TSKS11 Hands-On Session 6

Fall 2018

In this computer project, we will study recommendation systems of the type that are used by, for example, Netflix. For Netflix, the objective of the recommendation system is to predict how much a user of the service would appreciate a certain movie and then recommend to each user those movies that the user is likely to enjoy. The prediction is based on how the user in question has rated other movies, and how other users have rated the different movies. The theory needed for this project is described in [1, Chap. 4].

About the data files. In this project, we use real data from the MovieLens project (`www.movielens.org`). MovieLens is a recommendation website where users can rate movies they have seen and get personalized recommendations for other movies. We gratefully acknowledge the GroupLens Research Project faculty at the University of Minnesota, USA, who have kindly given us permission to re-distribute parts of their data to be used for this project work. We stress that the data files represent derivative works of the MovieLens database, obtained by randomly permuting the data and removing certain elements, and must not be mistaken for the original MovieLens dataset. **The data files must not be distributed beyond the audience of the TSKS11 class.**

Summary of project tasks. This project consists of two parts that should be solved and reported in written form, as described in what follows.

All course participants will obtain a unique data file, `XXXXXXXX.zip`, that can be downloaded from the course webpage. (`XXXXXXXX` here are the first eight letters in the particpant's name.) This ZIP file contains six plaintext files: `XXXXXXXX.training`, `XXXXXXXX.test`, `XXXXXXXX.moviename`, `task2.training`, `task2.test` and `task2.moviename`.

Task 1: The Baseline Predictor

The first task is to implement the baseline predictor presented in [1, Sec. 4.2.1]. To do so, you should use the training data set in the file `XXXXXXXX.training`, which contains

ratings made by $U = 2000$ users on $M = 1500$ movies. In total, the file contains roughly 200000 ratings. To evaluate the predictor you should use the test data set in the (individual) file `XXXXXXXX.test`, which contains roughly 20000 ratings. Each rating is a number from the set $\{1, 2, 3, 4, 5\}$. The files are stored in comma-separated value (CSV) format, and each row in the `.training` files has the format

```
u,m,r
```

where `u` is a user ID number, `m` is a movie ID number, and `r` is user `u`'s rating of movie `m`. The movie names in plaintext are listed in the file `XXXXXXXX.moviename`.

Baseline predictor. The baseline predictor models the data according to

$$\hat{r}_{um} = \bar{r} + b_{U,u} + b_{M,m} \tag{1}$$

where

- $\hat{r}_{um}$ is the predicted rating for user u of movie m,
- $\bar{r}$ is the average rating over all users and movies,
- $b_{U,u}$ is the bias of user u compared to the average $\bar{r}$,
- $b_{M,m}$ is the bias of movie m compared to the average $\bar{r}$.

The goal is to find $b_{U,u}$ and $b_{M,m}$ that minimize the root mean-square error (RMSE).

Task. Implement the baseline predictor by solving the a least-squares problem as explained in [1, Sec. 4.2.1]. The matrix $\boldsymbol{A}$ and the vector $\boldsymbol{c}$ should be constructed based on the data in `XXXXXXXX.training`. After the optimal $\boldsymbol{b}^\star$ has been found, the predictor should be tested on the data sets in `XXXXXXXX.training` and `XXXXXXXX.test`.

Make sure to provide

(i) the RMSEs associated with the training data and with the test data. Do not round the predicted ratings to integers. However, truncate the predicted ratings by setting any prediction that falls below 1 to 1, and any prediction that exceeds 5 to 5. The RMSE values shall be reported with three decimals (`x.yyy`).

(ii) a histogram of the absolute errors (deviations between the predicted ratings and the true ratings), for the test data. When plotting this histogram, first round each predicted ratings to the nearest integer in $\{1, 2, 3, 4, 5\}$. In view of the RMSE value computed in (i), does the plot look reasonable?

(iii) an explanation, in your own words, for why it is important to evaluate the predictor on test data that was not used to construct the predictor.

Task 2: Improved Predictor

Implement an improved predictor by using either one or more of the methods discussed in [1, Chap. 4], or any other reasonable method that you can come up with. At a minimum you should implement one of the basic neighborhood methods presented in [1, Sec. 4.2.4], see Equations (4.6)–(4.7). The goal is to achieve as low RMSE as possible. For this part, you should use the training data in `task-2.training` and the test data in `task-2.test`. (These files are the same for all students.) The data are stored the in the same format as the files used in Task 1.

Please note that solutions with an "improved" predictor that is worse than the baseline predictor will not be approved.

Hints

For task 2, if you choose to implement the "neighborhood method", the following hints might be useful. Let's assume you are using the cosine similarity for movies (if you're using the cosine similarity for users, the reasoning is analogous).

Have a look at Eq. (4.6), [1, Sec. 4.2.4].

In calculating the cosine similarity d_ij, the textbook tells us that the summation is "of course only over those users u that rated both movies i and j in the training set". Note, however, that there might be only a single user that have rated a specific pair of movies, so that there is only one term in the sum.

What is d_ij in that case? The nominator and denominator will cancel out each other, so d_ij = 1. This turns out to be the case for quite a lot of movie pairs in the training set. Since d_ij is between -1 and 1, this would be the maximum value.

Now we are supposed to sort the absolute values |d_ij| in descending order and pick the L largest ones. But if many of these values are 1, these will of course be among the top L, and this would suggest that these movies were extremely similar – whereas in reality what was actually going on was that they were only rated by a single user!

Therefore, introduce a minimum number of users, u_min, that must have rated both movies i and j for the sum to be valid. For instance, d_ij is calculated as normal if at least u_min = 10 users rated both movies, and otherwise d_ij = 0 (which is the most neutral value – neither similar nor dissimilar).

By doing so it might make sense to increase L. You might want to experiment with different values of L and u_min to see what gives you the lowest RMSE. For example, u_min = 10 and L = 750 may work fine, but please play around – to choose these parameters is more a form of art (meaning essentially experimentation) than it is hard science.

Examination

- The program code you have written should be submitted to Urkund: `ollab13.liu@analys.urkund.se`.
- Collaboration on this homework in small groups is encouraged, but each student should perform programming work individually, and individually demonstrate understanding of all tasks.
- Library functions from SNAP, Matlab and the Python standard library may be used freely, but copying of code from other libraries and/or toolboxes, from the Internet, from other students, or from previous years' students is prohibited.
- Individual oral examination takes place in class (computer lab).

References

[1] M. Chiang, *Networked Life: 20 Questions and Answers*. Cambridge University Press, 2012.