CSCI 5523: Introduction to Data Mining

Spring 2023

Assignment 5

Deadline: May. 1st 11:59 PM CT

1. Overview of the Assignment

In this Assignment, you will implement the **K-Means** and **Bradley-Fayyad-Reina (BFR)** algorithm. The goal is to help you be familiar with clustering algorithms with various distance measurements. The datasets you are going to use are synthetic datasets.

2. Requirements

2.1 Programming Requirements

a. You must use **Python** to implement all tasks. **Spark is <u>not</u> a requirement.**

2.2 Submission Platform

We will use Gradescope to automatically run and grade your submission. You must test your scripts on **the local machine** before submission.

2.3 Programming Environment

Python 3.9.12

2.4 Write your own code

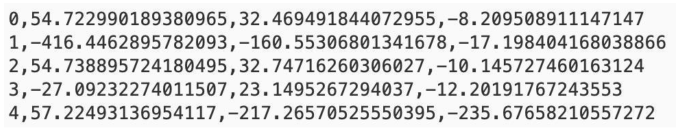
Do not share code with other students!!

For this assignment to be an effective learning experience, you must write your own code! We emphasize this point because you will be able to find Python implementations of some of the required functions on the web. Please do not look for or at any such code!

TAs will combine all the code we can find from the web (e.g., Github) as well as other students' code from this and other (previous) sections for plagiarism detection. We will report all detected plagiarism.

3. Datasets

Since the BFR algorithm has a strong assumption that the clusters are **normally distributed with independent dimensions**, we have generated synthetic datasets by initializing some random centroids and creating data points with these centroids and some standard deviations to form the clusters. We have also added some data points as **outliers. The "cluster" number of these outliers is represented by -1 (i.e., no clusters)**. Figure 1 shows an example of the data points (in CSV format). The first column is the data point index. The rest columns represent the features/dimensions of the data point.



```
0,54.722990189380965,32.469491844072955,-8.209508911147147
1,-416.4462895782093,-160.55306801341678,-17.198404168038866
2,54.738895724180495,32.74716260306027,-10.145727460163124
3,-27.09232274011507,23.1495267294037,-12.20191767243553
4,57.22493136954117,-217.26570525550395,-235.67658210557272
```

You can access and download the following datasets from google drives.

a. Folder test1 (or test2) contains multiple files of data points. We will treat these files as separate data chunks. **In each iteration**, you will load **one** file (one chunk of points) to the memory and process these data points with the BFR algorithm.

b. Files cluster1.json and cluster2.json provide the ground truth cluster for the data points in test1 and test2. The key is the data point index (as string). The value is its corresponding cluster index. The cluster of outliers are represented as -1. **Both datasets have 10 clusters (indexed as 0 to 9).**

c. We have generated 10 testing sets using similar method (two of them are provided here, i.e., test1 and test2). **Notice that the number of the dimensions, the number of the files, and the number of the data points for each dataset could vary.**

4. Tasks

You need to submit the following files on Gradescope: (all in lowercase)

a. [REQUIRED] Python scripts: bfr.py

b. [OPTIONAL] You can include other scripts to support your programs (e.g., callable functions).

You don't need to include your results. TAs will grade the assignment using a separate testing dataset (the data format remains the same).

4.1 Task description

You will write the K-Means and Bradley-Fayyad-Reina (BFR) algorithms from scratch. You should implement **K-Means** as the main-memory clustering algorithm that you will use in BFR. You will iteratively load the data points from a file and process these data points with the BFR algorithm. See below pseudocode for your reference.

```
for file in input_path:
    data_points = load(file)
    if first round:
        run K-Means for initialization
    else:
        run BFR(data_points)
```

In BFR, there are three sets of points that you need to keep track of: **Discard set (DS), Compression set (CS), Retained set (RS)**. For each cluster in the DS and CS, the cluster is summarized by:

a. **N**: The number of points
b. **SUM**: the sum of the coordinates of the points
c. **SUMSQ**: the sum of squares of coordinates

The conceptual steps of the BFR algorithm: Please refer to the slide.

The implementation details of the BFR algorithm: Please refer to the section 7 Appendix.

4.2 Execution commands **(Please make sure you use exactly the same input parameters names!)**

Python command: $ python3 bfr.py --input_path <input_path> --n_cluster <n_cluster> --out_file1 <out_file1> --out_file2 <out_file2>

We use the "**argparse**" module to parse the following arguments.

Param	<input_path>: the folder containing the files of data points <n_cluster>: the number of clusters <out_file1>: the output file of cluster results <out_file2>: the output file of intermediate results

4.3 Output format

a. You must write your clustering results in the JSON format (see Figure 2). The key is the data point index (as string). The value is its corresponding cluster index.

```
{"0": 0, "1": 0, "2": 1, "3": 1, "4": 2}
```

Figure 2: An example of the output clustering results

b. You must output the intermediate results in the CSV format (see Figure 3). Each line represents the following components in each iteration: "round id" (starting from 1), "the number of clusters in the discard set", "the total number of the discarded points", "the number of clusters in the compression set", "the total number of the compressed points", and "the number of points in the retained set". **The total number of rounds must be the number of data chunks in the folder. The total number of the discarded points are accumulated with iterations.**

```
round_id,nof_cluster_discard,nof_point_discard,nof_cluster_compression,nof_point_compression,nof_point_retained
1,10,2898,20,147,82
2,10,5326,14,256,15
3,10,7642,10,345,0
...
```

Figure 3: An example of the intermediate results

4.4 Grading

We will use the normalized mutual information (NMI) score to evaluate your clustering results. The NMI should be **above 0.8** to be considered as a successful clustering process. We will also evaluate the intermediate results to ensure your BFB is correctly processing the data points. We will grade your code with 10 cases (each is 1 pts). If you pass >=8 cases, you will receive full marks. We will compensate for the missing 2 points during manual grading phase.

5. Grading Criteria

(% penalty = % penalty of possible points you get)

1. You can use your free 5-day extension separately or together. During grading, TAs will count the number of late days **based on the submission time.** For example, if the due date is 2023/09/10 23:59:59 CDT and your submission is 2023/09/11 05:23:54 CDT, the number of late days would be 1. **TAs will record grace days by default (NO separate Emails)**. However, if you want to save it next time and treat your assignment as late submission (with some penalties), **please leave a comment in your submission.**
2. There will be no point if your submission cannot be executed on the grading platform. Please carefully follow the submission instructions in section 6
3. There is no regrading. Once the grade is posted on Canvas, we will only regrade your assignments if there is a grading error. No exceptions.
4. Homework assignments are due at 11:59 pm CT on the due date and should be submitted on Canvas. Late submissions within 24 hours of the due date will receive a 30% penalty. Late submissions after 24 hours of the due date will receive a 70% penalty. Every student has FIVE free late days for homework assignments. You can use these free late days for any reason, separately or together, to avoid the late penalty. There will be no other extensions for any reason. **You cannot use the free late days after the last day of the class (May 7th).**

6. Submission Instructions

You will submit your programming assignment via Gradescope. You can access it from Canvas. The Gradescope will be available on April. 24th.

7. Appendix

The implementation details of the BFR algorithm **(you can/should have your own implementation; this is only for reference).** Suppose the number of clusters is *K* and the number of dimensions is *d*.

a. Load the data points from one file.

b. Run K-Means on a small random sample of the data points to initialize the *K* centroids using the Euclidean distance as the similarity measurement.

c. Use the K-Means result from b to generate the DS clusters (i.e., discard points and generate statistics).

d. The initialization of DS has finished, so far, you have *K* clusters in DS.

e. Run K-Means on the rest of the data points with a large number of clusters (e.g., 5 times of *K*) to generate CS (clusters with more than one points) and RS (clusters with only one point).

f. Load the data points from next file.

g. For the new points, compare them to the clusters in DS using the Mahalanobis Distance and assign them to the nearest DS cluster if the distance is < $\alpha\sqrt{d}$.

h. For the new points that are not assigned to DS clusters, using the Mahalanobis Distance and assign the points to the nearest CS cluster if the distance is < $\alpha\sqrt{d}$.

i. For the new points that are not assigned to any clusters in DS or CS, assign them to RS.

j. Merge the data points in RS by running K-Means with a large number of clusters (e.g., 5 times of *K*) to generate CS (clusters with more than one points) and RS (clusters with only one point).

k. Merge clusters in CS that have a Mahalanobis Distance < $\alpha\sqrt{d}$.

l. Repeat the steps f – k until all the files are processed.

m. If this is the last round (after processing the last chunk of data points), merge clusters in CS with the clusters in DS that have a Mahalanobis Distance < $\alpha\sqrt{d}$.

(α is a hyper-parameter, you can choose it to be around 2, 3 or 4)