INF 553 – Spring 2018

Homework #3: Collaborative Filtering

Due: March 22, Thursday
100 points

In this homework, we consider the latent factor modeling of utility matrix M (e.g., a rating matrix where rows represent users and columns products such as movies). Recall that in latent factor modeling, the goal is to decompose M into lower-rank matrices U and V such that the difference between M and UV is minimized.

The homework consists of the following **two** tasks:

1. [80 points] Implement the incremental UV decomposition algorithm as discussed in class (and described in the text), where the element is learned one at a time.

   Assume initially all elements in latent factor matrix U and V are 1's.

   The learning starts with learning elements in U row by row, i.e., U[1,1], U[1,2], …, U[2, 1], … It then moves on to learn elements in V column by column, i.e., V[1,1], V[2,1], …, V[1, 2], … When learning an element, it uses the latest value learned for all other elements. It should compute the optimal value for the element to minimize the current RMSE as described in class.

   The learning process stops after a specified number of iterations, where a round of learning all elements in U and V is one iteration.

   The algorithm should output RMSE after each iteration (remember that the mean is computed based on non-blank elements in the input matrix M).

   Write your code in Python (**do not use Spark here**). Name your algorithm, **FirstName_LastName_uv.py**. If you are using Python 3, please name it to **FirstName_LastName_uv3.py.** It can be invoked as follows.

   Execution format:

   python uv.py ratings-file n m f k

   - "ratings-file" is a subset of **MovieLens** latest dataset (https://grouplens.org/datasets/movielens/) that has 100k movie ratings, from which you need to build your utility matrix described above. It comes in sparse format.

```
userId,movieId,rating,timestamp
1,31,2.5,1260759144
1,1029,3,1260759179
1,1061,3,1260759182
2,10,4,835355493
2,17,5,835355681
```

You need to ignore entire last column – *timestamp,* and first row that has header information so you get utility matrix like below

```
1,31,2.5
1,1029,3
1,1061,3
2,10,4
2,17,5
```

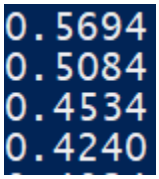
- n is the number of rows (users) of the matrix, while m is the number of columns (movies).
- f is the number of dimensions/factors in the factor model. That is, U is n-by-f matrix, while V is f-by-m matrix.
- k is the number of iterations.

Output format:

For given ratings_task1.csv file, example execution (**One of the final testcases**)

python uv.py ratings_task1.csv 100 4382 40 10

Write your output to **console (Standard output)**. After each iteration, print RMSE value to the console with 4 floating points – **%.4f**. Below is just the sample output format for first 4 iterations (Not an actual output)



2. [20 points] In this task, you are asked to modify the parallel implementation of ALS (alternating least squares) algorithm in Spark, so that it takes a utility matrix as the input. The code for the algorithm is als.py under the <your spark installation directory>/examples/src/main/python. A copy is also provided with this handout. Make sure that you use the version for **spark-2.1.0 or higher** (it is slightly different from previous versions). Name your file **FirstName_LastName_als.py**. If you are using Python 3, please name it **FirstName_LastName_als3.py**

Execution format:

bin/spark-submit als.py ratings-file n m f k p output-file

All parameters are the same as for uv.py, except for additional parameters **p**, which is the number of partitions for the utility matrix, and **output-file** is a file where you will write your output to. Do not hard code it to any name. You should create the file with name passed as a parameter.

Output format:

For given ratings_task2.csv, example execution (**One of the final testcases**):

bin/spark-submit als.py ratings_task2.csv 671 9066 60 10 2 out_task2.txt

Write your output (RMSE) to **output-file** (In this example, out_task2.txt) with 4 floating points – **%.4f**. Below is the sample output format for first 4 iterations (Not an actual output)

```
1 0.6731
2 0.6017
3 0.6015
4 0.6014
```

Submission on Blackboard:

A zip file **FirstName_LastName_hw3.zip** containing two Python files. Please name your files exactly like below based on Python version.

a. FirstName_LastName_uv.py (For python 3, FirstName_LastName_uv3.py)
b. FirstName_LastName_als.py (For python 3, FirstName_LastName_als3.py)

Grading Criteria:

- Program execution time needs to be under **6 minutes for each task**. You can be ensured that the final testcases given here (for both tasks) are the biggest, and other testcases we will test with would be smaller (Either smaller ratings-file, or smaller f – number of dimensions).
- There will be 30% penalty if your program takes more than **6 minutes**.
- There will be 20% penalty if output format is not followed.
- 10% penalty if naming convention for python scripts is not followed.
- Assignment is due at 11:59 pm on 03/22 (3 weeks). Late submission will have 10% of penalty for every 24 hours that it is late. No credit will be given after 72 hours of its due time.

Note:

- You may use numpy package in this homework. To install it on EC2, execute "sudo pip install numpy".
- **We will be using Moss for plagiarism detection**. Do not share your code with anyone, or copy from others!
- Start early to work on efficiency of your program!