Introduction to Recommender Systems (Spring 2023)

Homework #1 (100 Pts, March 29)

Student ID 2022712151

Name 정지원

죄송합니다. 보고서는 4월 1일까지 제출인줄 알았습니다.

Instruction: You should submit your code and report to i-campus. Follow the submission format below.

- **RS_HW1_STUDUENT_ID_NAME.zip: compress 1) your code and 2) this document in pdf format.**

NOTE 1: You should write your codes **only in 'EDIT HERE.'**

NOTE 2: You need to install Python, NumPy, Scikit-learn (sklearn), Pandas, tqdm, and Matplotlib libraries.

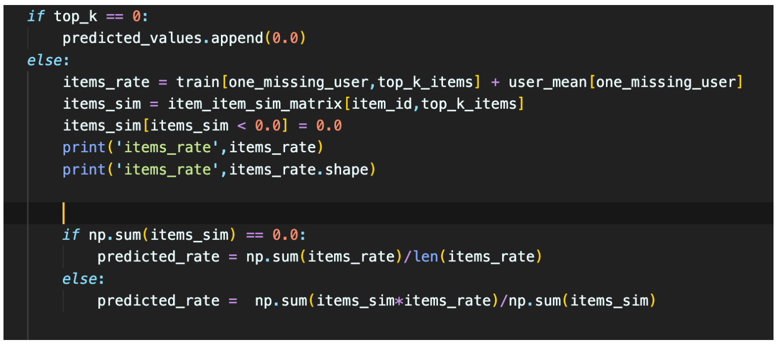
(1) [40 pts] We provide all template codes and datasets. Refer to 'models/userKNN_explicit.py,' and write your code to implement the item-based collaborative filtering algorithm on 'models/itemKNN_explicit.py.' Run 'CF_main.py' to run the source code.

(a) [20 pts] Implement the training and evaluation codes (Fill in your own codes in the EDIT HERE parts) of the function "ItemKNN" in 'models/itemKNN_explicit.py' using the Adjusted cosine similarity. The similarity is defined as follows:

$$sim(i,j) = \frac{\sum_{u\in\mathcal{S}_{ij}}(r_{ui}-\bar{r}_{u*})(r_{uj}-\bar{r}_{u*})}{\sqrt{\sum_{u\in\mathcal{S}_{ij}}(r_{ui}-\bar{r}_{u*})^2}\sqrt{\sum_{u\in\mathcal{S}_{ij}}(r_{uj}-\bar{r}_{u*})^2}}$$

Note: Fill in your code here. You also have to submit your code to i-campus.

Answer:



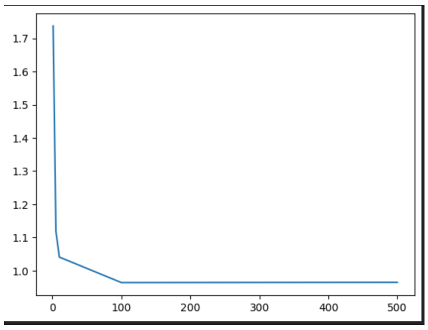
```
if top_k == 0:
    predicted_values.append(0.0)
else:
    items_rate = train[one_missing_user,top_k_items] + user_mean[one_missing_user]
    items_sim = item_item_sim_matrix[item_id,top_k_items]
    items_sim[items_sim < 0.0] = 0.0
    print('items_rate',items_rate)
    print('items_rate',items_rate.shape)


    if np.sum(items_sim) == 0.0:
        predicted_rate = np.sum(items_rate)/len(items_rate)
    else:
        predicted_rate =  np.sum(items_sim*items_rate)/np.sum(items_sim)
```

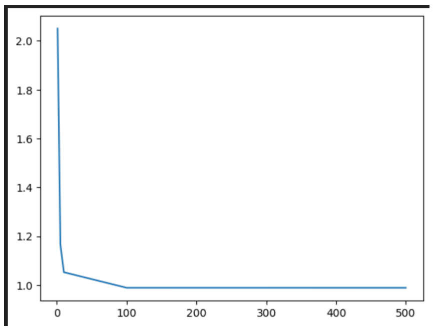
(c) [20 pts] Given the data ('movielens_100k.csv'), draw the plots of RMSE by adjusting K (i.e., the number of nearest neighbors) for UserKNN and ItemKNN, respectively. (You can change the top_k value of UserKNN and ItemKNN, respectively.) (i) For varying K, explain the results and how much K affects RMSE. (ii) evaluate which of the two algorithms, user-based and item-based collaborative filtering, is better?

- Search space of K = {1, 5, 10, 100, 500}

Note: Show your plots and explanations in short (3-5) lines.

Answer:





User_KNN **Item_KNN**

In the movielens_100k.csv dataset, user_knn has a lower rmse score than item_knn, so it shows better performance. The best performance is when user_knn's top_k is 100, and item_knn's best performance is when top_k is 500.

UserKNN,K=1: 1.737231, UserKNN,K=5: 1.119028, UserKNN,K=10: 1.040761, UserKNN,K=100: 0.963961, UserKNN,K=500: 0.964693

ItemKNN,K=1: 2.050220 ,ItemKNN,K=5: 1.165208, ItemKNN,K=10: 1.052499, ItemKNN,K=100: 0.988824, ItemKNN,K=500: 0.988756

(2) [60 pts] (Kaggle competition.) The task is **rating prediction.** Please use the given data to predict movie ratings for each user. Please refer to '**http://www.kaggle.com/competitions/skku-rs2023-assignment-1**'. It is a private competition, so you should visit via the above link. "code/baseline.ipynb" is a baseline UserKNN code for the competition. It includes codes for data preprocessing, model training, evaluation, and making submission file.

[Scoring policy]

The final evaluation will be made by adding the private leaderboard score and the idea score. Please write a report on your project solution with a maximum of 2 pages.

[Competition Rules]

- Do not cheat.
- Use Python.
- No limitation on python libraries. (Pytorch, Tensorflow, etc.)

- You must use "{Student ID}_{Name}" for your team name in the Kaggle competition.
- No late submission in the Kaggle competition.
- Any use of external data is prohibited.
- Your submission to Kaggle is limited to five times a day.

Please write down your solution of Kaggle competition here:

I wrote the code using the surprise library. A knn-based neighborhood based collaborative filtering model was used, and model based collaborative filtering models such as SVD, SVDPP, and NMF were used.

Grid search was applied using 5-validation based on the hyperparameters included in each model. As a result of the experiment, the performance of the KNNwithMeans model was the best, and the result was submitted to kaggle.

Evaluation metrics were based on RMSE, MAE, MSE, and FCP.

More details are provided in the files surprise_utils.py and kaggle.ipynb