DSGA 1004 Big Data - Final project checkpoints

Name: Yue Feng, Luoyao Chen, Shaojun Jiang

NetID: yf1451, lc4866, sj2539

Link to GitHub

https://github.com/nyu-big-data/final-project-group_3

Data Preprocessing

1. Remove duplicate records in the datasets, keep only the first record.
2. Drop ratings with null values.
3. Drop ratings that are too low/high - we only kept ratings in range [0.5,5]
4. Keep movies with more than 10 rating records.
5. Keep users with more than 10 rating records.

Train/Validation/Test Split

We first split our data into train/validation/test sets as dataframes based on userId with weights 0.6, 0.2, 0.2 respectively. Then, to guarantee that we have all the user history in the training set - to avoid the cold start problem - we sort each user's rating records in the validation and test set by timestamps and put the earlier half of the rating records from each user into the training set, while leaving only the later half of the ratings in the validation and test sets. Finally, we repartition each of the train/validation/test data frames and create parquets for further operations.

Choice of evaluation criteria

Using MAP to evaluate a recommender algorithm implies that we are using the users' score of each movie to create a ranking list. Since a user has a finite amount of time and attention, not only do we want to know what kinds of products they might like, but also which are liked the most or which we are most confident of for recommendation. For the baseline popularity model, we sorted the movies recommended by confidence, and compared with the ground truth (movies watched by each user) using MAP.

For the ALS model: We fit the model and tuned using two criterions: MSE and MAP. However, MSE does not give evaluation on how accurate the model performance was, so we used MAP to select the optimal parameters.

Baseline Implementation

As our baseline, we used a popularity-based model. For each movie appearing in the training set, we calculated an average rating. Then, we sorted the movies by the averaging rating and chose the top 100 to recommend to users.

Baseline Results

	Small Dataset	Large Dataset
val MAP	0.000703125	3.53E-06
test MAP	0.0003571428571	3.55E-06

Documentation of latent factor model's hyper-parameters

The parameters we tuned included rank (the number of latent factors) and regularization factor (penalty coefficient). Since the training set (small) has 610 distinct users and 9058 distinct movies, we determined that the rank we tuned were 50, 100, 300, 500, and regularization were 0.01 and 0.1.

Evaluation of latent factor model on small datasets:

rank	reg	MAP
50	0.1	0.02901785714
50	0.01	**0.05125925926**
100	0.1	0.03066666667
100	0.01	**0.05244444444**
300	0.1	0.03266666667
300	0.01	0.05555555556
500	0.1	0.029375
500	0.01	**0.05857142857**

As a preliminary conclusion, using the same regularization, the map increases as rank increases (500); Using the same rank, as regularization becomes harsher (0.1), the MAP is lower. This makes sense, since when the model has a smaller number of latent factors, or the penalization is harder (model is more general), the accuracy is not as good as a more specific model.