11-442/11-642: Search Engines	Spring 2025
Homework 3	
Released: February 22, 2025	Due: March 17, 2025

The goal of this homework assignment is to get you familiar with core problems in recommender systems. It will cover,

- similarity-based recommendation
- matrix factorization-based recommendation
- preference-based recommendation
- learning to rank for recommendation

1 Material

1.1 Data

Download the dataset for the assignment from https://boston.lti.cs.cmu.edu/classes/11-642/HW/HW3/movierec.tgz. The dataset consists of four files recording the implicit feedback signals from users.

Consumption data (consumption.jsonl). Feedback for a user-item pair. Each line in this jsonl represents all of information we have about user consumption. This includes a user id (`uid`), a list of implicit feedback information (`implicit`), and a list of explicit feedback information (`explicit`).

Each element of the implicit feedback list includes the movie id (`mid`), an integer timestamp representing the seconds since January 1, 1970 (`time`), and a float between 0 and 10 representing the implicit feedback score (`value`). All time zones are aligned. The implicit feedback signal is the output of a linear model of how much a user liked a movie that they watched based on the user's view time, clicks, and shares of the movie (after they watched the movie). In other words, the implicit feedback signal for user u after consuming movie i is

$$\tilde{y}_{u,i} = b + w_{u,i}^{\mathsf{T}} x_{u,i}$$

where $x_{u,i}$ is a vector of observed user-item signals from log data, $w_{u,i}$ is a vector of weights, and b is a bias term. The model predictions are comparable within an individual user's data but may not be comparable across users. You do not need to know x or w for this assignment. The engineering team has let you know that there was a bug in the implicit

feedback model during all of February 2024, August 2024, and October 2024 that added 2 to the bias term of the linear model.

Each element of the explicit feedback list includes the movie id (`mid`) and a float between 0 and 5 representing the explicit rating provided by the user for that movie (`value`). We can consider ratings as ground truth and comparable across users.

Movie information (movies.jsonl). Each line of this jsonl represents all of the information we have about a movie. This includes a movie id (`mid`), its name (`name`), a list of zero or more genres (`genre`), a list of zero of more user-provided tags (`tags`), and the release year (`year`).

Test movies (test.jsonl). Each line of this jsonl represents the movies to rank for 500 users in the test set.

Explicit feedback (explicit.qrel). A separate file of explicit data from `consumption.jsonl` in `trec_eval` format. This is ground truth data you can use to evaluate system rankings.

1.2 Code

For linear algebra, such as matrix factorization, you should use `scipy.sparse.linalg`.

For Bayes Personalized Ranking (BPR), you should use the `cornac` package.

As with previous assignments, we will be using `trec_eval` to evaluate performance. As such, all results should be generated in five-column `trec_eval` format,

```
<uid> Q0 <mid> <rank> <score> <model id>
```

where scores should rank movies in decreasing order.

2 Experiments

2.1 Recommendation based on user similarity

Implement user similarity based on cosine similarity and Pearson correlation using the implicit consumption data.

Next implement a k-nearest neighbor algorithm to predict the rating of a new item. If $\kappa(u, v)$ is the similarity between users u and v using either cosine or Pearson, then the k-nearest

neighbor predictor will output,

$$\hat{y}_{u,i} = \frac{1}{\mathcal{Z}} \sum_{v \in \mathcal{N}_u^k} (\tilde{y}_{v,i} - \overline{y}_v) \times \kappa(u, v)$$

where $\mathcal{N}_u^k$ are the k most similar users to u given similarity κ, $\overline{y}_u$ is the mean (non-zero) implicit feedback score for user u, and,

$$\mathcal{Z} = \sum_{v \in \mathcal{N}_u^k} \kappa(u, v)$$

If a movie is not present in the implicit data, you should score it as -10000 and rank it last. You use these scores to rank movies and evaluate the rankings using `trec_eval` and the explicit data `explicit.qrels`.

You should submit the following files, each set of per-user rankings of movies in `test.jsonl`. You should only rank those movies for each user. Please generate rankings for $k = \{1, 5, 10\}$.

- `knn-cosine-k.run`
- `knn-pearson-k.run`

where k is replaced by the value in the run.

2.2 Recommendation based on matrix factorization

In this part, we will use singular value decomposition (SVD) to predict implicit scores. Let $\mathbf{A}$ be the $m \times n$ user-item matrix of implicit values from `consumption.jsonl`. In other words, $A_{u,i} = \tilde{y}_{u,i}$. As discussed in class, we can use SVD to decompose A into three matrices,

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\mathsf{T}$$

We can take the k largest singular values as,

$\mathbf{U}_t$	$m \times k$ submatrix of $\mathbf{U}$ associated with the largest k singular values.
$\mathbf{\Sigma}_t$	$k \times k$ submatrix of $\mathbf{\Sigma}$ associated with the largest k singular values.
$\mathbf{V}_t$	$n \times k$ submatrix of $\mathbf{V}$ associated with the largest k singular values.

and reconstruct $\mathbf{A}$ as,

$$\tilde{\mathbf{A}} = \mathbf{U}_t\Sigma_t\mathbf{V}_t^\mathsf{T}$$

so that $\tilde{A}_{u,i}$ is the predicted ratings of movie i for user u. If a movie is not present in the implicit data, you should rank it last.

We can also compute the SVD using mean-shifted values,

$$\tilde{y}_{u,i} - \overline{y}_u$$

where $\overline{y}_u$ is defined in Section 2.1. Note that we only shift the observed values in the implicit data, not the unobserved values. We can compute SVD on this adjusted matrix as with the original implicit data. If a movie is not present in the implicit data, you should rank it last.

You should submit the following files, each set of per-user rankings of movies in `test.jsonl`. You should only rank those movies for each user. Please generate rankings for $k = \{25, 50, 100\}$.

- `svd-k.run`
- `svd-k-mean-shifted.run`

2.3 Recommendation based on pairwise preferences

We will consider Bayes Personalized Ranking to learn from pairwise preferences.

You can read your data into a `cornac.data.Dataset` by loading a list of $\langle u, i, \tilde{y}_{u,i} \rangle$ tuples through `cornac.data.Dataset.build`. As a convenience, `cornac.data.Dataset` includes two maps `uid_map` which converts user id strings into row indexes and `iid_map` which converts item id strings into column indexes.

To train a model, you should use `cornac.models.BPR`, initialized as,

```
bpr = BPR(k=50, lambda_reg=0.001)
```

where `k` indicates the embedding dimension. You can then train the model using `bpr.fit(data)` if `data` is a `cornac.data.Dataset`. Please use `lambda_reg=0.001` for all experiments. If you would like to observe progress during training, you can include `verbose=True`.

In order to score items for a user, you can call `bpr.score` with a user and item index. If a movie is not present in the implicit data, you should rank it last.

You should submit the following files, each set of per-user rankings of movies in `test.jsonl`. You should only rank those movies for each user. Please generate rankings for $k = \{25, 50, 100\}$.

- `bpr-k.run`
- `bpr-k-mean-shifted.run`

2.4 Learning to rank

Finally, we will combine the movie scores based on cosine similarity, Pearson correlation, SVD, and BPR using a learning to rank framework of your choice from Homework 2. You can generate training data from the explicit feedback information in `consumption.jsonl`.

Consider the following feature sets,

- $f_{1:6}$: one feature for each of the six rankers in the previous questions,
  - f_1: nearest neighbor with cosine similarity, $k = 10$.
  - f_2: nearest neighbor with Pearson correlation, $k = 10$.
  - f_3: SVD, $k = 50$.
  - f_4: mean-shifted SVD, $k = 50$.
  - f_5: BPR, $k = 50$.
  - f_6: mean-shifted BPR, $k = 50$.
- $f_{7:8}$: two custom features you are free to define using information in `movies.jsonl`. You should design features that are likely to be correlated with utility to the user.

Train a model using only $f_{1:6}$ and a second using all features $f_{1:8}$.

You should submit the following files, each set of per-user rankings of movies in `test.jsonl`. You should only rank those movies for each user.

- `ltr-base.run`
- `ltr-base+custom.run`

3 Questions

In addition to these runs, please provide answers to the following questions.

3.1 11-442 students

1. Explain any difference in performance between k-nearest neighbor with cosine similarity and Pearson correlation.
2. Explain any difference in performance between when mean-shifting when using SVD and BPR.
3. Describe your custom features and why they were reasonable proxies for utility.

3.2 11-642 students

1. Explain any difference in performance between k-nearest neighbor with cosine similarity and Pearson correlation.
2. Explain any difference in performance between when mean-shifting when using SVD and BPR.

3. Custom features.

   (a) Describe your custom features and why they were reasonable proxies for utility.

   (b) Describe the types of users or movies where they helped (or didn't). You can look at things like the amount of data associated with the user, the genre's the user is interested in, and so forth.

4. Using information in `movies.jsonl`, plot the relationship between rank position and (i) release years and (ii) genres. The horizontal axis of each scatterplot should be the year or genre and the vertical axis should be the rank position. Provide a brief explanation for any trends you observe.

4 Report

For 11-442 students, you can find a template for your report here.

For 11-642 students, you can find a template for your report here.

5 Submission

Forthcoming.