HW1

Sara Awid

August 2020

1 Clustering

1. Prove (using mathematical arguments) that using the squared Euclidean distance as the dissimilarity function and minimizing the distortion function, we will have:

$$\mu^j = \frac{\Sigma_i r^{ij} x^i}{\Sigma r^{ij}}$$

That is, μ_j is the center of jth cluster.
Hint: consider taken derivative of J with respect to μ_j

if we take derivative of j with respect to μ_j we get

$$\frac{\partial J}{\partial \mu} = \frac{\Sigma_i \Sigma_j r^{ij} \|x^i - \mu^j\|^2}{\partial \mu} = \Sigma_i 2(r^{ij} x^i - r^{ij} u^j) = 0$$

Therefore solving for μ^j gives us

$$\Sigma_i r^{ij} x^i = \Sigma_i r^{ij} u^j$$

which is equal to :

$$\mu^j = \frac{\Sigma_i r^{ij} x^i}{\Sigma r^{ij}}$$

2. Now suppose we replace the similarity function (the squared l2 distance) by another distance measure, the quadratic distance (also known as the Mahalanobis distance) -$d(x, y) = (x - y)^T \Sigma (x - y)$. how (prove) that the centroid in this case will be the same

$$\mu^j = \frac{\Sigma_i r^{ij} x^i}{\Sigma r^{ij}}$$

In order to show that similarity function using mahalanobis distance gives us the same centriod expression above, we can use the same approach by taking the partial derivative of d(x,y) respect to μ_j :

$$\frac{\partial d(x, u)}{\partial \mu^j} = \frac{\partial \Sigma_i \Sigma_j r^{ij} (x - u)^T \Sigma (x - u)}{\partial \mu^j}$$

$$= \Sigma_i r^{ij}(2\Sigma(x^i - u^i) = 0$$

Solving for μ^j we get

$$\Sigma_i r^{ij} x^i = \Sigma_i r^{ij} u^j$$

which leads to

$$\mu^j = \frac{\Sigma_i r^{ij} x^i}{\Sigma r^{ij}}$$

ii) We would define the assignment function based on the mahalanobis distance:

$$r^{ij} = argmin_{j=1,2..k}(x^j - u^j)^T \Sigma (x^j - u^j)$$

3. Prove (using mathematical arguments) that K-means algorithm converges to a local optimum in finite steps.

Given the objective function

$$min \frac{1}{m} \Sigma_i \Sigma_j r^{ij} \|x^i - \mu^j\|^2$$

We know that the minimum value of this function itself is finite, which is zero since that would be the smallest distance. We also know that each iteration of kmeans decreases the objective function by the following:

$$\pi(i) = argmin_{j=1..k} \|x^i - \mu^j]^2$$

And the center adjustment step decreases the objective s.t:

$$\mu^j = argmin_u \Sigma_{i:\pi(i)} \|x^i - u\|^2$$

Therefore, the objective function is monotonically decreasing given the steps above. Given that we have a finite data points, and finite m^k cluster assignment combination, then the algorithm will converge in finite steps.

4. (10 points) Calculate k-means by hands. Given 5 data points configuration in Figure 1. Assume k = 2 and use Manhattan distance (a.k.a. the l1 distance: given two 2-dimensional points (x1, y1) and (x2, y2), their distance is (x1 x2) + (y1 y2)). Assuming the initialization of centroid as shown, after one iteration of k-means algorithm, answer the following questions.

(a) Show the cluster assignment;
The cluster assignment is based on calculating the min distance from each point to A and B:
1. Cluster label of point 1 =min(distance to A, distance to B)= (1, 8) = 1 therefore $\pi(1) = A$
2. distance label point 3= min(1,8)= 1 therefore $\pi(3) = A$
doing this for all points we end up with A=[1,3] ,B=[2,4,5]

(b) Show the location of the new center;
centers will be based on min distance from the center of A to the points: min

$|2 - A_1| + |3 - A_1| + |2 - A_2| + |1 - A_2|$ plotting above for A_1, A_2, we see that we have a min at $2 <= A_1 <= 3, 1 <= y <= 2$ hence the cluster center for A remains the same.
Similarly for B, plotting the distance $| - 1 - B_1| + | - 1 - B_1| + | - 2 - B - 1| + |1 - B_2| + | - 2 - B_2| + | - B_2|$ separately for B_2, B_1 will results into global minima at (-1,-1)

(c) Will it terminate in one step? Yes, since the absolute value function has only global minima then we can only have one solution, thus it will terminate at one step

2 Image Compression Using Clustering

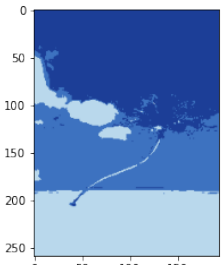
1. (10 points) Within the k-medoids framework, you have several choices for detailed implementation. Explain how you designed and implemented details of your K-medoids algorithm, including (but not limited to) how you chose representatives of each cluster, what distance measures you tried and chose one, or when you stopped iteration.
the K-medoids algorithm I implemented as the following:
Initialize the cluster center μ_j , j = 1, ..., k. • Iterate until convergence:
– Update the cluster assignments for every data point x^i : $r^{ij} = 1$ if $j = argmin_j D(x^i, \mu^j)$, and $r^{ij} = 0$ otherwise.
– Update the center for each cluster j: Given current center for cluster j, swap a random sample of points from the cluster and set it as the new center, calculate the l2 norm within cluster j. I limited the sample to be 0.2% of the sample size to increase runtime
- If the new center l_2 norm is less than l_2 norm of the old center then update the center with the new point
- I have decided to use the stopping criteria defined as within cluster variation as WCSS which equivalent to J in Question 1:

$$J = WCSS(r^{ij}) = \Sigma_i \Sigma_j \|x^i - u^j\|^2$$

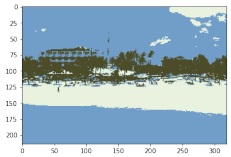
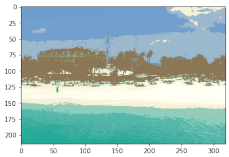
If WCCS(n-1)/WCCS(n-2) < 0.7 then stop, where n is the number of iterations and the WCSS variation has improved by 30%. Having said that, due to slow algorithm time run, I had to restrict K-medoids to end after m steps in order to retrieve results in a reasonable time.

2. Attach a picture of your own. We recommend size of 320 240 or smaller. Run your k-medoids implementation with the picture you chose, as well as two pictures provided (beach.bmp and football.bmp), with several different K. (e.g, small values like 2 or 3, large values like 16 or 32) What did you observe with different K? How long does it take to converge for each K? Please write in your report.

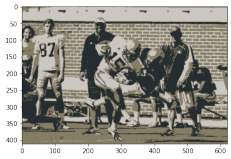
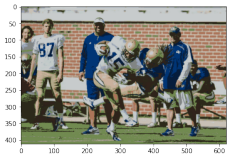
Below is an image I have selected to work with as a reference below is the unprocessed image

 Original  Kmedoid =3

Let's take a look at beach and see how it looks with different K:

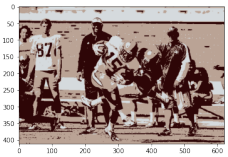
 Kmedoid =3  Kmedoid =15

Let's take a look at football and see how it looks with different K:

 Kmedoid =3  Kmedoid =15

As we can see from above, higher the K, the better the clustering results turn out. The longer the K however, the longer the runtime. For example, for K=15, the runtime for 10 iterations lasted for 3 mins while the runtime for 10 iterations for K=3 took only 40 seconds.

10 points) Run your k-medoids implementation with different initial centroids/representatives. Does it affect final result?

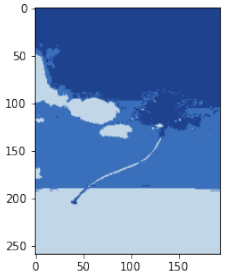
 Poor Starting Centers Kmedoid =3

Do you see same or different result for each trial with different initial assignments?
Yes it appears that if I select the first 3 points in the image matrix and set them as the centers, I get a much less clear image separation with K=3 compared to the same image above corresponding K=3 with random assignment. The difference in quality was very clear in football picture where we could see the impact of poor assignment.

(4) Repeat question 2 and 3 with k-means. Do you see significant difference

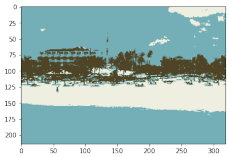
between K-medoids and k-means, in terms of output quality, robustness, or running time? Please write in your report.

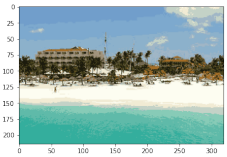

Original

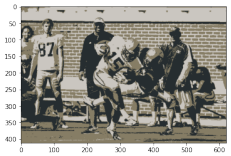


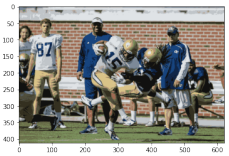
Kmean =3

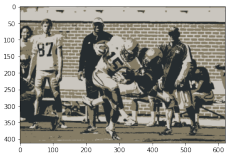
Let's take a look at beach and see how it looks with different K:


Kmean =3


Kmean =15


Kmean =3


Kmean =15


Poor Starting Centers Kmean =3

In terms of Quality and robustness, it seems that Kmeans produced better results, which is the opposite of what I expected. The reason behind this may be due to the fact that I did NOT sample data to calculate kmeans but I did so in Kmedoids where I sample 0.2% of each cluster to measure if the similarity improved or not. So really we are looking at apples vs oranges.

In terms of runtime, the difference was very clear, kmeans ran much faster than Kmedoids. For K=3 Kmedoids run for 1 min vs Kmeans ran for 16 seconds. And in the case of K=15, Kmediods ran for 3 mins while Kmeans ran only for 1min for 10 iterations only.