikit library was used.

K = 3:

```
main 3 colors in RGB is:
[[ 29.18064159  22.63008653  22.05907855]
 [192.36221851 175.29949255 139.13829218]
 [104.73695965  86.59594518  56.71692522]]
```

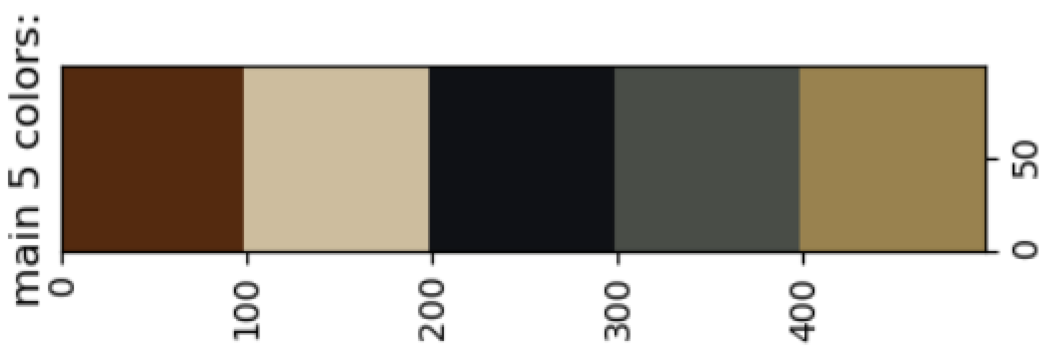
Main 3 colors:



K = 5:

```
main 5 colors in RGB is:
[[ 84.43677971  42.43709432  15.85021195]
 [205.00491742 189.939774   158.53738979]
 [ 15.40522346  17.38750748  21.83925703]
 [ 73.97372204  77.06463658  71.72827476]
 [153.57801426 130.78573909  79.72013317]]
```

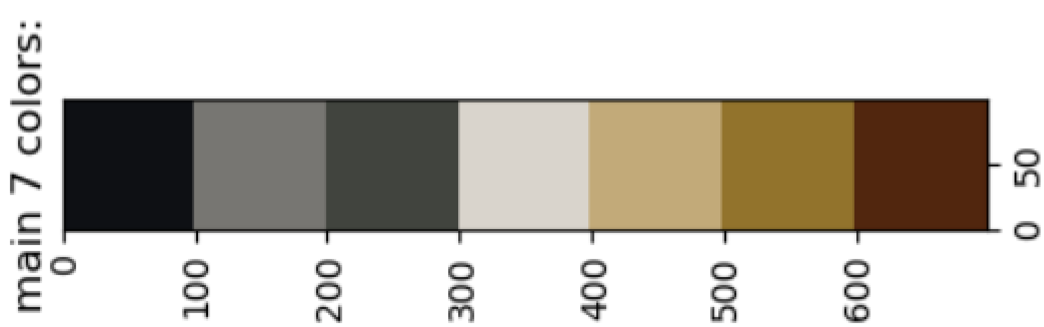
Main 5 colors:



K = 7:

```
main 7 colors in RGB is:
[[ 14.34446811  16.26746566  20.680938  ]
 [119.5195302  118.40846908 114.93519935]
 [ 65.27225348  68.78746288  62.72378065]
 [217.23042616 212.05358913 204.9287838 ]
 [194.22560119 170.68587109 121.7533473 ]
 [146.33622253 115.21973598  44.31515195]
 [ 81.92767454  38.67988089  14.71207936]]
```

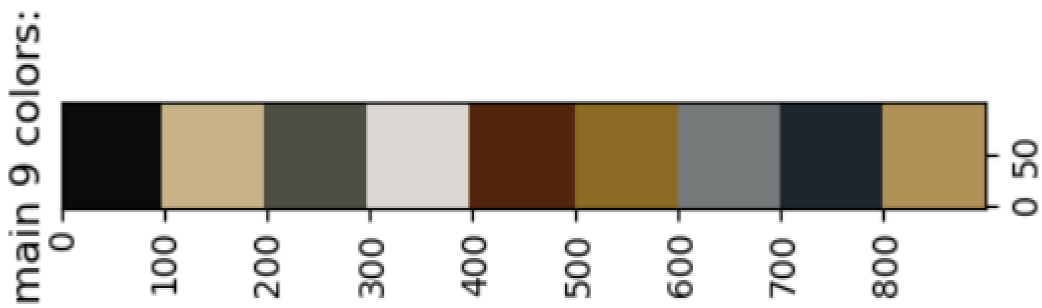
Main 7 colors:



K = 9:

```
main 9 colors in RGB is:
[[ 11.13797796  11.15741472  14.1440691 ]
 [200.30509962 179.68301956 137.40208371]
 [ 79.06746928  78.348201    66.51336162]
 [219.70875711 216.88441648 212.15462851]
 [ 81.60257336  37.95849301  14.32209571]
 [139.75627942 107.29120649  36.30434783]
 [116.66271686 121.55250028 121.54314548]
 [ 29.83386142  38.12580032  45.84052619]
 [176.70182717 146.74172648  88.19944999]]
```

Main 9 colors:

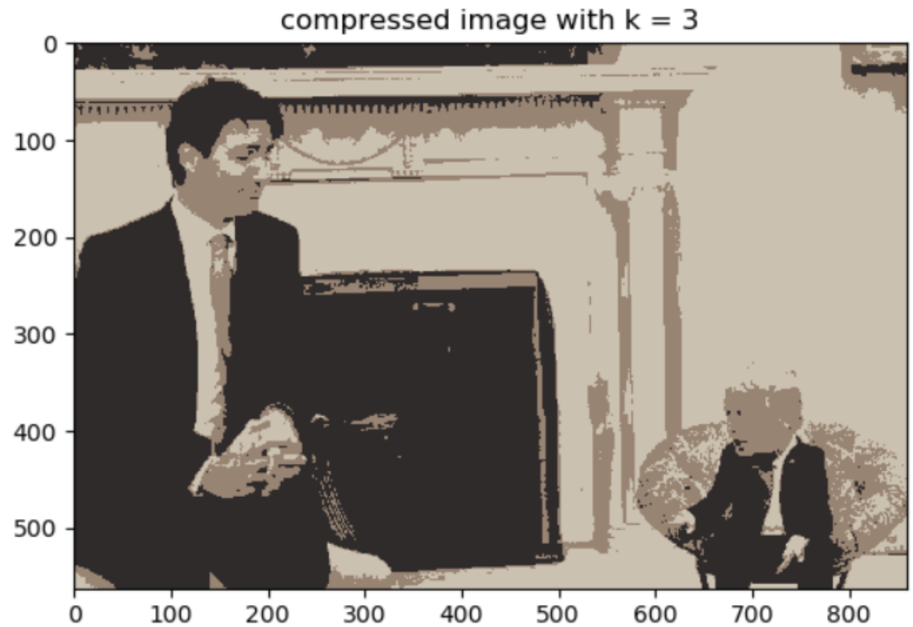


b.

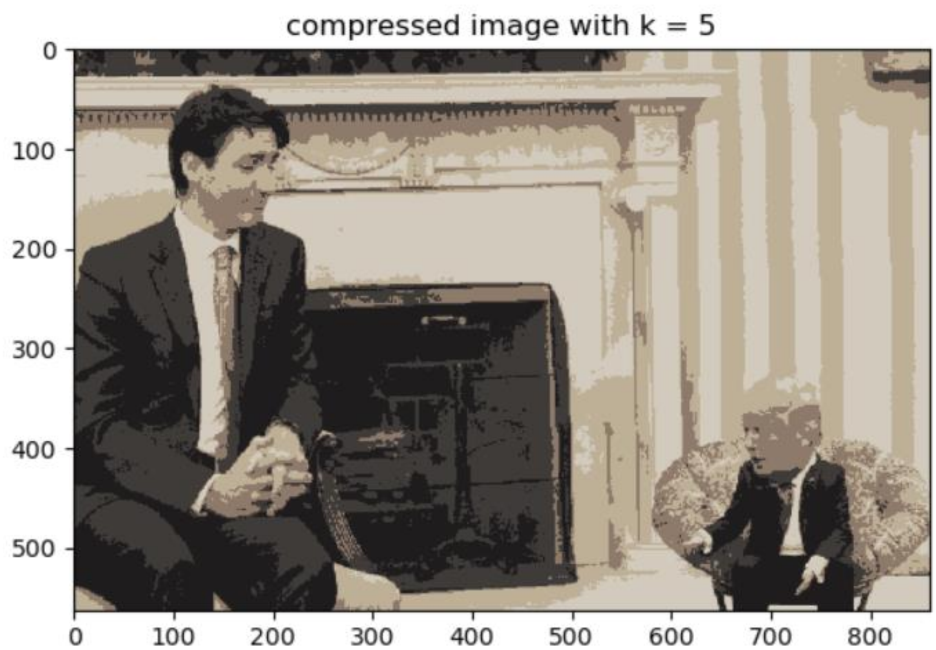
The idea behind this problem (Image Segmentation) is probably finding 'K' clusters then assign centroids' values (Centroid RGB) of each cluster to all datapoint (RGB pixel) whitin that cluster as a representation of the color of that cluster.

(Related script is p3.b)

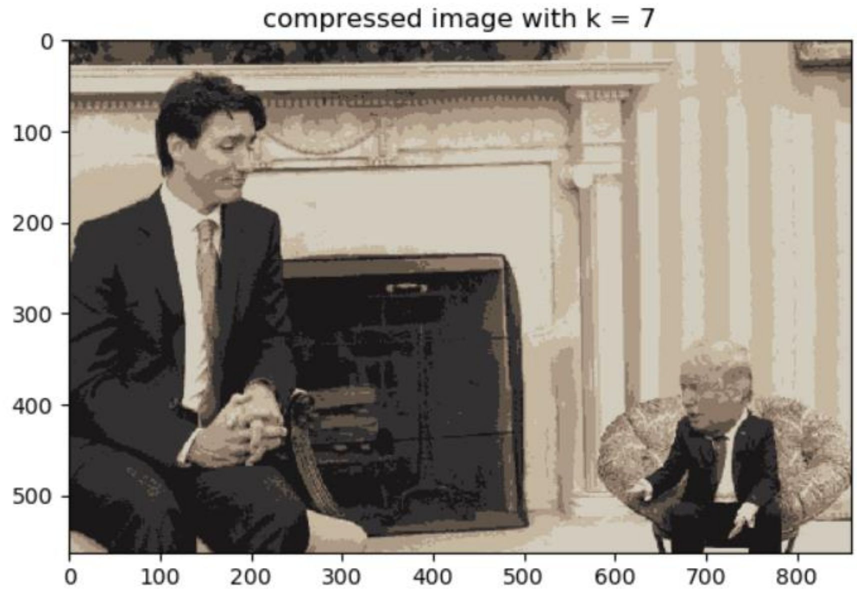
k = 3:


compressed image with k = 3
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

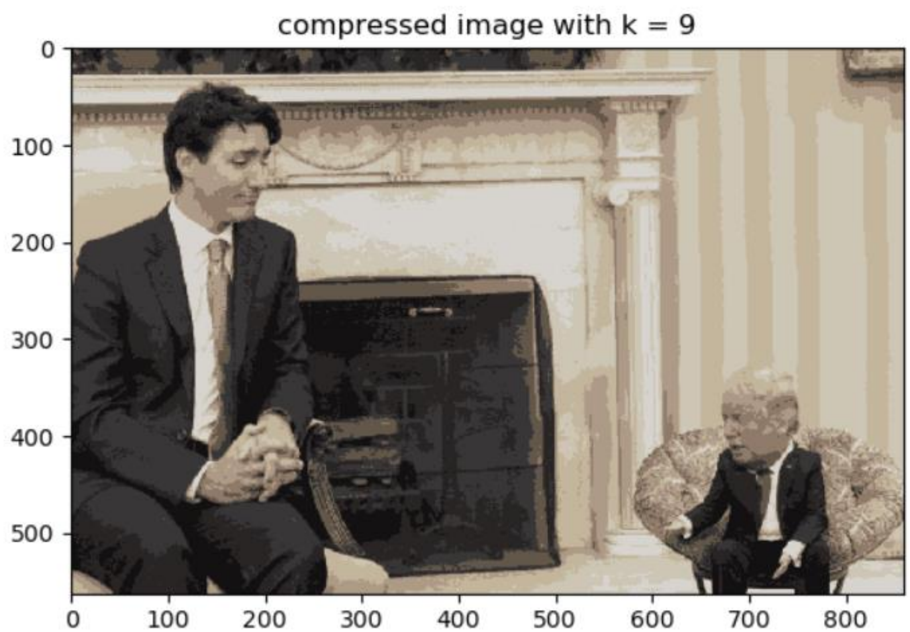
k = 5:


compressed image with k = 5
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

k = 7:


compressed image with k = 7
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

k = 9:


compressed image with k = 9
0
100
200
300
400
500
0
100
200
300
400
500
600
700
800

c.

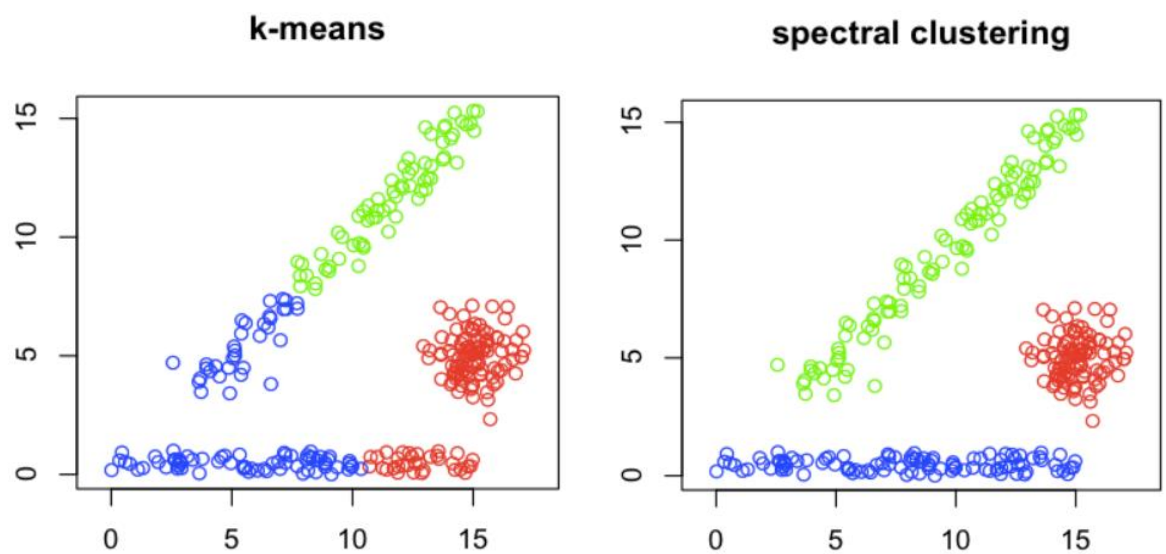
???

4.

a.

One probable solution for clustering compact and elongated clusters simultaneously may be Spectral Clustering. Spectral Clustering separate clusters based on connectedness of datapoints (as against k-means which work with distance metrics). Spectral clustering uses the spectrum (eigenvalues) of a matrix to cluster the points.

Below, an example is shown on performance of both Spectral Clustering and K-means:



b.

Unlike similar names thanks to letter ‘K’, K-means and KNN are completely different methods with no meaningful relations. Each of the mentioned learning algorithms works under separate categories. KNN is a classification (or regression) supervised learning algorithm which is applied to classify datapoints based on K nearest neighbors of that point, hence, in KNN label of each datapoint is known. Reversely, K-means becomes under unsupervised learning algorithm (clustering) which tries to find K number of clusters which do exist based on a given data consisting of datapoints with no specified labels or classes. The only similarity between the two mentioned algorithms is that their function is based on distance metrices.

c.

Solely, under one criterion, the final clustering result may vary even with the same starting points! That criterion is existing of datapoints which may be exactly in the middle of centroids. Since, in the mentioned situations the datapoint may be assigned randomly to each cluster with the same distance from centroids. Ignoring this condition, the solution with same number of clusters and starting points will always be the same.