DSCI553 Foundations and Applications of Data Mining

Fall 2020

Assignment 6 Clustering (Optional)

Deadline: Nov. 17th 11:59 PM PST

NO LATE SUBMISSIONS

1. Overview of the Assignment

In Assignment 6, 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. **We will accept the partial submission for any of two methods.**

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 2.3.2

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

Dataset for the BFR algorithm

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/

Google Drive link for the data is here
https://drive.google.com/drive/folders/1Hknh0xmC80Fll7t8R38tv34eVqORlabH?usp=sharing

We will use **2** testing data set for k-means and **10** testing sets[1] using a similar method. Notice that the number of the dimensions, the number of the files, and the number of the data points for **each dataset are different.**

Dataset for the k-means algorithm

We will use **2** testing data sets for k-means that have the same format as the dataset for the BFR algorithm. We will grade your implementation of k-means algorithm, but as mentioned above, we will use the blinded dataset to test your implementation of the BFR algorithm.

K-means algorithm

a. We will use (kmeans_data1.tex, kmeans_data1.tex) as test data for k-means algorithm. Your code will read that file and cluster the users based on the data points in that data set.

b. Files cluster*.json (kmeans_cluster1.json, kmeans_cluster2.json) provide the ground truth cluster for the data points in test*. The key is the data point index (as string). The value is its corresponding cluster index. The cluster of outliers are represented as -1. **Submission script will only grade**

[1] Five of them are provided to you, the other five will be used as blind dataset.

kmeans_cluster1 **and** kmeans_cluster2 **for you.** You are provided with cluster*.json for 2 datasets for yourself to calculate the accuracy. The number of clusters for each dataset are: **test1: 3, test2: 10.**

Bradley-Fayyad-Reina (BFR) algorithm

c. Folders test* (test1, test2, etc.) contain 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.

d. Files cluster*.json (cluster1.json, cluster2.json, etc.) provide the ground truth cluster for the data points in test*. The key is the data point index (as string). The value is its corresponding cluster index. The cluster of outliers are represented as -1. **Submission script will only grade test1 and test2 for you.** You are provided with cluster*.json for all 5 datasets for yourself to calculate the accuracy. The number of clusters for each dataset are: **test1: 10, test2: 10, test3: 5, test4: 8, test5: 15.**

4. Task

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

a. [REQUIRED] Python scripts: **kmeans.py, bfr.py**

b. [REQUIRED FOR SCALA] Scala scripts: **kmeans.scala, bfr.scala**; one Jar package: **hw6.jar**

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

4.1 Task description

K-Means algorithm (1 pt)

You will write the K-Means algorithm from scratch. Your code will firstly read the data set and the number of clusters as inputs. Then, it will cluster the users based on the datapoint provided in that dataset. As a output, your code will return the clustering results, following **4.3 Output format (a)**.

Bradley-Fayyad-Reina (BFR) algorithm (3 pts)

You will write 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 a general idea.

```
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 kmeans.py <input_file> <n_cluster> <out_file1>

Scala command: $ spark-submit --class kmeans hw5.jar <input_file> <n_cluster> <out_file1>

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. **Both for K-means and BFR:** 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 (Any order of cluster indices is fine).

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

Figure 2: An example of the output clustering results

b. **Only for BFR**: 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". We will mainly measure following criteria: **The total number of rounds must be the number of data files/chunks in the folder**. **The total number of the discarded points should only go up 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 2 cases for k-means, and all the 10 cases for BFR.

5. About Vocareum

a. Your code can directly access the datasets under the directory: /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 (2 datasets). The execution time of grading period need to be less than 3,000 seconds (10 datasets).

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 **cannot use** your free late days on this assignment as its deadline is the end data of the class.

b. Late submission will not be accepted for the same reason.

c. There will be a 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.

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

e. 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.

7. Appendix

Here are some implementation details of the BFR algorithm **(There are multiple ways to implement the BFR algorithm. Here is a method you can use for a reference. Your own implementation can be different from this).** Suppose the number of clusters is ***K*** and the number of dimensions is ***d***.

1. Load the data points from one file.

2. Run a clustering algorithm: K-Means or an improved version of K-Means (e.g. K-Means++) on the data points or a random sample of the data points (in case your code cannot handle all the data points in the first file). Initialize the algorithm with a large number of centroids (e.g. 3 or 5 times of ***K***) and use Euclidean distance as the similarity measurement.
3. Among the result clusters of step 2, move all the clusters that contain only one or very few points (up to you) to **RS** as outliers. You will now have 2 groups of data points: outlier data points and non-outlier data points.
4. Run the clustering algorithm again to cluster the non-outlier data points into ***K*** clusters. Use these ***K*** clusters as your **DS**. Discard their points and generate the **DS** statistics.
5. Run the clustering algorithm again to cluster the outlier data points into a large number of clusters (e.g. 3 or 5 time of ***K***). Generate **CS** and their statistics from the clusters with more than 1 data point and use the remaining as your new **RS**.

The initialization of DS has finished. So far you have ***K*** **DS** (from step 4), some number of **CS** (from step 5) and some number of **RS** (from step 5).

6. Load the data points from another file (if you used a portion of the first file in step 2, load the remaining data points from the first file).
7. For the new data points, compare them to each of the **DS** using the Mahalanobis Distance and assign them to the nearest **DS** clusters if the distance is < $\alpha\sqrt{d}$ (e.g. $2\sqrt{d}$).
8. For the new data points that are not assigned to any **DS** cluster, compare them to each of the **CS** using the Mahalanobis Distance and assign them to the nearest **CS** clusters if the distance is < $\alpha\sqrt{d}$ (e.g. $2\sqrt{d}$).
9. For the new data points that are not assigned to any **DS** cluster or **CS** cluster, add them to your **RS**.
10. Run the clustering algorithm on the RS with a large number of centroids (e.g. 3 or 5 time of ***K***). Generate **CS** and their statistics from the clusters with more than 1 data point and add them to your existing **CS** list. Use the remaining points as your new **RS**.
11. Merge CS clusters that have a Mahalanobis Distance < $\alpha\sqrt{d}$ (e.g. $2\sqrt{d}$).
12. Output your intermediate result after all the data points in the data file have been evaluated.

Repeat step 6 to 12.

If this is the last run (after the last chunk of data), merge your CS and RS clusters into the closest DS clusters.