# 2IMW30 - FOUNDATION OF DATA MINING
## Assignment 1b Report

Duy Pham (0980384, d.pham.duy@student.tue.nl)
Mazen Aly(0978251, m.aly@student.tue.nl)

(Both students contributed equally in this assignment)

EXERCISE 1

In this assignment, we implement the k-means algorithm for clustering the input data points. The whole source code of the project can be found in the Github repository `https://github.com/MazenAly/datamining`.

Clustering is considered as an unsupervised machine learning technique. That means there are no labels in the data that we need to predict, but it helps us get some insights about the input data.

We developed four methods for initializing the first $k$ cluster centroids (where $k$ is an input). The first method is just choosing the first $k$ data points to be the centroids, the second one is randomized selections of the initial centroids, the third one is k-means++, and the fourth method is using Gonzales algorithm to pick the initial centroids.

For each method, we run k-means for different values, $k \in 3,4,5$, and in randomized methods, we experiment that 5 times. We visualize the results to get more understanding and learning experience as shown below. In figure 12, we shows the k-means clustering result on the dataset C2.txt with various initialization strategies and different values of $k$.

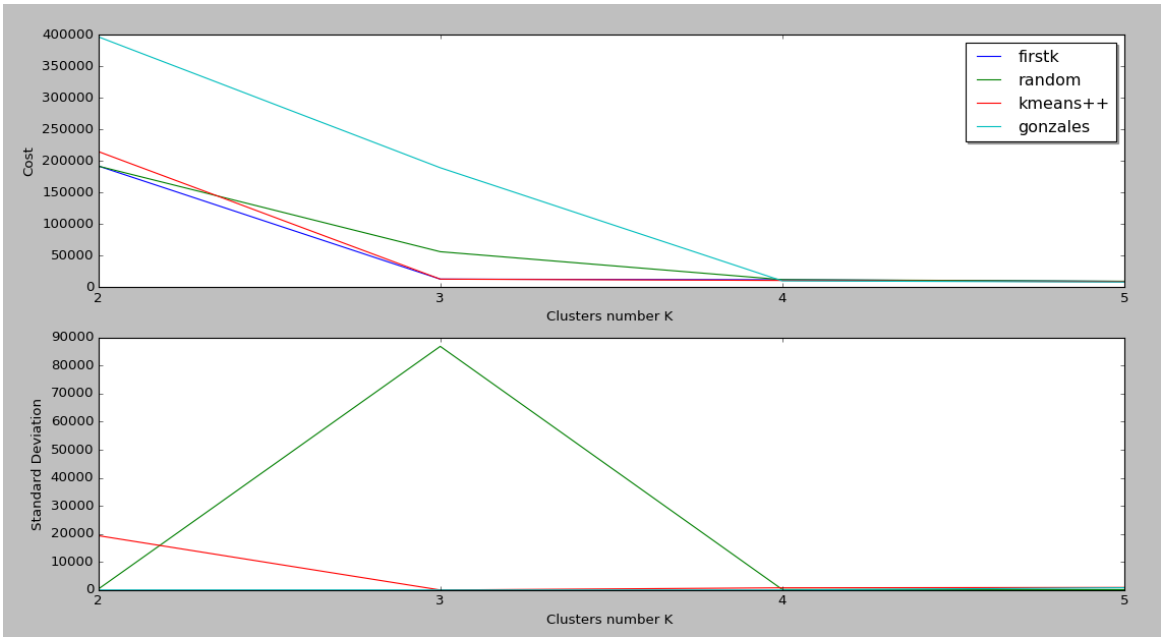



Figure 1: The average costs(top) and average Standard deviations(bottom) of several runs of k-means for different clusters numbers and using different methods of initialization.

The top digram of figure 1 shows the average cost of 5 runs of each method of initialization for k-means for different number of clusters, we can see that Gonzales method has the worst average cost across all the methods for 2 and 3 clusters. Starting of 4, all the methods have similar costs on average. The reason for this is the distribution of the data points in the file C2, where there is an outlier point that is very far from all the other points. This point is thus always chosen by Gonzales' algorithm as a center of one of the clusters.

The bottom figure shows standard deviation of the costs across all the 5 runs. We can see that the random initialization can have higher standard deviation which means that one or more of the runs can be highly deviated from average of the runs.

Figure 2 shows the convergence rate of k-means algorithm using the first 5 points as the initial centroids. We can see that of course it will be the same for all the runs, because there are no randomizations.

Doing the same for the random method, we can see in figures 6, 7 and 8 that for 3,4 and 5 clusters we can see different rates of convergence in each of the 5 runs for each value of $k$, which shows that k-means algorithm is dependent on the initial centroids.

Figure 3 and 4 shows the convergence rate of k-means algorithm of 3 clusters using Gonzales's algorithm centers as the initial centroids, we can see it converged in just one step.

But for 5 clusters, Gonzales' algorithm convergence rates are diverse between 1 and 15 steps as in figure 5.

For k-means++, we ran the algorithm 5 times for each different clusters number to visualize the convergence rate as in figures 9 , 10 and 11. We can see that the rates are also diverse across different runs but on average k-means++ converges faster due to the probabilistic strategy of choosing the first initial centroids.

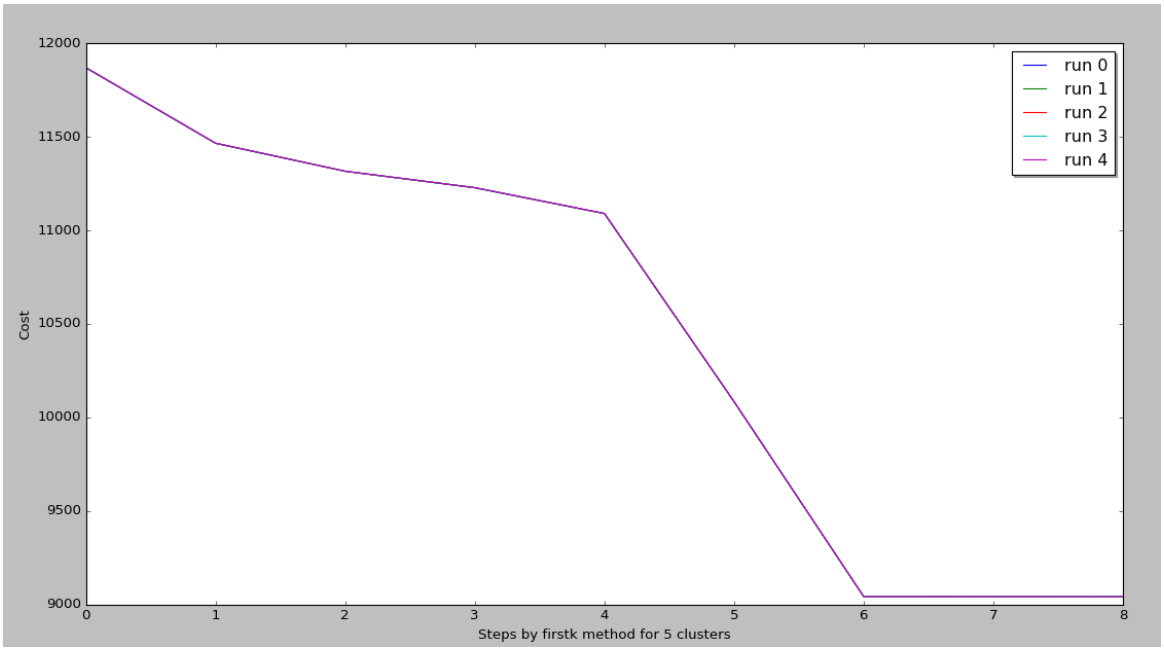



Figure 2: Convergent rate of k-means using the first 5 points as initial centroids.

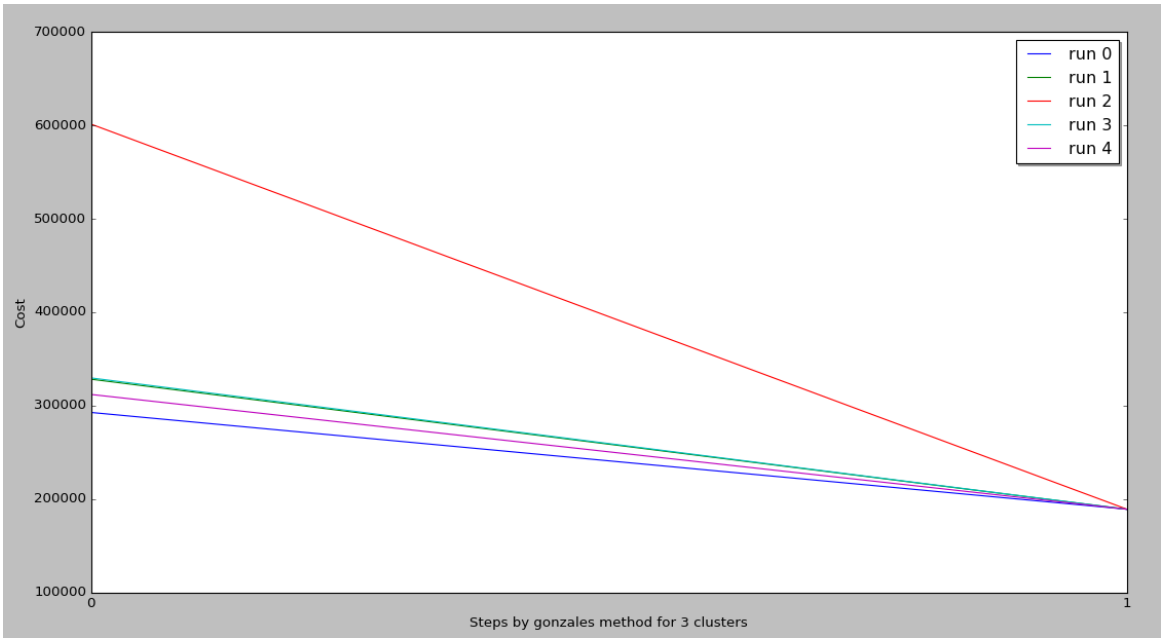



Figure 3: Convergence rate of k-means using initialization of Gonzales' algorithm and 3 clusters for 5 different runs

Based on our experiments, k-means++ is the most robust implementation as it allows some weighted randomizations. The naive method of choosing the first $k$ centers depends on how good the data is originally sorted, which does not usually happen. The uniform randomization tries to overcome that weakness, but there are also (many) other bad configurations that can be selected randomly as well. For the Gonzales algorithm, we can take a look at figure 12j. The noise is in its own cluster, and 2 meaningful clusters are grouped into only one. This example shows that using Gonzales algorithm for initializing k-means is not a good idea. This algorithm is very sensitive to noise as it always chooses the farthest point to be the next center.

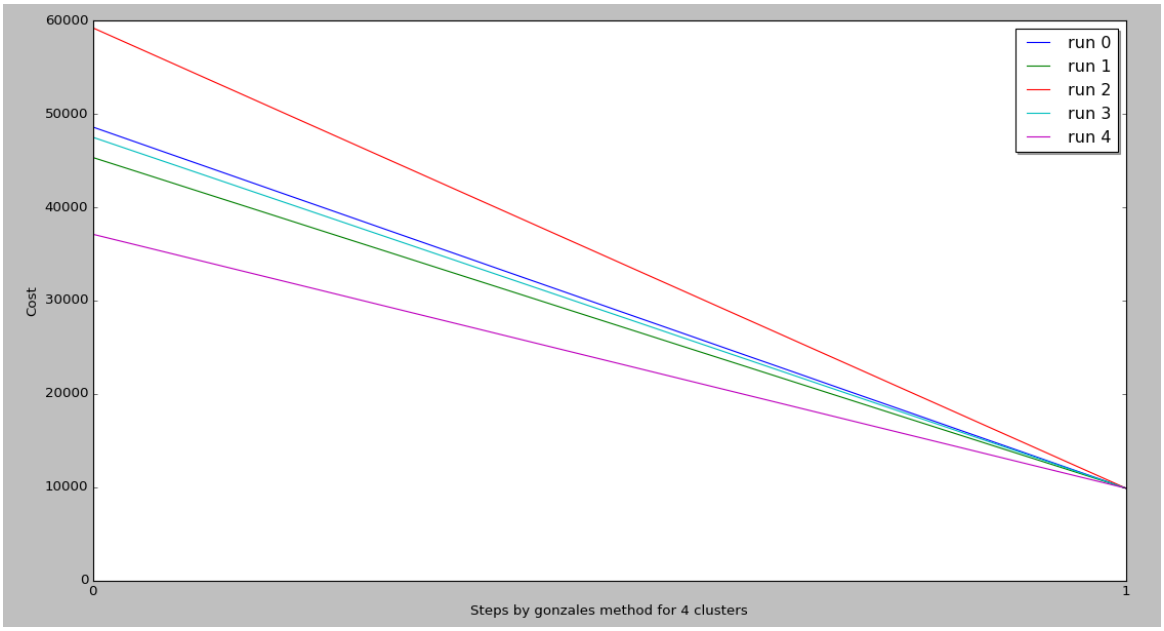



Figure 4: Convergence rate of k-means using initialization of Gonzales' algorithm and 4 clusters for 5 different runs

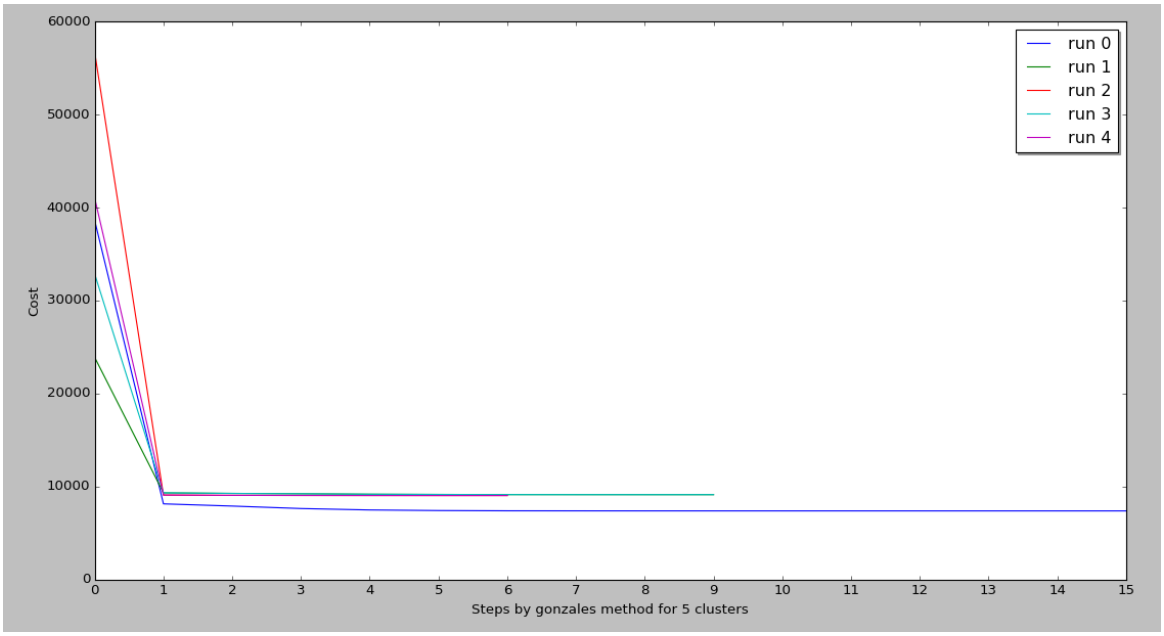



Figure 5: Convergence rate of k-means using initialization of Gonzales' algorithm and 5 clusters for 5 different runs

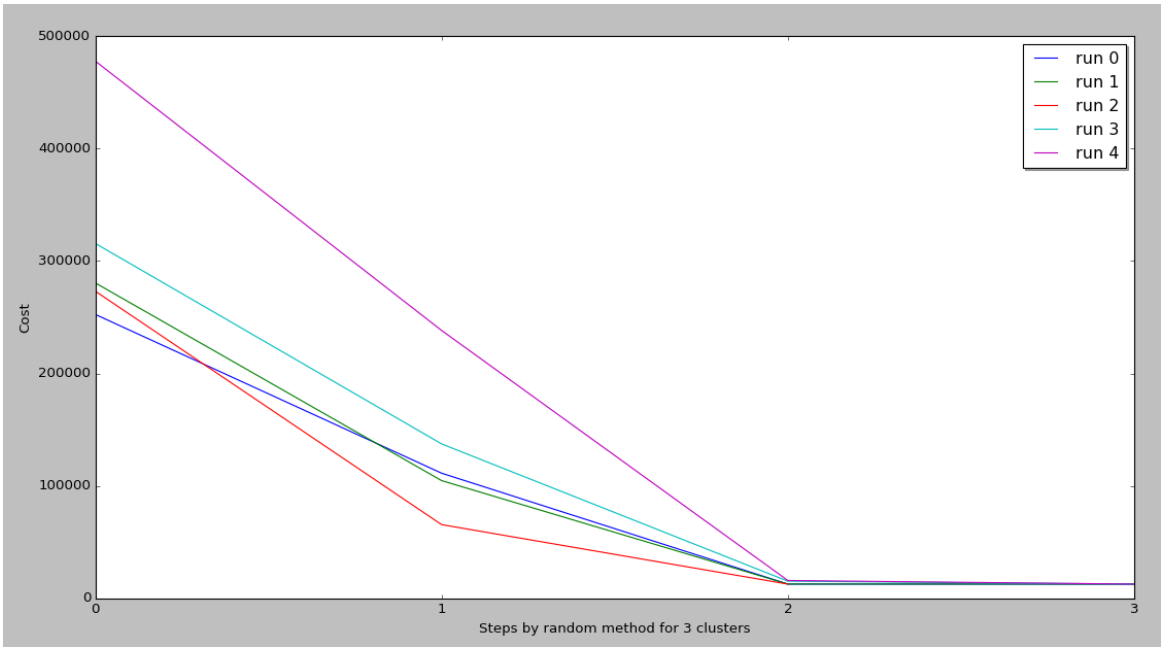



Figure 6: Convergence rate of k-means using random initialization and 3 clusters for 5 different runs

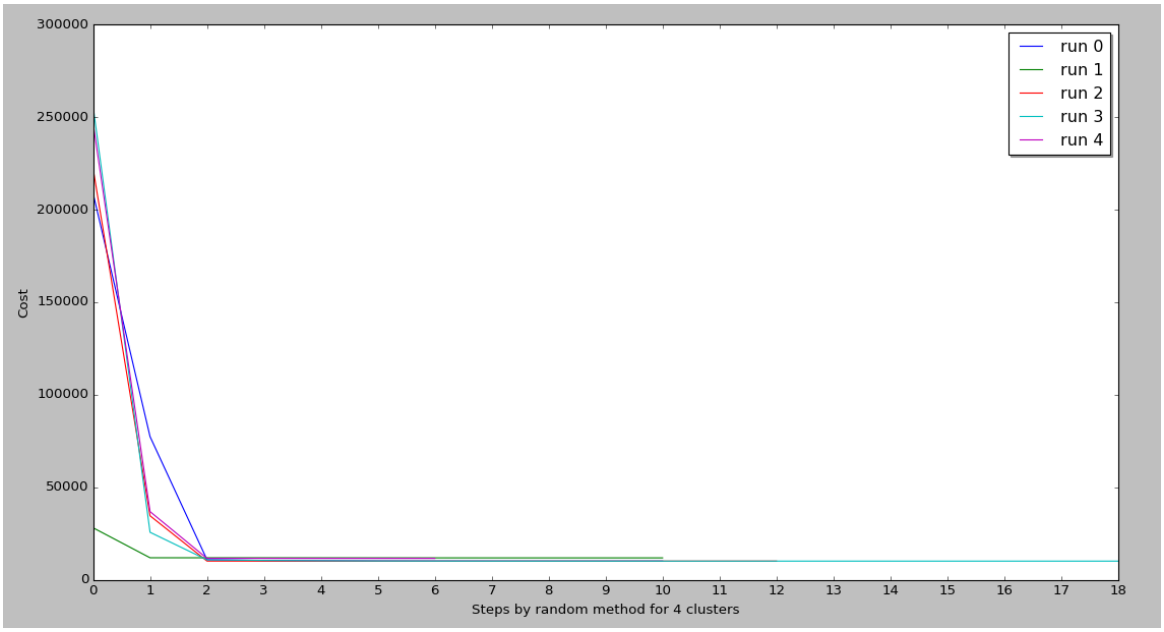



Figure 7: Convergence rate of k-means using random initialization and 4 clusters for 5 different runs

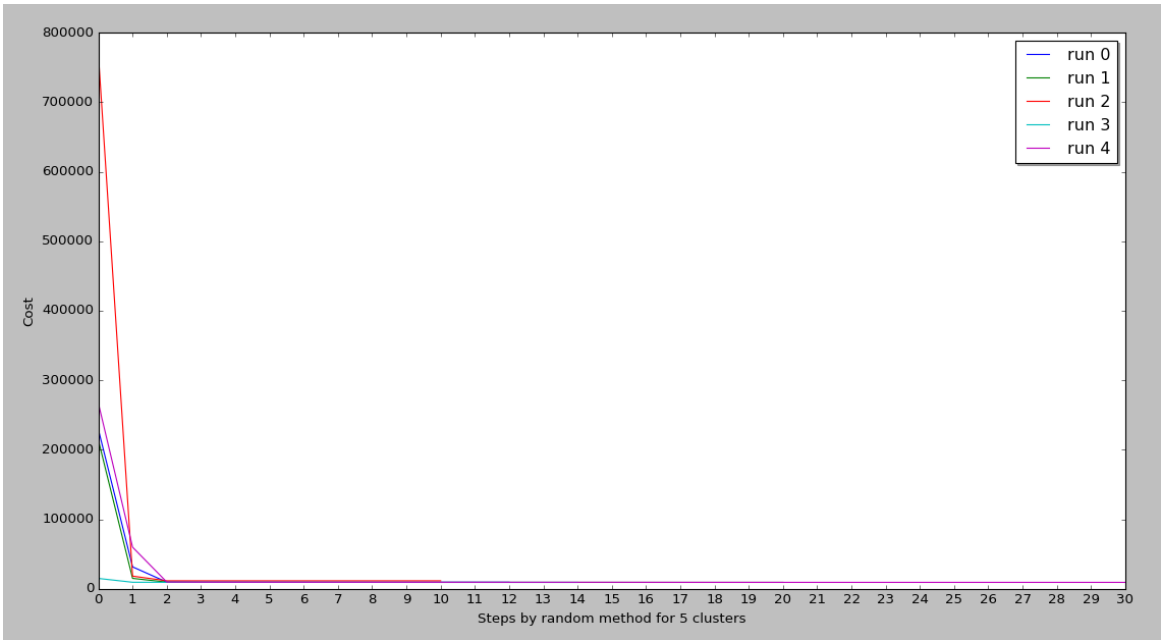



Figure 8: Convergence rate of k-means using random initialization and 5 clusters for 5 different runs

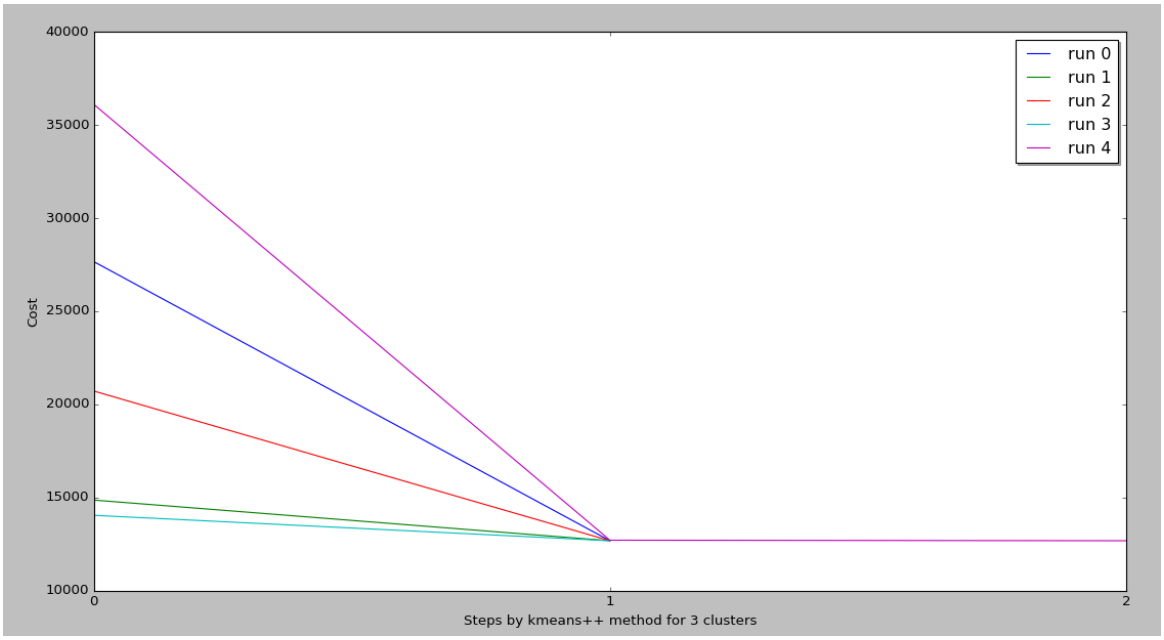



Figure 9: Convergence rate of k-means using kmeans++ initialization and 3 clusters for 5 different runs

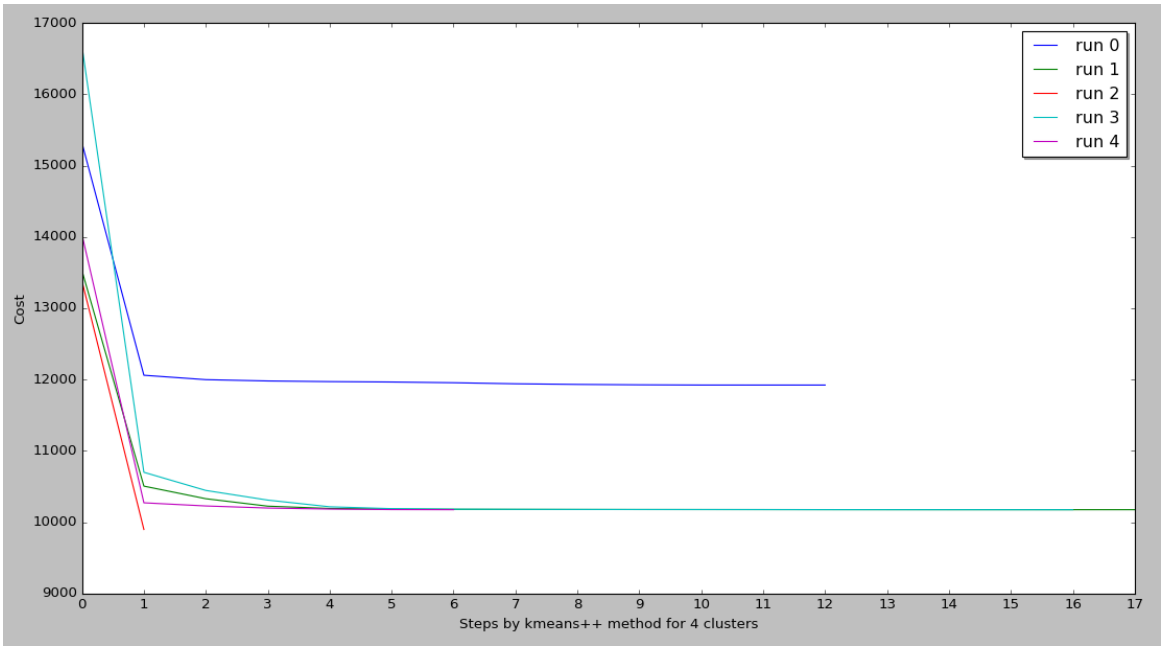



Figure 10: Convergence rate of k-means using kmeans++ initialization and 4 clusters for 5 different runs

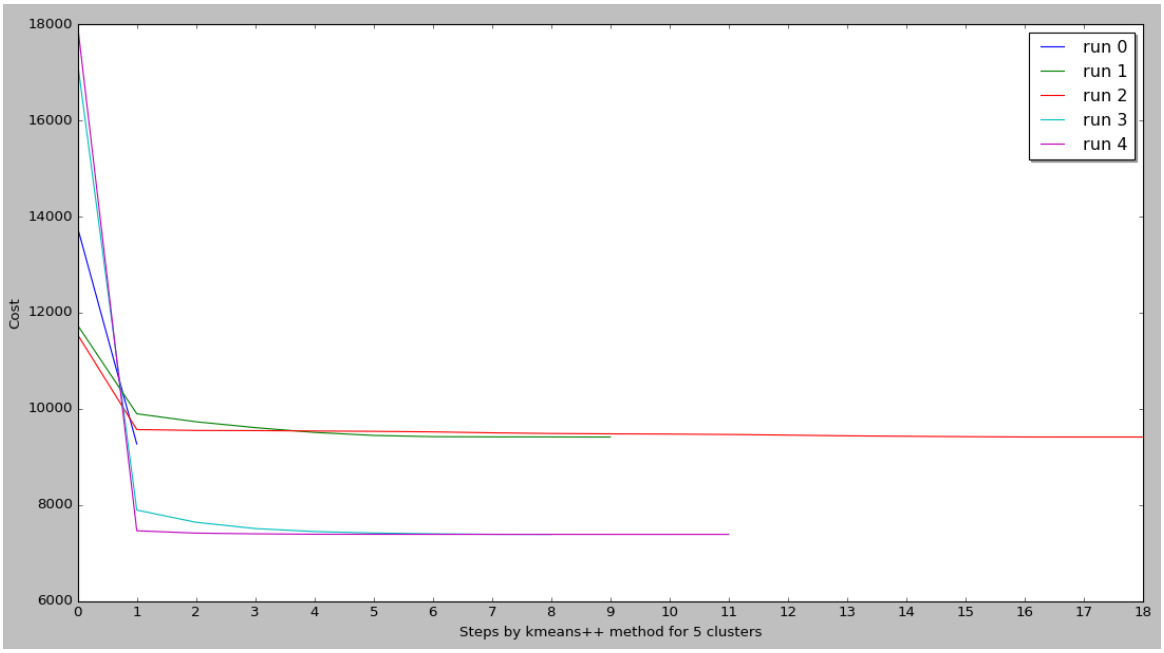
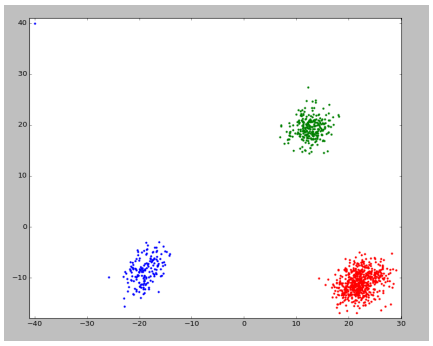
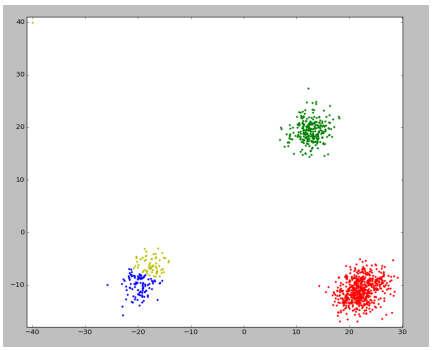


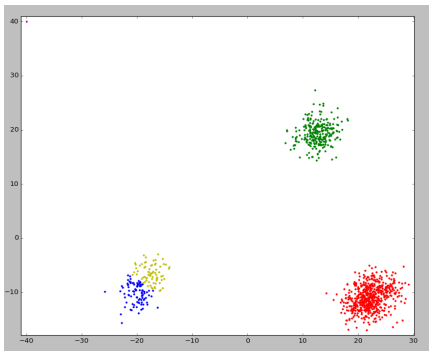

Figure 11: Convergence rate of k-means using kmeans++ initialization and 5 clusters for 5 different runs



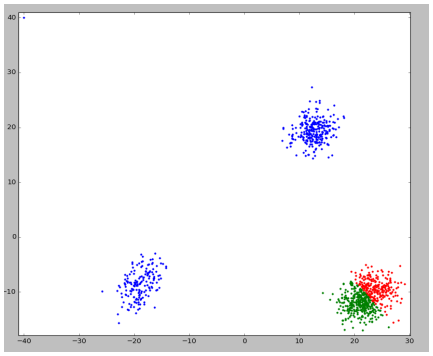
(a) Kmeans by picking first 3 points



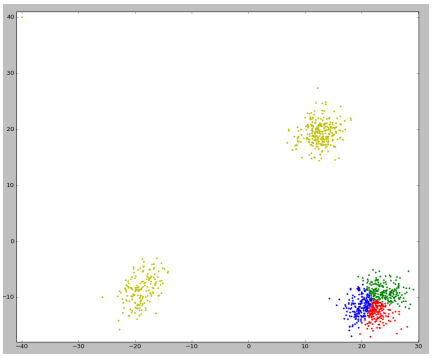
(b) Kmeans by picking first 4 points



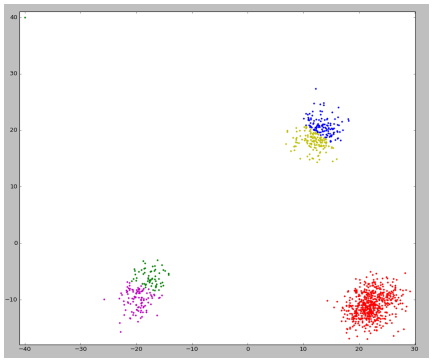
(c) Kmeans by picking first 5 points



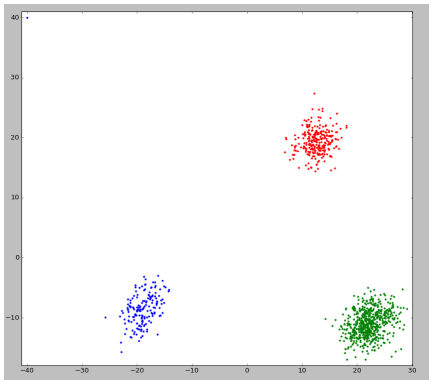
(d) Kmeans by picking first 3 random points



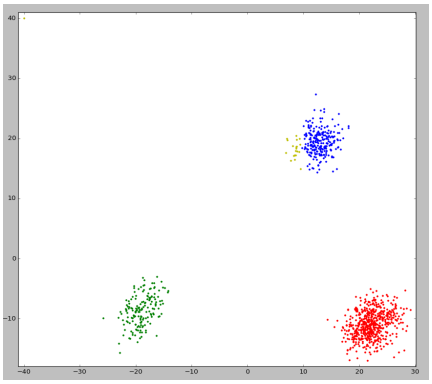
(e) Kmeans by picking first 4 random points



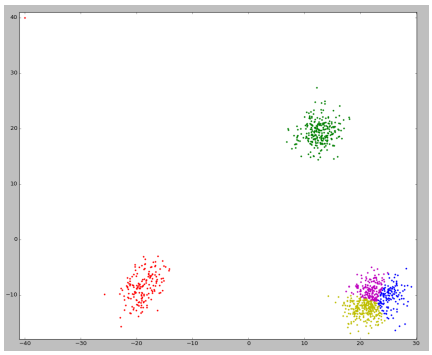
(f) Kmeans by picking first 5 random points



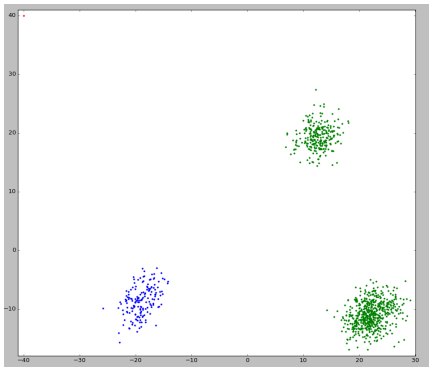
(g) Kmeans++ with k = 3



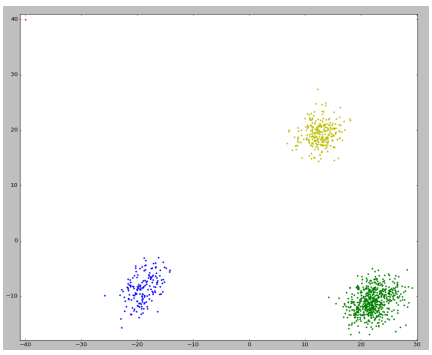
(h) Kmeans++ with k = 4



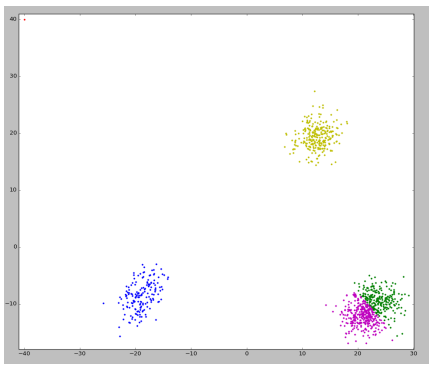
(i) Kmeans++ with k = 5



(j) Kmeans using Gonzales algorithm to choose the seeds, k = 3



(k) Kmeans using Gonzales algorithm to choose the seeds, k = 4



(l) Kmeans using Gonzales algorithm to choose the seeds, k = 5

Figure 12: K-means results with different settings

EXERCISE 2

In this excercise, we are proving that barycenter computed in the update step minimizes the cost function.

Let's first assume that $d = 1$. Then the cost function is

$$\phi(U, b) = \Sigma_{p_i \in U} ||p_i - b||^2$$

The function is minimized when its derivative is 0. In this case, we are considering the changes of the function with respect to the way we choose the center. Thus, we can take the partial derivative with respect to b.

$$\begin{aligned}
\frac{d}{d_b} \Sigma_{p_i \in U} ||p_i - b||^2 &= \Sigma_{p_i \in U} \frac{d}{d_b} ||p_i - b||^2 \\
&= \Sigma_{p_i \in U} 2 \cdot ||p_i - b|| \cdot \frac{||p_i - b||}{p_i - b} \cdot (-1) \\
&= 2\Sigma_{p_i \in U} (b - p_i)
\end{aligned}$$

The cost function is minimized when

$$\begin{aligned}
2\Sigma_{p_i \in U} (b - p_i) &= 0 \\
\Sigma_{p_i \in U} b &= \Sigma_{p_i \in U} p_i \\
b &= \frac{1}{n} \Sigma_{p_i in U} p_i
\end{aligned}$$

where $n = |U|$.

The update step also assigns each point to the nearest center, which itself minimizes the total cost because the cost of each point is minimized.

When $d > 1$, the update step in each cluster is independent from the other clusters. Thus, the sum of the costs of each clusters is also minimized when each individual cost is minimized.

Therefore, the update step actually minimizes the cost function.

EXERCISE 3

In this exercise, we show an example that the cosine distance does not follow the triangle inequality.

Let $p, q$ be 2 vectors where the angle $\alpha$ between them is $\frac{\pi}{2}$, and $r$ be the bisector of that angle. Thus, the angle $\beta$ between $p$ and $r$ and the angle $\gamma$ between $r$ and $q$ are both $\frac{\pi}{4}$. We have:

$$\begin{aligned}
dist(p, q) &= 1 - cosine(\alpha) = 1 \\
dist(p, r) + dist(r, q) &= 1 - cosine(\beta) + 1 - cosine(\gamma) \\
&= 2 - \sqrt{(2)} < 1
\end{aligned}$$

which means the cosine distance does not follow the triangle inequality in this case.

This completes the example.

EXERCISE 4

*a*

In this exercise, we consider another variant of the clustering problem, the k-median problem. The idea is similar to k-means in the sense that the distance from each point to its center is minimized.

The k-median approach can be more robust against noise because noise does not affect the order of the data, so the medians are not modified if the noise increases. If we use k-means, noise can biase the centroids.

*b*

We use a similar algorithm to k-means, just replace the means by the medians. That is, when updating a cluster, we take the point having the medians of the features as coordinates as the new center. Also, when comparing the clusters, the median points are used instead of the centroids. The median of a cluster is computed by taking the medians of the dimensions and using them as the coordinates. This approach is obviously not optimal in term of cost-minimization, as the solution is proven in Exercise 2. So we will experiment the actual cost we get in the set C3.txt.

We experiment our approach with Euclidean Distance and Manhattan Distance. The code is in our Github repo, which was mentioned in Exercise 1.

We run the k-median clustering 5 times on the dataset C3.txt for each of the metrics, and use the probabilistic approach similar to k-means++ to initialize the algorithm. For Euclidean Distance, we achieve the average cost of 5219.84203478, the standard deviation of 809.135940197, and the smallest cost of 4812.71023179. For Manhattan Distance, we achieve the average cost of 8569.62584451, the standard deviation of 24.8998452121, and the smallest cost of 8519.87957668.