RATING PREDICTION

CMP2003 Data Structures and Algorithms (C++) Project
Due Date: Jan 06, 2023

1 The Problem

Recommender systems help people find items of interest by making personalized recommendations according to their preferences. For example, a recommender system can make personalized recommendations of items such as movies, books, hotels, or music to people. In order to model users' preferences their past interactions such as product views, ratings, and purchases are used. In the recommender systems area of research (which is a subfield of machine learning and information retrieval) different algorithms have been developed which are currently being used by many large companies.

In this project you will implement neighborhood based collaborative filtering (NBCF) algorithms in order to make predictions for movie ratings of people. There are two main types of NBCF algorithms: user-based (UBCF) and item-based (IBCF). Let us describe each with an example dataset.

2 UBCF and IBCF

One of the fundamental problems in recommender systems is to predict the rating of a user for a particular item. For example, given the dataset in Table 1, what might be the rating of User 2 for Movie 3? If we can predict this rating then, if it is a high value like 4 or 5, we can decide to recommend Movie 3 to User 2.

Table 1: An example dataset.

	Movie 1	Movie 2	Movie 3	Movie 4
User 1	1	2	1	1
User 2	4	3		2
User 3	2	5	4	4
User 4	4	4	3	2
User 5	1	1	5	5

In its simplest form, UBCF works as follows: in order to predict the rating of user u to item i, we first find the most similar k users to u (who rated i) and predict the rating as the average ratings of these most similar k users on item i. For finding the similarity between two users different methods can be used. In collaborative filtering we look at the ratings of other users and find users which have similar ratings. One popular method to find such a similarity is called cosine similarity which is given below:

$$cosine(A, B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

For example, given two vectors $A = [2, 4, 5]$ and $B = [3, 2, 4]$, their cosine similarity is given by:

$$cosine(A, B) = \frac{2 \times 3 + 4 \times 2 + 5 \times 4}{\sqrt{2^2 + 4^2 + 5^2} \times \sqrt{3^2 + 2^2 + 4^2}} \tag{2}$$

IBCF, on the other hand, works as follows: in order to predict the rating of user u to item i, we first find the most similar k items to i (which are rated by user u) and predict the rating as the average rating of user u for these most similar k items. For similarity calculation, again cosine similarity can be used.

3 Evaluation

The dataset that will be given to you will be a text file and will have the following format (each line contains a rating of a particular user to a particular movie):

UserID, MovieID, Rating
10, 100, 5
10, 101, 4
10, 102, 5
11, 100, 5
11, 201, 4
...

For testing your performance you will be given a test set similar to the following:

UserID, MovieID
10, 200
10, 201
10, 202
11, 300
11, 301
...

As can be seen, the ratings are hidden, your task is to predict these ratings and make a submission to Kaggle web site (more details about this submission will be given later). Your aim is to make the best predictions, that is, predict the actual ratings as close as possible. The error metric for calculating your scores will be root mean squared error (RMSE), which is given below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{3}$$

where y_i is the actual rating and $\hat{y}_i$ is the predicted rating. For example if $y = [2, 4, 5, 3]$ and $\hat{y} = [3, 4, 2, 2]$ then

$$RMSE = \sqrt{\frac{(2-3)^2 + (4-4)^2 + (5-2)^2 + (3-2)^2}{4}} \tag{4}$$

4 Model Building and Experiments

The above description of UBCF and IBCF is a simple and general one. You are free to use different variations of these methods, especially you can try different similarity metrics (other than cosine). You can find more information on NBCF on many web resources. You are free to use any method you like (other than UBCF and IBCF) but you should implement at least UBCF or IBCF and show that it is working.

Since the ratings file that will be provided will be a large file, try to find the best (efficient) data structures and algorithms for applying UBCF and IBCF.

Before submitting your predictions, you might want to do offline experiments. In its simplest form you can split your data into training and test sets. The test set might be constructed by randomly selecting a subset of the rows (1000 rows for example) of the dataset provided and putting all the rest of the ratings

to the training set. Then you can try to predict the ratings in the test set using the training set and calculate the MAE without making a submission. Note that you can make at most 5 submissions to Kaggle web site every day.

5 Requirements and What to Submit

- You can form teams of at most 3 students. You can work alone but we encourage everyone to be part of a team. Team members can be from different sections.
- Codes should be written in C++. You are allowed to use C++ standard library. No other library can be used.
- In the project report you should clearly explain the data structures and algorithms you used. Also, you should clearly write which parts of the project you completed. Also, provide the time it takes to make all the predictions (omit the time for reading the ratings file). Designing efficient data structures and algorithms will lead to higher grades, so describe them in detail.
- You should submit a video recording in which every team member should explain his/her part. Recordings should be about 10-15 minutes long.

You should submit (to itslearning) the following as three separate files in the given formats. **Submissions in other formats will not be evaluated**. Every team should make a single submission so write team member names clearly at the top of your reports. There will be a penalty of -10 points for every late day after the due date.

- All header (.h) and source (.cpp) files. (zip format)
- Project report (PDF format)
- Video recording (mp4 format)

6 Grading

- **40 points** Reading the input file and correctly printing the top 10 users and top 10 movies in decreasing order of number of ratings that they have. An example output is given in the Tables 2 and 3 below:

Table 2: Top users who have the most ratings.

	# Ratings
User 126	68
User 8172	62
User 218	56
...	

Table 3: Top movies which have the most ratings.

	# Ratings
Movie 226	127
Movie 4172	114
Movie 2213	108
...	

- **40 points** Successfully applying the UBCF or IBCF, making a submission to Kaggle web site, and getting $RMSE < 1.0$. Note that choosing efficient data structure and algorithms and having lower running times will also be considered. Teams with the same RMSE score can get different points depending on the data structures and algorithms they implement.

- **20 points** You will be ranked according to your RMSE scores. The top team(s) will get 20 points, the bottom team(s) will get 0 points, and the others will get points between 0 and 20 according to their ranks.

7 Cheating Policy

You are not supposed to use each other's source code. Also do not use source blocks of code from Internet, another person or from a textbook.

All the source codes will be filtered through a similarity analysis tool, which is known to be effective against many types of code copying and changing tricks. If a cheating is detected, these projects will be graded as 0 and also university regulations will apply.

Also, methods (such as getting a good RMSE by trial and error) which do not use valid recommender systems techniques but which get good scores will get partial grades according to the work that has been done.