Homework 3 - CS584

Hoang-Dung Bui
Team Name: bui1720
Mason ID: G01301478
Rank:
Highest Public Score for IRIS: 0.95
Highest Public Score for handwriting Number dataset: 0.55

1 Introduction

In this homework, there are two parts which process two datasets: IRIS and number images. For the IRIS datasets, I wrote the ordinal k-mean algorithm with cosine similarity, and it clustered the dataset very well and reached the accuracy of 0.95. The challenge is the handwriting number dataset. I tried three clustering algorithms: *k-mean*, *bisecting-k-mean*, and *agglomerative clustering*. The accuracy of *k-mean* and *bisecting-k-mean* were from 0.53 - 0.55. The agglomerative clustering algorithm was run on Google Colab, however, there was some problem, and it could not output the result. This will be discussed the the conclusion part.

The details of the algorithm are described in the following section.

2 IRIS dataset

Model's selection:

- Cosine similarity is used to calculate the distance. The data whose similarity is highest to a centroid will be grouped with that centroid.

The functions in the program:

- *data_preprocess*: it preprocesses the data by converting the string data from the text file into numpy array. This function also was used to process the handwriting number dataset.
- *cosin_similarity*: It calculates the cosine similarity between two vectors. If the result is close to 1, it means the two vectors are very similar, and vice versa.
- *k_mean*: It implemented the k-mean algorithm and its input are the data and number of clustering. In the first step, it initializes the centroids randomly. However, the distance among the centroids must be longer than a threshold. I used a while loop to select the centroids until the distances are larger than the threshold. Next, the cosine similarity between each datum to the centroids are calculated. The data are assigned to the centroid with greatest similarity; Then the centroids are updated by averaging all the data points. The algorithm stops as the centroids are stable.
- In the main function, it opens the data file, calls the data_preprocess function to process the raw data, then call the k_mean function.

Experiment Results: The highest F-1 score: 0.95.

The program is quite simple, and it took several seconds from my laptop to output the result. To run the program, simple call python3 iris_k_mean.py

3 Handwriting Number Dataset

This is the main and difficult part of the homework 3. I used three algorithms to test the data, they are: *k-mean*, *bisecting-k-mean*, and *agglomerative clustering*, and the result is much lower than the IRIS dataset. I also used the cosine similarity and data_preprocess functions as for IRIS data. The ordinal *k-mean* is the same to the part 1. I will described the others algorithm in this section.

The functions in the program:

- *bisecting_k_mean_algorithm*: In this algorithm, the number of cluster is step by step increased one to the data and the algorithm starts by two. After each clustering step, the cluster with largest number or highest SSE will be selected to divide. To separate the data, the ordinary k-mean algorithm is used. Adding new cluster stops as it reached the desired cluster number. The adding step is implemented by a *while loop*, which ends the cluster number reaching the desired number. In the end, the k-mean algorithm is invoked one more time to clustering data with known centroids.

- *agglomerative_clustering*: In this algorithm, I used the proximity matrix to storage the similarity among the clusters and the highest similarity will be selected, then the two corresponding clusters are grouped together. The grouping, updating the matrix and cluster set are all implemented in the *While* loop which will be terminated if the remained cluster number is equal to the desired cluster number. The similarity between two clusters are calculated between two farthest points from each cluster. The updating is first performed by removing two cluster from the cluster set and proximity matrix. The two grouping selected clusters are merged, its proximity vector are calculated, then all of them are added back the cluster set and proximity matrix, respectively.

Experiment Results: To speed up the processing time, I used *numba* package, which supports the parallel computation.

- For k-mean and bisecting-k-mean-algorithm, the accuracy reached 0.55.

- For agglomerative clustering: I tested in on a sub_data of 1000 samples, and it took 20 minutes to run on Google Colab. Then I tested on the whole dataset, however, it ran about 17 hours, and terminated without sending any output. I do not know the exact reason. That could be due to the internet connection or my laptop problem. Please, take a look at my code.

The programs, especially, agglomerative clustering is a burden for computer computation. I have tested the group similarity by MIN and Group Average, but they were both so sensitive with the noise and outlier data. Thus, I performed the MAX similarity between two group, and it provided better result.

4 Conclusion

For the k-mean and bisecting-k-mean algorithms, they are compact and their running time are much shorter than the agglomerative algorithm, even the data are extended much larger.

For Agglomerative clustering, if using and closest mean distance to determine the similarity, the result will be significant affected by the noise and outlier. Moreover, the computation's time is much longer than others algorithms. The computation time is squared proportional to the data size.

The clustering accuracy of my algorithms were still low. It is a hint that the data is needed to be preprocessed carefully.

Google Colab is a good choice to run the code, however, it relies on internet connection. For my case, I will buy a new laptop with strong GPU, and will run my codes all on my machine.