TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

MV_Exercise4: Clustering

Student

Siddesh Bramarambika Shankar

Matricule Number

12329513

Teacher

Markus Vinzce

Degree course

MSE

Module

376.081

Year

2022-2023

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

For the assignment IV followed the honor code:

• Rule 1: Don’t look at solutions or code of other students; everything you submit should be your own work

• Rule 2: Don’t share your solution code with others; however, discussing ideas or general strategies is fine and encouraged

• Rule 3: If you use code from somewhere you need to cite the source directly in your code (e.g., StackOverflow).

I know that I have full responsibility for what I submit. I made sure the assignment allows the teaching staff to evaluate which competences I have acquired. I am aware that plagiarism is prohibited and may lead to failure of this course.

1.

Fig 1. K-means Clustering of image002 point cloud.

Fig 2. K-means Clustering of image000 point cloud.

The above Figure 1 shows you the clustering achieved by k-means algorithm on image002`s point cloud. As you can see some of the object`s cluster are marked as a single cluster, while the cup `s cluster is divided into 2. This is because in the context of k-means each cluster is typically represented by its centroid. The algorithm assigns each data point to the cluster whose centroid is nearest to the point. This is affected by the initial placement of the centroids, this can be seen by comparing fig 1 and fig 2 and since we are using the Forgy method, we are randomly assigning a cluster to each data point and then proceed to the update step, effectively treating the random points as the initial centroids.

Fig 3. Iterative K-means Clustering of image002 point cloud.

The above figure 3 shows the clustering generated from iterative k-means algorithm on image002`s point cloud. It is clearly better than the generic k-means algorithm as the iterative process allows the algorithm to refine the cluster assignments and centroids, leading to convergence on a set of stable clusters (i.e., further iterations do not significantly change the cluster assignment) but still this can converge to local minima, thus doesn't always find the best possible clustering.

Fig 4. G-means Clustering of image002 point cloud.

Fig 5. G-means Clustering of image000 point cloud.

The above figure 4 shows the clustering generated by the G-means algorithm on image002`s point cloud. Compared to k-means or the iterative k-means algorithm, G-means can automatically determine the number of clusters by incorporating a statistical test for the hypothesis that a subset of data follows a Gaussian distribution. If the data doesn't consist of Gaussian distributed clusters, the performance may degrade. The G-means algorithm is inherently variable in the number of clusters it identifies, which can lead to different results in terms of the number of clusters for the same dataset across different runs. This variability is primarily due to Statistical testing, initial conditions etc., this can be noticed by comparing the figure 4 and figure 5.

Fig 6. DBSCAN Clustering of image002 point cloud.

Figure 6 shows clusters generated using DBSCAN. DBSCAN is better as there is no need to number of clusters to be determined or estimated before running the algorithm, unlike k-means or G-means. It can also find arbitrarily shaped clusters. In contrast to k-means where it only finds spherical clusters. It is also less sensitive to outliers than K-means. DBSCAN has a small drawback, it requires setting the radius (eps) and the minimum number of points (minPts) parameters, which can be difficult to choose without domain knowledge and can greatly affect the outcome.

2.

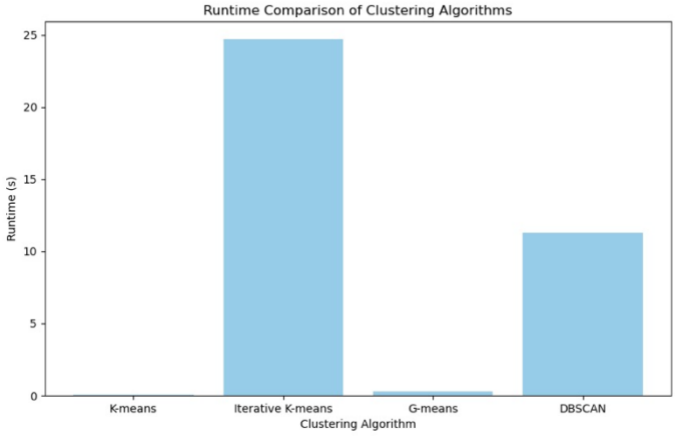

Fig 7. Runtimes of different algorithms on image000 point cloud.

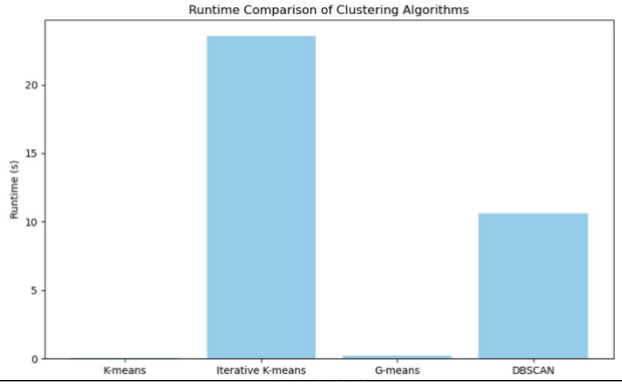

Fig 8. Runtimes of different algorithms on image002 point cloud.

Figure 7 and figure 8 above show the run time of different algorithm on different point clouds, it is evident that the iterative k-means is the most computationally expensive taking about 24 secs, while DBSCAN is the second taking up to 12 secs. K-means and G-means taking less than a sec to process. These runtimes are independent of the point clouds as they only differ by their orientations and contain the same amount of data points.

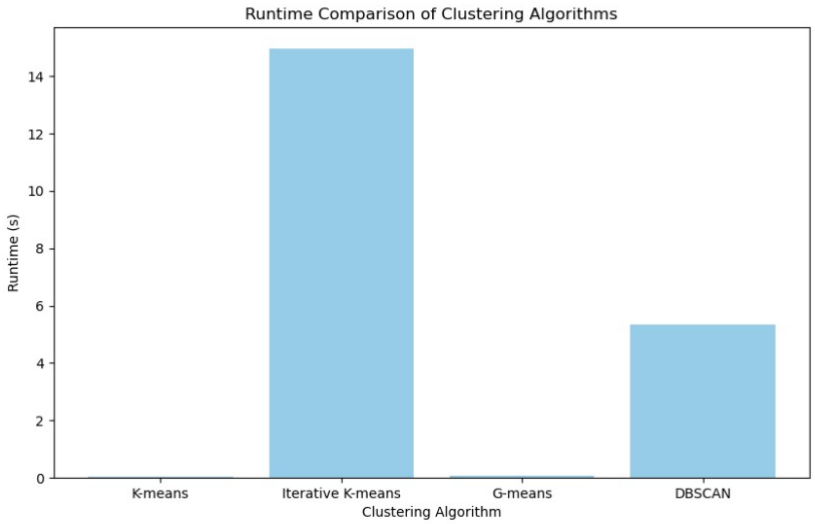

Fig 9. Runtimes of different algorithms on image000 point cloud (with 50% down sampling).

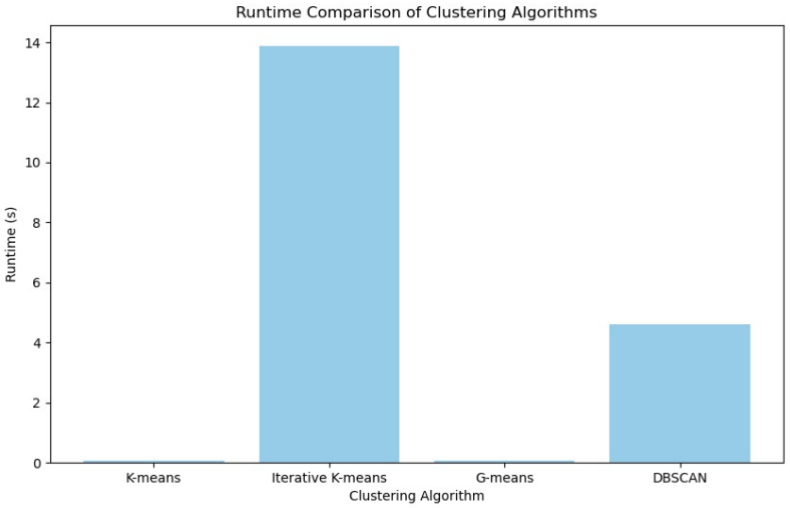

Fig 10. Runtimes of different algorithms on image002 point cloud (with 50% down sampling).

Figures 9 and 10 show the runtimes of different algorithm of two different point clouds which have been down sampled by 50%. When the point clouds are down sampled the runtimes also reduce proportionally, the above images show the runtimes of the different algorithms after the point clouds being down sampled by 50%. The runtimes have also proportionally reduced and now the iterative k-means is taking about 14 secs and DBSCAN is taking about 6 secs.

3.

Fig 11. DBSCAN Clusters of image000 point cloud (eps = 0.1, min_points = 10)

Figure 11 above show clusters generated on image000`s point cloud with eps = 0.1 and min_points = 10. We can see that increasing the radius (eps) from 0.05 to 0.1, two different objects are now being considered as one cluster. This is because in DBSCAN, a point is considered part of a cluster if there are a minimum number of points (`minPts`) within this `eps`. Clusters that were separate at a lower `eps` value can become connected into a single cluster if the distance between the clusters is less than the new `eps` value. This can create a bridge between the two clusters, causing the algorithm to label them as a single cluster.

Fig 12. DBSCAN Clusters of image000 point cloud (eps = 0.05, min_points = 20)

Figure 12 above show clusters generated on image000`s point cloud with eps = 0.05 and min_points = 20. When the min_pts is increased from 10 to 20, there is no noticeable change in the clustering, this is because the data points in each cluster may already be in sufficiently dense regions. The number of core points may have decreased but the clustering remains intact.