INF553 Foundations and Applications of Data Mining

Spring 2019

Assignment 5 Clustering

Deadline: Apr. 13th 11:59 PM PST

1. Overview of the Assignment

In Assignment 5, 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. Assignment Requirements

2.1 Programming Language and Library Requirements

a. You must use **Python** to implement all the tasks. You can only use standard Python libraries (i.e., external libraries like numpy or pandas are not allowed). **Spark RDD is optional for Python**. If you want to use Spark, please specify the following environment in your code:

```
os.environ['PYSPARK_PYTHON'] = '/usr/local/bin/python3.6'
os.environ['PYSPARK_DRIVER_PYTHON'] = '/usr/local/bin/python3.6'
```

a. There will be 10% bonus for Scala implementation in each task. **Spark RDD is required for Scala**. You can get the bonus only when both Python and Scala implementations are correct.

b. Spark DataFrame and DataSet are not allowed.

2.2 Programming Environment

Python 3.6, Scala 2.11, and Spark 2.3.0

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

2.3 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 to the university.

3. Dataset

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
```

Figure 1: An example of the data points with 3 dimensions

You can access and download the following datasets either under the directory on Vocareum: resource/asnlib/publicdata/ or Google Drive (USC email only): https://drive.google.com/open?id=1rQJbHRBewL_W5VQuQiblsW1t1DR4k3YB

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.**

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. Task

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

a. [REQUIRED] Python scripts: bfr.py

b. [REQUIRED FOR SCALA] Scala scripts: bfr.scala; one Jar package: hw5.jar

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

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:

N: The number of points

SUM: the sum of the coordinates of the points

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

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

Scala command: $ spark-submit --class bfr hw5.jar <input_ path> <n_cluster> <out_file1> <out_file2>

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

4.3 Grading

We will use normalized mutual information (NMI) score to evaluate your clustering results. The NMI should be **above 0.8** for all the datasets. 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 0.8 pts).

5. About Vocareum

a. Your code can directly access the datasets under the directory: ../resource/asnlib/publicdata/

b. You should upload the required files under your workspace: work/

c. You must test your scripts on both the local machine and the Vocareum terminal before submission.

d. During submission period, the Vocareum will run and evaluate the results for the two given datasets.

e. You will receive a submission report after Vocareum finishes executing your scripts. The submission report should show **the accuracy information** for each dataset.

f. The total execution time of submission period should be less than 600 seconds. The execution time of grading period need to be less than 3,000 seconds.

g. Please start your assignment early! You can resubmit any script on Vocareum. We will only grade on your last submission.

6. Grading Criteria

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

a. You can use your free 5-day extension separately or together. You must submit a late-day request via https://forms.gle/worKTbCRBWKQ6jqu6. This form is recording the number of late days you use for each assignment. By default, we will not count the late days if no request submitted.

b. There will be 10% bonus for each task if your Scala implementations are correct. Only when your Python results are correct, the bonus of Scala will be calculated. There is no partial point for Scala.

c. There will be no point if your submission cannot be executed on Vocareum.

d. There is no regrading. Once the grade is posted on the Blackboard, we will only regrade your assignments if there is a grading error. No exceptions.

e. There will be 20% penalty for the late submission within one week and no point after that.

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)