INF 553 HW2 Task3 Explanation

Siqi Liang
3330505568

Nov. 2019

Requirements

Python 3.6, Scala 2.11, Spark 2.3.3

Task 1

	Jaccard similarity
precision	1.0
recall	0.99
Time(sec)	12.40s

Speed-up tips:

1. Local test shows `numPartitions=5` helps to speed up (`numPartitions>=3` works better than `numPartitions=2`);
2. The calculation of signature matrix can be expressed as below: Assuming there are m hash functions, the signature column for any business b_i is $\mathrm{Sig}(b_i)$:

$$\mathrm{Sig}(b_i) = [\min_{u_j \in U(b_i)} \{h_1(u_j)\}, \min_{u_j \in U(b_i)} \{h_2(u_j)\}, \cdots, \min_{u_j \in U(b_i)} \{h_m(u_j)\}]^T$$

   where $h_k(u_j)$ is the hash value of user u_j on the kth hash function, $U(b_i)$ is the set of users who have rated the business b_i. This expression helps to speed up a lot comparing with looping the whole hash matrix and user-business characteristic matrix.
3. Pass the characteristic matrix in the form of one column for each business id `bid` as python `dict()`, that is, {`bid1:[uid1_1,uid1_2,...],bid2:[uid2_1,uid2_2,...],...`}, which is faster to collect all users who have rated the business b_i.

Task 2

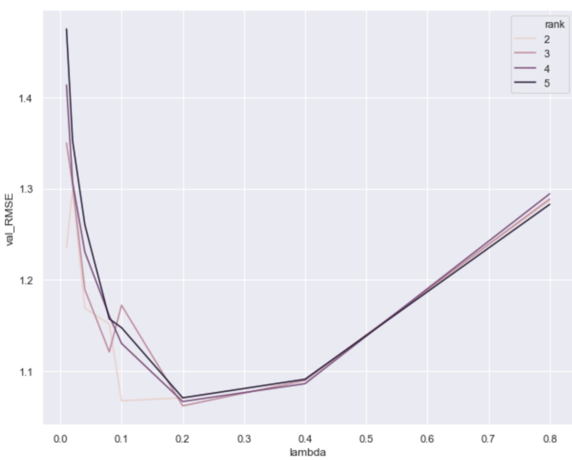
Baseline for model-based and user-based CF training on `yelp_train.csv` and testing on `yelp_val.csv`:

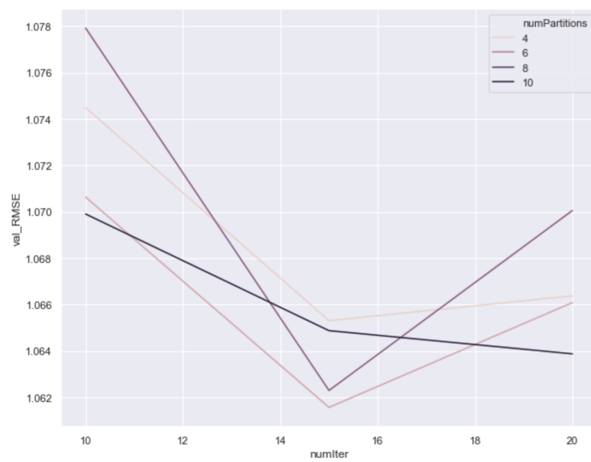
	Model-based	User-based
RMSE	1.066	1.16981
Time(sec)	17.60s	13.06s

Model-based CF

Tips:

1. Use `yelp_train.csv` and `yelp_val.csv` to tune the model hyperparameters, and the result shows that optimal hyperparameters could be `rank=3`, `lambda=0.2`, `iterations=15` for matrix factorization algorithm training process. The RMSE changes on `yelp_val.csv` on different hyperparameters:





User-based CF

Tips:

1. Only use the top-K most similar neighbors for the prediction. Most users have very limited corated-users on the business to predict. And the local test shows when only using top-20 most similar neighbors for the prediction could achieve lower RMSE on `yelp_val.csv`.

Figure 1: Frequency for different corated-user numbers