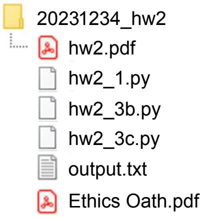
EE412 Foundation of Big Data Analytics, Fall 2023
HW2

Due date: 11/02/2023 (11:59pm)

Submission instructions: Use KAIST KLMS to submit your homeworks. Your submission should be one gzipped tar file whose name is `YourStudentID_hw2.tar.gz`. For example, if your student ID is 20231234, and it is for homework #2, please name the file as `20231234_hw2.tar.gz`. You can also use these extensions: tar, gz, zip, tar.zip. Do not use other options not mentioned here.

Your zip file should contain total six files; one PDF file for writeup answers (`hw2.pdf`), one text file for problem 3-c (`output.txt`), three python codes (`hw2_1.py`, `hw2_3b.py`, `hw2_3c.py`), and the Ethics Oath pdf file. Before zipping your files, please make a directory named `YourStudentID_hw2` and put your files in the directory. Then, please compress the directory to make a zipped file.



```
20231234_hw2
    hw2.pdf
    hw2_1.py
    hw2_3b.py
    hw2_3c.py
    output.txt
    Ethics Oath.pdf
```

If you do not follow this format, we will **deduct** 1 point of total score per mistake.

Submitting writeup: Prepare answers to the homework questions into a single PDF file. You can use the following template. Please write as succinctly as possible.
In some exercises, we recommend you to use Python. If you use Python in the exercise, **please attach your code to your document.**(`hw2.pdf`)

Submitting code: Each problem is accompanied by a programming part. Put all the code for each question into a single file. Good coding style (including comments) will be one criterion for grading. As long as your code is clear, there will be no penalty. Please make sure your code is well structured and has descriptive comments.
We will **deduct** 1 point per line if you print any other line except the answer (i.e., elapsed time, description, and etc.).

Ethics Oath: For every homework submission, please fill out and submit the **PDF** version of this document that pledges your honor that you did not violate any ethics rules required by this course and KAIST. You can either scan a printed version into a PDF file or make the Word document into a PDF file after filling it out. Please sign on the document and submit it along with your other files.

Discussions with other people are permitted and encouraged. However, when the time comes to write your solution, such discussions (except with course staff members) are no longer appropriate: you must write down your own solutions independently. If you received any help, you must specify on the top of your written homework any individuals from whom you received help, and the nature of the help that you received. *Do not, under any circumstances, copy another person's solution.* We check all submissions for plagiarism and take any violations seriously.

[**Note**] There are answers to frequently asked questions at the end of this document. Please refer to it.

1 Clustering (30 points)

(a) [5 pts] Solve the following problem, which is based on the exercises in the Mining of Massive Datasets 2nd edition (MMDS) textbook.

- Exercise 7.2.6

  - **Edit distance**: The distance between two strings $x = x_1x_2...x_n$ and $y = y_1y_2...y_m$ is the smallest number of insertions and deletions of single characters that will convert x to y.
    For example, edit distance between $abcde$ and $acfdeg$ is 3.

  Consider the space of strings with **edit distance** as the distance measure. Give an example of a set of strings such that if we choose the clustroid by minimizing the sum of the distances to the other points we get one point as the clustroid, but if we choose the clustroid by minimizing the maximum distance to the other points, another point becomes the clustroid.

(b) [25 pts] Implement the k-Means algorithm using Spark

We will implement the classic k-Means algorithm using Spark described in MMDS Chapter 7.3. In particular, we are interested in finding the right k value (i.e., the number of clusters) for the given data. For the distance measure, always use the Euclidean distance defined in Chapter 7.1.3. Your code should be finished less than **30 mins**.

The dataset can be downloaded from KLMS:
kmeans.txt[1]

This file consists of 4,601 lines where each line represent a document with 58 features in the following format:

[1] This dataset is from the Stanford Large Network Dataset Collection

```
<FEATURE 1> <FEATURE 2> ... <FEATURE 58>
```

Also, although not necessary in this assignment, the 58 words used as the 58 features can be downloaded from KLMS:
vocab.txt

Implement the initialization of clusters for k-Means using a variant of the first approach in Chapter 7.3.2 using a sequential algorithm in plain Python and **not** Spark[2]. Instead of picking the first point at random, however, use the first point in the dataset, which has already been randomized. (We want the results to be deterministic.)

Then implement the k-Means **algorithm in Chapter 7.3.1 using Spark**. (No need to take the optional step of fixing the centroids of the clusters and reassigning points.) Finally, compute the average diameter of the clusters. The diameter of a cluster is defined in Chapter 7.2.3.
When implementing the algorithm in Figure 7.7, please ignore the line "Adjust the centroid of that cluster to account for p;" in the for loop as it does not allow the algorithm to be parallelized.

Please **use command-line** arguments to obtain paths for data files and k value. (Do not fix the paths in your code.) For example, run:

```
bin/spark-submit hw2_1.py path/to/kmeans.txt k_value
```

After run, your code should **print the average diameter** of the given number-of-cluster (`k_value` in the command-line).

Run your implementation using different k values (1,2,4,8,...), as described in Chapter 7.3.3, and plot a number-of-clusters vs. average diameter graph like Figure 7.9, and find a good value of k. To plot the graph, you can use other programs.
Test cases for your implementation (just for checking values, not print format) - (k-value, average diameter): (3, 4225.7734...), (5, 1526.2137...)
Please submit the following results:

- The number-of-clusters vs. average diameter plot. (Please put your graph in `hw2.pdf`)
- The k value with an explanation why it is good for this data.
  Please choose the best k value based on your thought. You will get full points if the reasoning is logical. (Please put the answer in `hw2.pdf`)
- Your source code in one file. (`hw2_1.py`)

2 Dimensionality Reduction (20 points)

Solve the following problems, which are based on the exercises in the MMDS textbook. In some exercises, we recommend you to use Python. If you use Python in the exercise,

[2] For students who are not satisfied with this non-parallel approach, check out the k-Means++ paper http://theory.stanford.edu/ sergei/papers/vldb12-kmpar.pdf

please attach your code to your document(`hw2.pdf`)

- Exercise 11.1.7
  For the matrix of Exercise 11.1.5:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}$$

  (We recommend you to use **Python**, but you **CANNOT** use `numpy.linalg.eig()` function in this exercise.)

  (a) Starting with a vector of three 1's, use power iteration to find an approximate value of the principal eigenvector.

  (b) Compute an estimate the principal eigenvalue for the matrix.

  (c) Construct a new matrix by subtracting out the effect of the principal eigenpair, as in Section 11.1.3.

  (d) From your matrix of (c), find the second eigenpair for the original matrix of Exercise 11.1.5.

  (e) Repeat (c) and (d) to find the third eigenpair for the original matrix.

- Exercise 11.3.1
  Consider the matrix M below. It has rank 2, as you can see by observing that the first column plus the third column minus twice the second column equals $\mathbf{0}$.
  (We recommend you to use **Python**, but you **CANNOT** use `numpy.linalg.svd()` function in this exercise.)

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 5 & 4 & 3 \\ 0 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix}$$

  (a) Compute the matrices M^TM and MM^T.

  (b) Find the eigenpairs (eigenvalues, eigenvectors) for your matrices of part (a) using Python NumPy function (`numpy.linalg.eig()`).

  (c) Find the SVD for the original matrix M from parts (b). Note that there are only two nonzero eigenvalues, so your matrix Σ should have only two singular values, while U and V have only two columns.

  (d) Set your smaller singular value to 0 and compute the one-dimensional approximation to the matrix M.

  (e) How much of the energy of the original singular values is retained by the one-dimensional approximation? (Hint: energy = sum of the squares of the singular values)

Table 1: A utility matrix for exercises

	a	b	c	d	e	f	g	h
A	4	5		5	1		3	2
B		3	4	3	1	2	1	
C	2		1	3		4	5	3

3 Recommendation Systems (50 points)

(a) [10 pts] Solve the following problems, which are based on the exercises in the MMDS textbook.
You **CANNOT** use **Python** in this part. Please write your answer with the **solving process** in your document(`hw2.pdf`).

- Exercise 9.3.1
  Table 1 is a utility matrix, representing the ratings, on a 1–5 star scale, of eight items, a through h, by three users A, B, and C. Compute the following from the data of this matrix.

  (a) Treating the utility matrix as boolean, compute the Jaccard distance between each pair of users. (Hint: Blank is false, and others are true.)

  (b) Repeat Part (a), but use the cosine distance.

  (c) Treat ratings of 3, 4, and 5 as 1 and 1, 2, and blank as 0. Compute the Jaccard distance between each pair of users.

  (d) Repeat Part (c), but use the cosine distance.

  (e) Normalize the matrix by subtracting from each nonblank entry the average value for its user.

  (f) Using the normalized matrix from Part (e), compute the cosine distance between each pair of users.

- Exercise 9.3.2
  In this exercise, we cluster items in the matrix of Table 1. Do the following steps.

  (a) Cluster the eight items hierarchically into four clusters. The following method should be used to cluster. Replace all 3's, 4's, and 5's by 1 and replace 1's, 2's, and blanks by 0. use the Jaccard distance to measure the distance between the resulting column vectors. For clusters of more than one element, take the distance between clusters to be the minimum distance between pairs of elements, one from each cluster. (It is okay that you only choose one possible result with the detailed explanation.)

  (b) Then, construct from the original matrix of Table 1 a new matrix whose rows correspond to users, as before, and whose columns correspond to clusters. Compute the entry for a user and cluster of items by averaging the nonblank entries for that user and all the items in the cluster.

  (c) Compute the cosine distance between each pair of users, according to your matrix from Part (b).

(b) [20 pts] Implement collaborative filtering

We will implement user-based and item-based collaborative filtering and run it on a real movie dataset.

The dataset can be downloaded from KLMS:
ratings.txt[3]

This file consists of about 90,000 lines where each line is in the following format:

```
<USER ID>,<MOVIE ID>,<RATING>,<TIMESTAMP>
```

where `<USER ID>` is a user ID, `<MOVIE ID>` is a movie ID, `<RATING>` is a rating of that user to the movie (1–5 stars), and `<TIMESTAMP>` is the timestamp of the rating.

Although not necessary for this problem (but possibly useful for 3(c)), the movie information can be downloaded from KLMS:
movies.txt

Each line is in the following format:

```
<MOVIE ID>,<MOVIE TITLE>,<MOVIE GENRE>
```

Implement the collaborative filtering algorithm as described in Chapter 9.3. As a first step, compute the utility matrix M with normalized ratings as in Exercise 9.3.1(e) and Figure 9.6 from `ratings.txt`. That is, for normalization, subtract from each rating the average rating of that user. However, **you use the normalized ratings for only finding similar users or items. After finding similar users or items, you have to use values of the original matrix in the range of [1,5] when predicting the ratings.** Then implement the following two methods described in Chapter 9.3.2 for predicting the rating of user U to movie I:

- *User-based:* Find the 10 most similar users to U using the cosine distance for measuring similarity. Then compute the average ratings for I, only for the similar users who have rated I (i.e., similar users who have not rated I are not counted).
- *Item-based:* Find the 10 most similar movies to I. While there are multiple ways to compute movie-movie similarity, compute the cosine distance of their rating vectors in M. Then take the average of the ratings that U gave to those similar movies (i.e., movies that were not rated by U are not counted). *Note: this method takes longer than the user-based method.*

For example with user based method, let's assume that the users 1, 2, 3, ..., 10 are the most similar users to the target user U. We want to predict U's rating for some movie I. If only the users 1, 2, 3 rated I while other users(e.g. user 4, 5, ..., 10)

[3] This dataset is derived from the MovieLens dataset

didn't, then the predicted result should be the average of the user 1, 2, 3's rating to I.

Please **use command-line** arguments to obtain paths for data file. (Do not fix the paths in your code.) For example, run:

```
python hw2_3b.py path/to/ratings.txt
```

Consider the user U whose ID is 600 in `ratings.txt`. We have zeroed out U's ratings for movies ID=1 to ID=1000. Please submit the following results:

- The top-5 movies among movies 1 to 1000 for U with the highest predicted ratings using the *user-based* method. **Print** the result in descending order, i.e., highest rating first, in the following format for each line:
  `<MOVIE ID><TAB><PREDICTED RATING>`
- The top-5 movies among movies 1 to 1000 for U with the highest predicted ratings using the *item-based* method. When you find similar items, you consider all movies except ID=1 to ID=1000. **Print** the result in descending order, i.e., highest rating first, in the following format for each line:
  `<MOVIE ID><TAB><PREDICTED RATING>`
- Your source code in one file. (`hw2_3b.py`)

(c) [20 pts] Movie Recommendation Challenge

Similar to the NetFlix Challenge, we will have a EE412 Movie Recommendation Challenge. Improve the Recommendation System you implemented in 3(b) using any method you like (e.g., better normalization, clustering, different similarity measures, UV-decomposition, or even winning techniques used in the NetFlix Challenge) and predict the ratings in the test set which is uploaded at KLMS:
ratings_test.txt

This file consists of 10,000 lines where each line is in the following format:

```
<USER ID>,<MOVIE ID>,,<TIMESTAMP>
```

Please fill in the `<RATING>` column (the ratings can be real numbers) so that each line in the output file is in the following format:

```
<USER ID>,<MOVIE ID>,<RATING>,<TIMESTAMP>
```

We will compute the RMSE against the actual ratings, and award points accordingly. That is, the lower the RMSE, the more points you will get. Even trivial solutions will get some points.

Please **use command-line** arguments to obtain paths for data file. (Do not fix the paths in your code.) For example, run:

```
python hw2_3c.py path/to/ratings.txt path/to/ratings_test.txt
```

After run, your code should **save an output.txt file in the current directory** automatically.

Please submit the following results:

- The output ratings file `output.txt`.
  As mentioned above, your code must save this file automatically. Furthermore, you should also submit the `output.txt` file.
- A description of the techniques used. (Please put the answer in `hw2.pdf`)
- Your source code in one file. (`hw2_3c.py`)

Answers to Frequently Asked Questions

- About using **external libraries**: We will only allow you to import **pyspark, numpy, re, sys, math, csv, time, and os** in your code. You are not allowed to use any other libraries.
- If you do not know how to download text files after opening links for text files, just click the right button on the mouse and choose download among options.
- Please use the print **built-in function** in Python to produce outputs.
- Also, note that `<TAB>` means `\t` in the Python print function.
- Questions about Problem 2
  - You do not need to care about the case that points are far from multiple centroids equivalently.
  - For Exercise 11.3.1-(c), you can change the sign of matrix V to recover the matrix M. In this case, you can explain the reason in the pdf file.
- Questions about Problem 3
  - In *User-based* method case, you first find the 10 most similar users to U. Then compute the average ratings for I. When you compute the average, you have to except the similar users who do not rate I.
  - No additional test set is provided for validation. Therefore, you have to **split the data** and make validation set for testing your algorithm.
  - If there are multiple movies scored 5.0, you should sort "Movie ID" in ascending order and then pick top-5.
  - If there are the movies that are rated by none of users, you can exclude them or give zero score.
  - Time Limit: 3-(b) less than 1 hour, 3-(c) No limit
  - A hint for efficient implementation: Let q, v_i is a vector of dimension d, for $i = 1, ..., 1000$. Let $V = [v_1; ...; v_{1000}]$ is a matrix which consists of v_i columns. Then, 1 matrix-vector multiplication ($q * V$) is quite faster than 1000 dot products ($q * v_i$ for all i) in many matrix libraries such as numpy.
  - If you want to utilize movies.txt file for 3-(c), you can use the path in code or modify the command line (python hw2_3c.py rating.txt movies.txt rating_test). In this case, you have to mention clearly in the pdf file.