Project-3:Customer Profile Segmentation using Complete Linkage Agglomerative

(Bottom-Up) Clustering Technique

Group-43
Priyanshu kumar(21CS10053)

1. **Abstract**

The project aimed to design a model to cluster data points using kmeans algorithm and Complete Linkage Agglomerative with cosine function as distance function.

2. Methodology

All missing values have been replaced with the floor(mean) of the remaining values in that column.

It namely contains two modules k_means and CLAC.

k_means(k:number of clusters):

1. **__init__:**In this module, I've initialized the clusters with first k data points of dataset.
2. **Run:** It takes the number of clusters as a parameter and classifies data points into clusters based on the k-means algorithm.

In each iteration, it calculates the distance of all data points from the centroid of each cluster and appends it to the data set to which it is closest.
self.clusters_mean contains the centroid of each cluster.
clusters: contains all data points in each cluster.

CLAC:

1. **__init__:**Initializes buckets/clusters with one data point in each bucket.
2. **run(k:no. of clusters):** Each iteration merges two of the closest clusters in terms of their max distance(complete linkage). For that, it iterates over all pairs of clusters and checks the max distance between the two clusters.
   The pair of clusters for which this value is minimum are merged together. It continues until only k clusters remain.
   It doesn't use any special-purpose library.
3. **optimized_run(k:no. of clusters):** The same algorithm has been implemented as above, but here I'm storing the distance of a cluster from other clusters in sorted order in min_tree. I'm using the bisect library to implement binary search-kind of algorithm to find the index for insertion and deletion.

3. Result:

Since brute-force agglomerative algorithm takes a lot of time, the dataset size has been reduced to 1000. The whole program takes 239 seconds to complete its execution on the above dataset.

Silhouette coefficient values:

For dataset size=1000:

For k=3,silhoutte_coeff= 0.37918629668234144
For k=4,silhoutte_coeff= 0.6262807803732491
For k=5,silhoutte_coeff= 0.5600693604820105
For k=6,silhoutte_coeff= 0.6627598247616723
It is max for k=6. So, k=6 is the optimal number of clusters.
jaccard score: [0.5436507936507936, 0.6912225705329154, 0.00641025641025641, 0.3706293706293706, 0.012195121951219513, 0.21518987341772153]

For whole dataset:

Agglomerative clustering has been run using the optimized_run function since the brute-force algorithm was expected to take around 24-30 minutes. The whole program takes 168 seconds to run on the whole dataset. Since both algorithms implement the same agglomerative clustering logic apart from using sorted list in optimized_run, any one of them can be used to cluster the points. Their Jaccard score was found to be 1 for every dataset they were tested on.

For k=3,silhoutte_coeff= 0.43091655315648564
For k=4,silhoutte_coeff= 0.6347622448700103
For k=5,silhoutte_coeff= 0.5594301692307645
For k=6,silhoutte_coeff= 0.6344050375477762
It is max for k= 4. So, k=4 is the optimal number of clusters.
jaccard score: [0.5107334525939177, 0.004048582995951417, 0.0008748906386701663, 0.0035587188612099642]

4. Conclusion:

All of the missing values were replaced with the int(mean) of other values in the same column.

Since the brute-force algorithm for agglomerative clustering takes a lot of time, the dataset size has been reduced to 1000. An optimised algorithm has also been implemented, which for the dataset of size 1000, reduced the run time of agglomerative clustering by around 30.

When running kmeans algorithm on the database for both sizes, k=6 was the optimal number of clusters for size=1000 and k=4 for the whole dataset. For both sizes, k=4 and k=6 were seen to provide the highest silhouette coefficients.

When running the two implemented algorithms for agglomerative clustering on the same dataset, their Jaccard score was found to be the same for every dataset size.

Jaccard's score is found to vary a lot with each cluster; some clusters are found to be almost similar, but some have a lot of uncommon elements.
Since we’ve limited the number of iterations in kmeans to 20, it is one of the reasons for so much variance. Also, since the complete linkage agglomerative clustering uses the largest distance between the two clusters, it acts differently when compared to the kmeans algorithm, which changes clusters point-wise based on the minimum of the average distance of the point from each cluster.