Jiayue Shi's Description

1. Version

Spark version: 2.2.1

Scala version: 2.11

Hadoop version: 2.7

2. How to run

First of all, the input files should under the data folder, and output file will also be under data folder.

Task1_1

./bin/spark-submit --class JaccardLSH Jiayue_Shi_hw3.jar data/video_small_num.csv data/Jiayue_Shi_SimilarProducts_Jaccard.txt

Task2_1

./bin/spark-submit --class UserBasedCF Jiayue_Shi_hw3.jar data/video_small_num.csv data/video_small_testing_num.csv data/Jiayue_Shi_UserBasedCF.txt

Task2_2

./bin/spark-submit --class ModelBasedCF Jiayue_Shi_hw3.jar data/video_small_num.csv data/video_small_testing_num.csv data/Jiayue_Shi_ModelBasedCF.txt

Task2_3

./bin/spark-submit --class ItemBasedCF --driver-memory 2g Jiayue_Shi_hw3.jar data/video_small_num.csv data/video_small_testing_num.csv data/Jiayue_Shi_SimilarProducts_Jaccard.txt data/Jiayue_Shi_ItemBasedCF.txt

3. Task1 Result

Task1	Jaccard	Cosine
Precision	1	0.9410651799018869
Recall	0.9235976599949131	0.7374794987863281
Time	44s	25min

4. Task2 Result

Task2	ModelBased	UserBased	ItemBased - USE LSH

Accuracy	>=0 and <1: 4601 >=1 and <2: 2194 >=2 and <3: 568 >=3 and <4: 329 >=4: 8 RMSE: 1. 2954668	>=0 and <1: 4452 >=1 and <2: 2308 >=2 and <3: 650 >=3 and <4: 286 >=4: 4 RMSE: 1. 320513	>=0 and <1: 4117 >=1 and <2: 2310 >=2 and <3: 810 >=3 and <4: 400 >=4: 63 RMSE: 1.4189949
Time	10s	25s	17s

Compare: If we use LSH, we will use less time to get our result (17s < 25), because after LSH's filter, we will use less movies to calculate their pearson similarity. But we will have larger RMSE (1.41 > 1.32), because LSH may filter some valuable item pairs, which may make influence on result.

5. Improvement

In UserBased task, there are lots of ratings which algorithm cannot predict. So, I calculated RMSE of use certain user's average rating or certain movie's average rating, Then I find the previous RMSE is smaller than the later. But when I try the average of both to fill missing value, I find the average of both of them will get lower RMSE. Therefore, I use (0.6 * certain user's average rating + 0.4 * certain movie's average rating) to fill missing value, and this one get best result.