Programming Project
Collaborative Filtering

Algorithms for Data Science

October 10th, 2024

The objective of this programming assignment is to be able to implement a system for recommending items (e.g., movies), using collaborative filtering methods.

The programming assignment is *individual*, and, please do not use any LLM such as ChatGPT.

1 Preliminaries

The input will be a dataset that contains user-movie ratings:

```
user_id,movie_id,rating,timestamp
```

The dataset is available at `https://phparis.net/uploads/m2_ds_algods_ratings.zip`. In the following, we assume that user and movie IDs are 1-indexed, i.e., the second line represents the first rating.

Your task is to implement a command-line program, *in the programming language of your choice*, which takes the following two parameters:

1. the name of the file containing the user-movie ratings
2. the similarity threshold (optional, to filter out weak recommendations)

The command line syntax is thus the following:

```
./<program_name> <ratings_file> <similarity_threshold>
```

The console output will be the pairs of user IDs and recommended movie IDs, followed by their predicted rating:

```
./<program_name> ratings.csv 0.05
```

will return the following output:

```
0 12 4.5
1 34 3.9
...
```

2 Assignments

2.1 Algorithm Implementation

Implement the full workflow for collaborative filtering:

1. Decide on whether you will use *user-user* or *item-item* collaborative filtering (justify your choice).
2. Compute the similarity between users (for user-user filtering) or between movies (for item-item filtering) using cosine similarity or any other suitable metric.
3. Use the computed similarities to predict missing ratings for unrated movies.
4. Output recommendations for each user by selecting the movies with the highest predicted ratings.

An important note for implementation: *all the algorithms should be implemented by you*, including similarity computation and rating prediction.

2.2 Analysis & Experimental Evaluation

For a full submission to the project, you must write a document containing an implementation analysis and an experimental section.

The *implementation* analysis section will contain a write-up of the implementation choices. For example, it should contain an analysis of the data structures used, but also a discussion of the optimizations used in the implementation.

The *experimental evaluation* analysis will contain an empirical evaluation of the algorithm in the form of plots evaluating the execution time, memory used, for a selection of parameters: similarity threshold, number of users, number of items, etc. Another important parameter to evaluate is the *precision of recommendations* – how well the predicted ratings match the true ratings for each user. You might also want to explore how different similarity metrics affect performance.

The above are just suggestions: other evaluations are welcome and appreciated and will count in the final grade – if they are relevant to the project focus.

3 Submission & Evaluation

Submit your solutions by **Thursday, 31st October 2024, 11:59pm** for full credit. Submissions sent by Sunday, November 3rd, 11:59pm will incur 5 points of penalty out of 20. Submissions received after this date will receive no credit.

Your submissions should contain two files, and go into two places:

1. an archive (.zip) consisting of the source code and instructions on how to compile (if necessary) and execute the code
2. a PDF document containing the report

Your submission will be evaluated based on private tests run on your algorithms for correctness, on the evaluation of the code and implementation quality, and the quality of your evaluation document. The implementation and documentation will have equal weight – 50% – in the final grade.