Homework #3: Recommending Movies in Spark using LSH & ALS

Due: November 3, Sunday
120 points

In this homework, we again consider the movie lens data set: https://www.kaggle.com/shubhammehta21/movie-lens-small-latest-dataset/. Recall that we only consider the "ratings.csv" file which has 100,836 rows (ignore the header).But we will also need the rating column in this homework, in addition to the first two columns: userId and movieId.

Our goal is to predict the ratings of movies users have not watched. We will consider two ways of doing this. The first approach is to use LSH to find similar users where each user is represented as a set of movies he/she has watched (regardless the rating). Then rate the movies based on the average rating of top-3 most similar users. Please use the hash function h(x) = (3x + 11*i) % 100 + 1, where x is the movie Id and "i" ranges from 1 to 50. Note that here we regard movie Id as row number (starting from 1). The second approach is to use UV decomposition (the ALS approach) where rows in the utility matrix represent users and columns represent movies.

Note that LSH part is worth 80 points and ALS 40 points.

Requirements & hints:

- Your implementation should be parallelized and written in Spark.
- You may simply adopt the ALS sample code for the 2nd approach above.
- Name your scripts: lsh.py and als.py respectively.
- Hints:
    - While computing signatures, you can divide users into partitions and compute signatures for the partitions in parallel.
    - While computing LSH, you could take the band as key (or part of key), the user ID as value, and then find the candidate pairs/users in parallel.
    - You need to compute the Jaccard similarities of similar pairs identified by LSH, based on which you find top-3 most similar users.

Execution & output format:

spark-submit lsh.py ratings.csv predictions.csv

spark-submit als.py ratings.csv predictions.csv

where predictions.csv stores the predicted ratings in csv format. Ratings are sorted first by user ID, and then movieID (both ascending), and finally by ratings (descending). One line per predicted user's rating on a movie.