CSE 6740 Assignment 1

Muyang Guo

2019 Sep 20th

1 Probability:

1.(a):

$$\Pr(worked\,in\,store\,C \mid woman) = \frac{\Pr(woman \cap worked\,in\,store\,c)}{\Pr(woman)}$$

Thus,

$$\Pr(worked\,in\,store\,C \mid woman) = \frac{\Pr(worked\,in\,store\,c) \times \Pr(womanC|worked\,in\,store\,c)}{\Pr(woman)}$$

Thus,

$$\Pr(worked\,in\,store\,C \mid woman) = \frac{\frac{100}{50+75+100} \times 0.7}{\frac{50 \times 0.5 + 75 \times 0.6 + 100 \times 0.7}{50+75+100}} = 0.5$$

1.(b):

$$\Pr(has\,disease \mid positive) = \frac{\Pr(has\,disease \cap positive)}{\Pr(positive)}$$

Thus,

$$\Pr(has\,disease \mid positive) = \frac{\Pr(has\,disease) \times \Pr(positive|has\,disease)}{\Pr(positive)}$$

Thus,

$$\Pr(has\,disease \mid positive) = \frac{0.005 \times 0.95}{0.995 \times 0.01 + 0.005 \times 0.95} = 0.323$$

1.(c):

Current Rank:
ATL Braves: 87-72 = 15
SF Giants:86-73 = 13
LA Dodgers:86-73 = 13
ATL Winning Scenarios:

1. If ATL Braves win the all 3 games:

$$\Pr(won\,3) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}$$

2. If ATL Braves win 2 games, SF Giants win 3 games or LA Dodgers win 3 games, ATL and either one of SF or LA has a tie with ATL and have the playoff game won:

$$\Pr(won\,2_1) = \frac{3}{8} \times 2 \times (\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2}) \times \frac{1}{2} = \frac{3}{64}$$

3. If ATL Braves win 2 games, SF Giants win 2 games or LA Dodgers win 2 games:

$$\Pr(won\,2_2) = \frac{3}{8} \times 2 \times \frac{3}{8} = \frac{9}{32}$$

4. If ATL Braves win 1 games, SF Giants win 2 games or LA Dodgers win 2 games,and either one of SF or LA has a tie with ATL to have the playoff game:

$$\Pr(won\,1) = \frac{3}{8} \times 2 \times \frac{3}{8} \times \frac{1}{2} = \frac{9}{64}$$

Thus,

$$\Pr(ATL\,win) = \Pr(won\,3) + \Pr(won\,2_1) + \Pr(won\,2_2) + \Pr(won\,1) = \frac{1}{8} + \frac{3}{64} + \frac{9}{32} + \frac{9}{64} = \frac{19}{32}$$

1.(d):
Additional Playoff Game Scenarios:
According to 1.(c). the playoff will happen in ATL winning scenario 2 and 4.
Thus:

$$\Pr(Additional\,Playoff) = \frac{3}{8} \times 2 \times \frac{1}{8} + \frac{3}{8} \times 2 \times \frac{3}{8} = \frac{3}{8}$$

2 Maximum Likelihood:

2.(a) Poisson Distribution:

Possion Distribution Equation:

$$P(x_i = k) = \frac{\lambda^k e^{-\lambda}}{k!}\,(k = 0, 1, 2...)$$

Likelihood for Poisson Distribution:

$$f(x_1, x_2, ...x_n \mid \lambda) = \frac{\lambda^{X_1} e^{-\lambda}}{x_1!} + \frac{\lambda^{X_2} e^{-\lambda}}{x_2!} + ... + \frac{\lambda^{X_n} e^{-\lambda}}{x_n!}$$

$$like(\lambda) = f(x_1, x_2, ...x_n \mid \lambda)$$

Using logarithm to the likelihood function:

$$log(\lambda) = log(f(x_1, x_2, ...x_n \mid \lambda))$$

$$log(\lambda) = \sum_{i=1}^{n}(-\lambda + x_i ln\lambda - ln(x_i!))$$

$$log(\lambda) = -n\lambda + ln\lambda \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} ln(x_i!)$$

Take derivative of λ to the log function to find the maximum:

$$\frac{dlog(\lambda)}{d\lambda} = -n + \frac{\sum_{i=1}^{n} x_i}{\lambda}$$

When:

$$\frac{dlog(\lambda)}{d\lambda} = -n + \frac{\sum_{i=1}^{n} x_i}{\lambda} = 0$$

Thus maximum likelihood estimator λ,

$$\lambda = \frac{\sum_{i=1}^{n} x_i}{n} = \overline{x}$$

2.(b) Multinomial Distribution:

The probability density function of Multinomial Distribution:

$$f(x_1, x_2, ...x_k; n, \theta_1, \theta_2, ..., \theta_k) = \frac{n!}{x_1!x_2!...x_k!}\Pi_{j=1}^{k}\theta_i^{x_j}$$

Using logarithm to the likelihood function:

$$log(f(x_1, x_2, ...x_k; n, \theta_1, \theta_2, ..., \theta_k)) = ln(n!) - \sum_{j=1}^{k} ln(x_i) + \sum_{j=1}^{k} x_j ln\theta_j$$

Since$\sum_{j=1}^{k} \theta_j = 1$, Take account of the constraints, applying Lagrange multiplier λ:

$$L(x_1, x_2, ...x_k; n, \theta_1, \theta_2, ..., \theta_k) = log(f(x_1, x_2, ...x_k; n, \theta_1, \theta_2, ..., \theta_k)) + \lambda(1 - \sum_{j=1}^{k})$$

$$L = ln(n!) - \sum_{j=1}^{k} ln(x_i) + \sum_{j=1}^{k} x_j ln\theta_j + \lambda(1 - \sum_{j=1}^{k} \theta_j)$$

Taking derivative of θ to Lagrange function,

$$\frac{\partial L}{\partial \theta} = \sum_{j=1}^{k} \frac{x_j}{\theta_j} - \sum_{j=1}^{k} \lambda$$

Let $\frac{\partial L}{\partial \theta} = 0$,

$$\theta_j = \frac{x_j}{\lambda}$$

As $\sum_{j=1}^{k} \theta_j = 1$,$\sum_{j=1}^{k} x_j = n$

$$\sum_{j=1}^{k} \frac{x_j}{\lambda} = 1$$

Thus,

$$\lambda = n$$

Thus, the maximum likelihood estimation of θ_j is:

$$\theta_j = \frac{x_j}{n}, j = 1, 2...k$$

2.(c) Gaussian Normal Distribution:
A uni-variate Gayssian Normal Distribution:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Taking logarithm:

$$L = log(\mathcal{N}(x; \mu, \sigma^2)) = -\frac{1}{2} nln(2\pi) - nln\sigma - \sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2\sigma^2}$$

Partial derivative for both parameters μ and σ, make them be zero respectively:

$$\frac{\partial L}{\partial \mu} = \frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i - \mu) = 0$$

$$\frac{\partial L}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2} \sum_{i=1}^{n} (x_i - \mu) \frac{1}{\sigma^4} = 0$$

Thus, the maximum likelihood estimator of μ and σ^2 :

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

3 Principal Component Analysis:

3.(a):

$$J = \frac{1}{N}\sum_{n=1}^{N}(< x^n - \tilde{x}^n >< x^n - \tilde{x}^n >)$$

$$J = \frac{1}{N}\sum_{n=1}^{N}(< x^n, x^n > + < \tilde{x}^n, \tilde{x}^n > -2 < x^n, \tilde{x}^n >)$$

$$J = \frac{1}{N}\sum_{n=1}^{N}((x^n)^\mathsf{T}x^n + (\tilde{x}^n)^\mathsf{T}\tilde{x}^n - 2(x^n)^\mathsf{T}\tilde{x}^n)$$

Taking partial derivative of J on z_j^n and let $\frac{\partial J}{\partial z_j^n} = 0$:

$$\frac{\partial}{\partial z_j^n}(\frac{1}{N}\sum_{n=1}^{N}((\tilde{x}^n)^\mathsf{T}\tilde{x}^n - 2(x^n)^\mathsf{T}\tilde{x}^n)) = 0$$

$$\frac{\partial}{\partial z_j^n}(\frac{1}{N}\sum_{n=1}^{N}((\sum_{i=1}^{M} z_i^n u_i + \sum_{i=M+1}^{D} b_i u_i)^\mathsf{T}(\sum_{i=1}^{M} z_i^n u_i + \sum_{i=M+1}^{D} b_i u_i) - 2(x^n)^\mathsf{T}(\sum_{i=1}^{M} z_i^n u_i + \sum_{i=M+1}^{D} b_i u_i))) = 0$$

$$\frac{\partial}{\partial z_j^n}(\frac{1}{N}\sum_{n=1}^{N}(\sum_{i}^{M} z_i^n z_i^n u_i \mathsf{T} u_i + 2\sum_{i=1}^{M}\sum_{i=M+1}^{D} z_i^n b_i u_i \mathsf{T} u_i - \sum_{i=1}^{M} 2z_i n(x^n)^\mathsf{T}u_i)) = 0$$

Here in the first term, i is equal for u_i, z_i, the second term , i is from $1, 2, ...M$ for z_i^n, u_i^T, i is from $M+1, M+2, ...D$ for b_i, u_i, and i is from $1, 2, ...M$ for the third term:
Thus,

$$\frac{\partial}{\partial z_j^n}((z_i^n)^2 - 2z_i^n(x^n)^\mathsf{T}u_i) = 0$$

is equivalent to :

$$\frac{\partial}{\partial z_j^n}((z_j^n)^2 - 2z_j^n(x^n)^\mathsf{T}u_j) = 0$$

As the i is from $1, 2, ...M$ for these terms and j is from $1, 2, ...M$. Thus,

$$2z_j^n - 2(x^n)^\mathsf{T}u_j = 0$$

$$z_j^n = (x^n)^\mathsf{T}u_j, j = 1, 2...M$$

3.(b):
As,

$$J = \frac{1}{N}\sum_{n=1}^{N}((x^n)^\mathsf{T}x^n + (\tilde{x}^n)^\mathsf{T}\tilde{x}^n - 2(x^n)^\mathsf{T}\tilde{x}^n)$$

Similarly, taking partial derivative of J on b_j and let $\frac{\partial J}{\partial b_j} = 0$:

$$\frac{\partial}{\partial b_j}(\frac{1}{N}\sum_{n=1}^{N}((\tilde{x}^n)^\mathsf{T}\tilde{x}^n - 2(x^n)^\mathsf{T}\tilde{x}^n)) = 0$$

$$\frac{\partial}{\partial b_j}(\frac{1}{N}\sum_{n=1}^{N}(\sum_{i=M+1}^{D} b_i b_i u_i \mathsf{T} u_i + 2\sum_{i=1}^{M}\sum_{i=M+1}^{D} z_i^n b_i u_i \mathsf{T} u_i - \sum_{i=M+1}^{D} 2b_i(x^n)^\mathsf{T}u_i)) = 0$$

the first term i is from $M+1, ..., D$, the second term i is from $1, 2, ...M$ for z_i^n, u_i^T, i is from $M+1, M+2, ...D$ for b_i, u_i,and i is from $M+1, M+2, ...D$ for the third term:
Thus,

$$\frac{\partial}{\partial b_j}(\frac{1}{N}\sum_{n=1}^{N}(b_i^2 - 2b_i(x^n)^\mathsf{T}u_i)) = 0$$

$$2b_i - 2\frac{1}{N}\sum_{n=1}^{N}(x^n)^\mathsf{T}u_i = 0$$

Thus,

$$b_j = \frac{1}{N}\sum_{n=1}^{N}(x^n)^\mathsf{T}u_j, j = M+1, ..., D$$

3.(c):

$$\tilde{x}^n = \sum_{i=1}^{M}((x^n)^\mathsf{T}u_i)u_i + \sum_{i=M+1}^{D}(\frac{1}{N}\sum_{n=1}^{N}(x^n)^\mathsf{T}u_i)u_i$$

$$x^n - \tilde{x}^n = \sum_{i=1}^{D}((x^n)^\mathsf{T}u_i)u_i - \sum_{i=1}^{M}((x^n)^\mathsf{T}u_i)u_i - \sum_{i=M+1}^{D}(\frac{1}{N}\sum_{n=1}^{N}(x^n)^\mathsf{T}u_i)u_i$$

$$x^n - \tilde{x}^n = \sum_{i=M+1}^{D}(((x^n)^\mathsf{T}u_i - \frac{1}{N}\sum_{n=1}^{N}(x^n)^\mathsf{T}u_i)u_i)$$

3.(d):

$$x^n - \tilde{x}^n = \sum_{i=M+1}^{D}(((x^n)^\mathsf{T}u_i - \frac{1}{N}\sum_{n=1}^{N}(x^n)^\mathsf{T}u_i)u_i)$$

$$x^n - \tilde{x}^n = \sum_{i=M+1}^{D}(((x^n)^\mathsf{T}u_i - (\bar{x}^\mathsf{T}u_i)u_i)$$

$$x^n - \tilde{x}^n = \sum_{i=M+1}^{D}(((x^n - \bar{x})^\mathsf{T}u_i)u_i)$$

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}((x^n)^\mathsf{T}u_i - (\bar{x}^\mathsf{T}u_i)u_i)^2$$

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}((x^n)^\mathsf{T}u_i - (\bar{x}^\mathsf{T}u_i))^2$$

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}((x^n)^\mathsf{T} - \bar{x}^\mathsf{T})u_i((x^n)^\mathsf{T} - \bar{x}^\mathsf{T})u_i$$

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}(u_i)^\mathsf{T}((x^n)^\mathsf{T} - \bar{x}^\mathsf{T})((x^n)^\mathsf{T} - \bar{x}^\mathsf{T})^\mathsf{T}u_i$$

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}(u_i)^\mathsf{T}(x^n) - \bar{x})((x^n)^\mathsf{T} - \bar{x}^\mathsf{T})u_i$$

Since $S = \frac{1}{N}\sum_{n=1}^{N}(x^n - \bar{x})(x^n - \bar{x})^\mathsf{T}$, plugin to J,

$$J = \sum_{i=M+1}^{D} u_i^\mathsf{T} S u_i$$

Applying Lagrange multiplier,

$$L = \sum_{i=M+1}^{D} u_i^\mathsf{T} S u_i + \sum_{i=M+1}^{D} \lambda_i(1 - u_i^\mathsf{T}u_i)$$

Taking derivative of u_i and let it be 0;

$$0 = 2Su_i - \lambda_i 2u_i$$

Thus,

$$Su_i = \lambda_i u_i$$

To Minimize J, λ_i should be as small as possible, as u_i should be set to the eigenvector corresponding to the ith largest eigenvalue.

4 Clustering:

4.(a):
Taking derivative about μ^k to J:

$$\frac{\partial J}{\partial \mu^k} = \frac{\partial}{\partial \mu^k} \sum_{n=1}^{N} \sum_{k=1}^{K} r^{nk} ||x^n - \mu^k||^2$$

$$\frac{\partial J}{\partial \mu^k} = \frac{\partial}{\partial \mu^k} \sum_{n=1}^{N} \sum_{k=1}^{K} r^{nk} ((x^n)^\intercal - 2(x^n)^\intercal \mu^k + (\mu^k)^\intercal \mu^k)$$

$$\frac{\partial J}{\partial \mu^k} = \frac{\partial}{\partial \mu^k} \sum_{n=1}^{N} \sum_{k=1}^{K} r^{nk} (-2(x^n)^\intercal \mu^k + (\mu^k)^\intercal \mu^k)$$

$$\frac{\partial J}{\partial \mu^k} = \sum_{n=1}^{N} r^{nk} (-2(x^n) + 2\mu^k)$$

Thus,

$$\mu^k = \frac{\sum_{n=1}^{N} r^{nk} x^n}{\sum_{n=1}^{N} r^{nk}}$$

4.(b):
In the first step assigning data points to clusters, each data from $x^1, x^2, ..., x^n$ and each cluster center k clusters centers $u^1, u^2, ..., u^k$ are iterated , to assign the data to k clusters, $\pi(n) \in 1, 2, 3...k$. The averaged square distances from each data point to its respective cluster center is small following:

$$\pi(n) = argmin_{j=1,2,...,k} ||x^n - u^j||^2$$

During this process, the loss function J is decreased.
In the step of adjusting the cluster centers, following:

$$u^j = \frac{1}{|n:\pi(n)=j|} \sum_{n:\pi(n)=j} x^n = argmin_u \sum_{n:\pi(n)=j} ||x^n - u||^2$$

This step also decreases the J,
Since all steps are decreasing the J,and the domain is limited, as there are most k^n ways to partition the data to k clusters , As iterations constantly reduce J,and the new clusters are based from old clusters but a lower J ,the K-means will converge to the local optimum in finite steps.
4.(c):
The average linkage is the most similar to K-means, it looks at the distance between all pairs and minimizes the average of those distances at each step. Its objective is similar to k-means.
4.(d):
The Single Linkage can separate the two moons.as can be seen in this data, there is a relatively large gap between each moon, while each moon contains dense data points close to each other. Intuitively to separate this kind of data, method like linking data points very close to each other is an approach. With the single linkage method, it merges the data pair with the minimum distance into a cluster in each step, at the end, it should be expected that two moons are successfully separated into two clusters, each cluster belong to each moon.
The Complete linkage will fail. Because the observable minimum gap between the two moon is obviously smaller than the largest distance between points pair from two moons.Thus, it most likely will end up grouping partial moon B to partial moon A, and partial moon A to partial moon B.
The Average linkage will fail as well. As it tries to minimize the average of distance between all data points, for this kind of non-convex shape, it will most likely ends up grouping partial moon B with partial moon A as well.

5 Programming:Image Compression

5.(1):

Similarly to K-means, I implemented the K-medoids following the below steps:

step 1. randomly select K points as cluster centers from the non-repeating pixels points.
step 2. assign the points to those K cluster centers.
step 3. For each point, calculate the distance between the point and the cluster center.
(Here I used the Manhattan Distance, I also tried with the Euclidean Distance, the results are similar but only small variations, however, Manhattan running time is faster then Euclidean Distance for lots of trials. So I decided to go with the Manhattan Distance.)
step 4. Calculate the center of points in each cluster, find out the point nearest to the mean of the cluster and use this point as representative of the cluster, and update this point as a center for next iteration.
step 5. repeat step 2 - 4, until the loop breaks with ending condition.

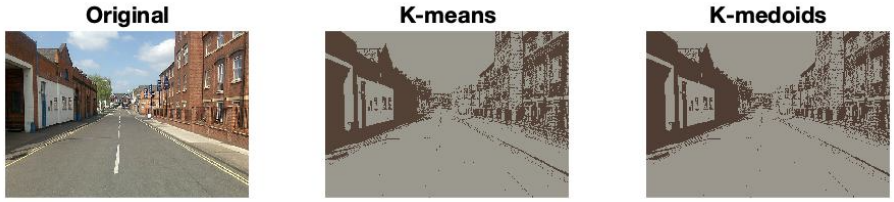
Ending Condition:
I set the ending condition is when the iterated class vector is equal to the previous class vector, meaning there are no points moving from cluster to cluster. And I add another stop condition for iteration numbers, if the iteration is exceeding over 100, the loop will also break. Either of this condition is satisfied the loop will break.

If too large K:
If the K is too large, I raised a warning indicating the K is too large, more than the non-repeating pixcels of the an image, reducing it to 32, I chose 32 as a preset maximum for K.

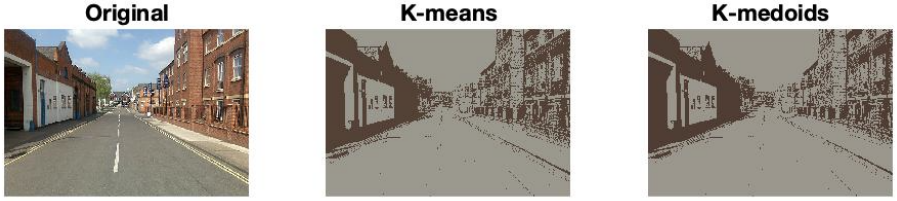
5.(2):

I chose this street picture from a wiki-page, the reason that I selected this picure is that I would like to see how the k-means/k-medoids clustering method worked for street scene segmentation based on objects color RGB.

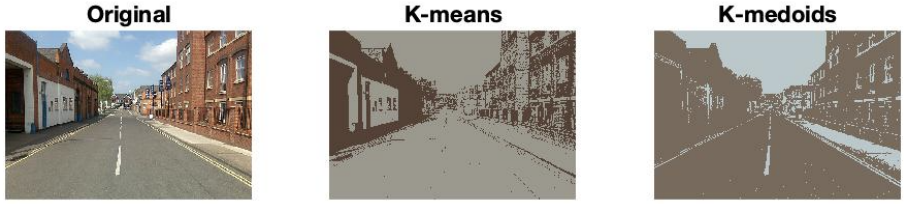
5.(3-4-5):



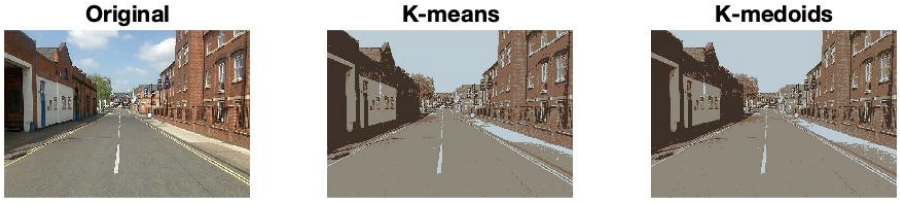
K=2 , kmeansTime = 5.2574s kmedoidsTime =2.8022s



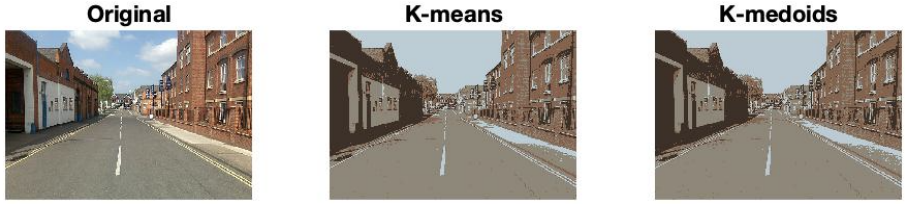
K=2 , kmeansTime = 5.6936s kmedoidsTime =2.4149s



K=2 , kmeansTime = 7.7760s kmedoidsTime =3.7337s



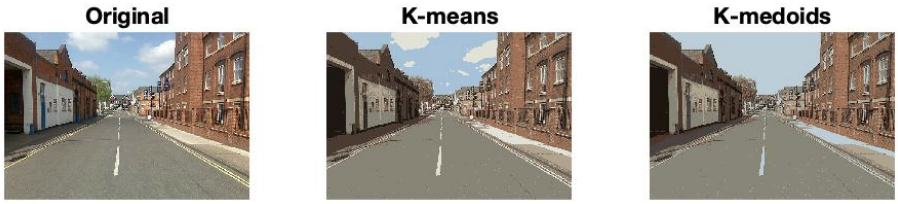
K=4 , kmeansTime = 7.5356s kmedoidsTime = 4.5753s



K=4 , kmeansTime = 15.8525s kmedoidsTime = 4.2472s

K=4 , kmeansTime = 8.3253s kmedoidsTime = 2.6811s

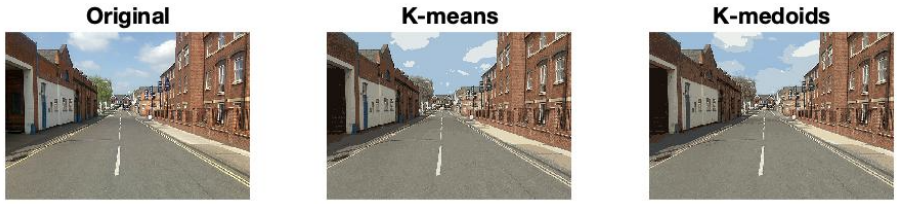
K=8 , kmeansTime =32.8409s kmedoidsTime = 11.4515s



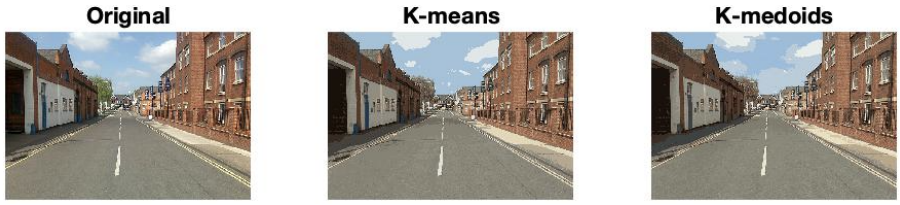
K=8 , kmeansTime =23.7053s kmedoidsTime = 12.2687s

K=8 , kmeansTime =16.7442s kmedoidsTime =8.8237s

K=16 , kmeansTime =66.0008s kmedoidsTime =18.1560s



K=16 , kmeansTime =65.6927s kmedoidsTime =16.0898



K=16 , kmeansTime =66.5397s kmedoidsTime =10.9130

K=32 , kmeansTime =100.8825s kmedoidsTime =52.7574s

K=32 , kmeansTime =101.0312s kmedoidsTime =30.0935s

K=32 , kmeansTime =102.0017s kmedoidsTime =25.1299s

5.(3):
I ran the k-medoids with K = 2,4,8,16,32, for each K I have tested with three trials. As K increases, the accuracy of the clusters based on color is increasing. The details are also becoming more and more clear. For example, clouds/skies, road edges are clustered clearly. As K increases, variations between each trial caused by the initial centers are getting smaller. When K is large, the same K trials have very similar results. The running time to converge is also increasing significantly. However, for same K trials will have a variation on the running time, due to the random initial selection is different, causing local minimum converge time different. By observation with this specific picture, when K = 2, average running time is around 2 seconds, K =4, average running time is around 4 seonds, K= 8, average running time is around 11 seconds, K=16,average running time is around 17 seconds, K=32, average running time is around 30 seconds, drifting over the average is increasing.

5.(4):
Yes. With different initial centroids/representative, it affect the results. I have observed different result for

each selection of K by running it three times. However, when K incresed, the influence on the clustered results is reduced for this specific picture, but the running time variation seems going up.

5.(5):

For this picture, when K is small, I observed that k-means have stable results with a better consistency in terms of quality, but when K is large, both methods are having similar quality results.

About the robustness, both methods work well for this particular picture,because I forced each iteration will have K clusters and reduce the K to 32 when K is too large. And to ensure the precision I set the max iteration relative large to 100.

In term of the running time, k-medoids usually takes much less time than K-means, when K is larger. This might due to my stopping condition, that I set to break the loop either reaching the maximum iterations or there are no points moving from cluster to cluster. AKA class is same. So for k-means, sometime the moving distance is very small, but still it will have to update the centroids and perform another iteration.