Programming Assignment 2

Personalized Item Recommendation

Web Retrieval and Mining

Spring 2025

Updates

Outline

- Programming Assignment
  - Method
  - File format
  - Evaluation
- Report
- Kaggle Competition
- Scoring
- Deadlines

Introduction

In this homework, you are asked to implement two small Learning-to-rank recommendation models.

We will give you only the historical interacted item list of each users, and your task is to find and recommend top 50 most relevant items (**except the training positive items**) for them individually.

You should implement **two designated methods** based on the **latent factor model** (Matrix Factorization). Details will be included in the following pages.

Method 1: Binary Cross Entropy Loss

You should regard this as binary classification task.

All data in train.csv represents the positive pairs (*user, item_pos*), so you should sample the negative pairs (*user, item_neg*) for each user from the entire item set by yourself.

You can adjust the ratio of positive pairs and negative pairs in your own way.

$$BCELoss = -[\sum_{(u,i)\in\mathcal{D}^+} \log\sigma(u^T i) + \sum_{(u,j)\in\mathcal{D}^-} \log(1-\sigma(u^T j)]$$

Method 2: Bayesian Personalized Ranking Loss

You should regard this as ranking task.

All data in train.csv represents the positive pairs (*user, item_pos*), so you should sample the negative pairs (*user, item_neg*) for each user from the entire item set by yourself. You can adjust the ratio of positive pairs and negative pairs in your own way.

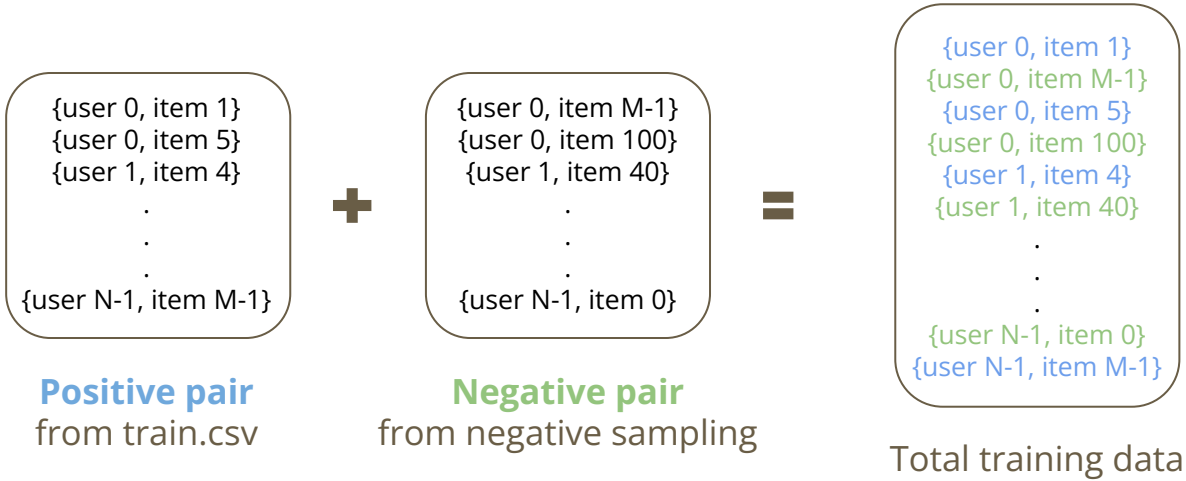
You have to use Bayesian personalized ranking loss for your hidden factor model (MF).

$$BPRLoss = \sum_{(u,i,j) \in \mathcal{D}} \ln \sigma(u^T i - u^T j) + \lambda \|\Theta\|^2$$

Paper link: https://arxiv.org/ftp/arxiv/papers/1205/1205.2618.pdf

Negative Sampling

N users, M items.



Positive pair
from train.csv

Negative pair
from negative sampling

Total training data

Input File Format

This is the only training data provided for this assignment.

Each line contains the chronologically ordered interaction history of a user, formatted as follows:

[*user_id*], [item ids of *user_id*]

Interacted item IDs for user 0, separated by spaces.

```
1  UserId,ItemId
2  0,1369 3099 775 601 2336 2114 282 2471 1750 1466 372 283 2493 189 1348 1831 1487 2300 3010 506 1288 2186 2430 3031 3 935 781 1818 2143
   182 86 1738 442 1180 862 136 217 106 762 2161 16 1020 67 257 1131 1739 1581 92 370 491 2152 1545 70
3  1,30 329 1375 357 1247 82 871 1995 50 647 2292 660 1 1519 1707 191 1555 260 1965 1077 2318 949 127 84 2845 819 2440 21 2421 351 254 387
   93 1600 2661 217 1095 530 1421 4 839 701 3164 604 47 3036 142 793 2887 3179 1766 2 321 2664 103 664 248 31 1900 625 270 94 421 1909 1376
   238 1046 2751 867 1350 1244 1813 702 1213 1085 694 137 2267 2463 2196 1879 66 1191 308 273 1093 509
```

Output Format

The first line includes two column names: “**UserId**”, “**ItemId**”.

“**UserId**”: the set of *user_id* in train.csv

“**ItemId**”: Ranking list of top 50 items of each user.
Should be separated by spaces.
Retrieved items should be sorted by their relevant score.
You can retrieve at most 50 items for each user.

```
UserId,ItemId
0,0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
1,0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
2,0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
3,0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
```

sample_submission_50.csv

Program Execution

You should write your own shell script (*run.sh*) to compile and run your program.

When testing your program, we will execute the following commands on R217 workstation, please make sure your program commands is executable on the workstation.

$./run.sh [*path of your output ranking list*]

Note: When testing your program, we will use the same testing file as it on Kaggle. i.e. You should get the same result as it on Kaggle.

We will only use CPU to test your program, so please check your program could output the ranking list within 5 minutes using CPU.

Evaluation

We will use the **Mean Average Precision (MAP@50)** value to evaluate your ranking list.

We will not provide testing data so you have to split train.csv into training and validation data.

Restrictions

- You can use any language, but using third party tools directly for MF or BPR is prohibited.
- Your program should finish in 5 minutes using only CPU.
- If you are not sure packages you used is legal or not, please inquiry TA by e–mail.
- Do not copy other's code.
- Do not copy code generated by Large Language Models
- If your model is too big to compress, please upload it to an accessible link and attach this link to README.md.

Submission

Please put report, scripts and code into the directory named your student ID. Package this folder into a zip file and submit it to NTU COOL, following is the structure and content of the **zip**:

For example: R12922XXX.zip ((1), (2)... generated by NTU COOL is allowed)

+---R12922XXX(directory) (with R in uppercase)

+---**run.sh**

+---**report.pdf**

+---**Your model** (Any types of extensions as long as your program can load!)

+---requirements.txt (Specify all packages you need.)

+---README.md (not necessary, if your model is too big to compress, upload it to an accessible link and attach this link to README.md)

+---All the other files and source code required by your program

Do not submit the training file and sample submission file.

Report

Please write your report as a **report.pdf** and put it into the zipped file. The report should contain the following content:

Q1 : Describe your **MF with BCE** (e.g. parameters, loss function, negative sample method and MAP score on Kaggle public scoreboard)

Q2 : Describe your **MF with BPR** (e.g. parameters, loss function, negative sample method and MAP score on Kaggle public scoreboard)

Q3 : Compare your results of Q1 and Q2. Do you think the BPR loss benefits the performance? If do, write some reasons of why BPR works well; If not, write some reasons of why BPR fails.

Q4 : Plot the MAP curve on testing data(Kaggle) for hidden factors d = 16, 32, 64, 128 and describe your finding.

(Bonus 10%) Q5 : Change the ratio between positive and negative pairs, compare the results and discuss your finding. Discuss how will negative sampling affect the training?

Kaggle Competition

Join competition:

https://www.kaggle.com/t/ab7b4474289d4e1aa091e1d75cda6dfb

- You must change your team name to your student ID (**upper case**).
- This is an individual assignment. Do NOT cheat.
- You can submit a maximum of 5 entries per day.

Leaderboard

- Public/Private leaderboard
- 50%/50% users for public and private respectively
- Best on public ≠ best on private
- Private leaderboard will be released after deadline

Scoring

- 30% for your report
- 15% for BCE loss implementation
- 15% for BPR loss implementation
- 10% for performance better than strong baseline on public leaderboard
- 10% for performance better than strong baseline on private leaderboard
- 10% for performance better than simple baseline on public leaderboard
- 10% for performance better than simple baseline on private leaderboard

Scoring

- Note that you will **receive 0 points** in the following situations:
  - You don’t have a record on the ranking website (for performance score)
  - We can’t run your code, or the run time limit exceeded (5 min)
  - You plagiarize someone else’s code
  - Your code or report is directly generated by LLMs
- -3 points for the wrong file format or missing files

Deadlines

- Kaggle submission Deadline: 2025/05/16 23:59 (UTC+8)
- Report submission Deadline (via NTU COOL): 2025/05/16 23:59 (UTC+8)
- Late policy: 10% deduction per day
- For questions, contact the TAs at: ir2025.ntu@gmail.com