CSE 158/258, MGTA 461, DSC 256, Fall 2024: Homework 3

Instructions

Please submit your solution **by Monday, Nov 11.** Submissions should be made on **gradescope**. Please complete homework **individually**.

These homework exercises are intended to help you get started on potential solutions to Assignment 1. We'll work directly with the Assignment 1 dataset to complete them, which is available from: `http://cseweb.ucsd.edu/classes/fa24/cse258-b/files/assignment1.tar.gz`

You'll probably want to implement your solution by **modifying the baseline code provided** in the assignment directory.

You should submit two files:

`answers_hw3.txt` should contain a python dictionary containing your answers to each question. Its format should be like the following:

```
{ "Q1": 1.5, "Q2": [3,5,17,8], "Q2": "b", (etc.) }
```

The provided code stub demonstrates how to prepare your answers and includes an answer template for each question.

`homework3.py` A python file containing working code for your solutions. The autograder *will not execute your code*; this file is required so that we can assign partial grades in the event of incorrect solutions, check for plagiarism, etc. Your solution should **clearly document which sections correspond to each question and answer**. We may occasionally run code to confirm that your outputs match submitted answers, so **please ensure that your code generates the submitted answers.**

You may build your solution on top of the provided stub:

Homework 3 stub : `https://cseweb.ucsd.edu/classes/fa24/cse258-b/stubs/`

Each question is worth 1 mark.

Tasks (Read prediction)

Since we don't have access to the test labels, we'll need to simulate validation/test sets of our own.

So, let's split the training data ('train_Interactions.csv.gz') as follows:
(1) Reviews 1-190,000 for training
(2) Reviews 190,001-200,000 for validation
(3) Upload to Gradescope for testing only when you have a good model on the validation set. If you can build such a validation set correctly, it will significantly speed up your testing and development time.

1. Although we have built a validation set, it only consists of positive samples. For this task we also need examples of user/item pairs that *weren't* read. For each (user,book) entry in the validation set, sample a negative entry by randomly choosing a book that user *hasn't* read.[1] Evaluate the performance (accuracy) of the baseline model on the validation set you have built.

2. The existing 'read prediction' baseline just returns *True* if the item in question is 'popular,' using a threshold based on those books which account for 50% of all interactions (`totalRead/2`). Assuming that the 'non-read' test examples are a random sample of user-book pairs, this threshold may not be the best one. See if you can find a better threshold (or otherwise modify the thresholding strategy); report the new threshold and its performance of your improved model on your validation set.

3. A stronger baseline[2] than the one provided might make use of the Jaccard similarity (or another similarity metric). Given a pair (u, b) in the validation set, consider all training items b' that user u has read. For each, compute the Jaccard similarity between b and b', i.e., users (in the training set) who have read b and users who have read b'. Predict as 'read' if the *maximum* of these Jaccard similarities exceeds a threshold (you may choose the threshold that works best). Report the performance on your validation set.

[1] This is how I constructed the test set; a good solution should mimic this procedure as closely as possible so that your leaderboard performance is close to their validation performance.

[2] This baseline is not *always* stronger, depending on the dataset.

4. Improve the above predictor by incorporating both a Jaccard-based threshold *and* a popularity based threshold. Report the performance on your validation set.[3]

5. To run our model on the *test* set, we'll have to use the files 'pairs_Read.csv' to find the userID/bookID pairs about which we have to make predictions. Using that data, run the above model and upload your solution to the (Assignment 1) Gradescope. If you've already uploaded a better solution, that's fine too! Your answer should be the string *"I confirm that I have uploaded an assignment submission to gradescope"*.

Tasks (Rating prediction)

Let's start by building our training/validation sets much as we did for the first task. This time building a validation set is more straightforward: you can simply use part of the data for validation, and do not need to randomly sample non-read users/books.

6. Fit a predictor of the form

$$\text{rating}(\text{user}, \text{item}) \simeq \alpha + \beta_{\text{user}} + \beta_{\text{item}},$$

by fitting the mean and the two bias terms as described in the lecture notes. Use a regularization parameter of $\lambda = 1$. Report the MSE on the validation set.

For this question note carefully that the objective optimized should be the ***sum of squared errors*** **plus the regularizer** (squared $\ell 2$ norm scaled by λ), i.e., it should not be the *mean squared error*. Although using the MSE vs SSE is equivalent (up to a change in λ) it is important that your objective follows this exact specification so that everyone's solution is the same.

7. Report the user IDs that have the largest and smallest (i.e., largest negative) values of β_u, along with the beta values.

8. Find a better value of λ using your validation set. Report the value you chose, its MSE, and upload your solution to gradescope by running it on the test data.

[3] You are welcome to combine them in any way you like; e.g. you could treat the two values as features in a classifier.