Data Mining 2020-2021

Netflix Challenge: Movie rating prediction

Description

The goal of this project is to **develop a recommendation algorithm for movies**.

The data for this project comprises **910,190 ratings** (on a **1 to 5 scale**) that were given by **6,040 users** to **3,706 movies**. Next to the ratings, the data contains some information on the users (**gender**, **age**, and **profession**) and information on the movies (**title and year of release**). You have to develop an algorithm that, based on this data, predicts as good as possible what rating a particular user will give to a particular movie.

Instructions

Use your @tudelft.nl email address to make an account on kaggle.com. Once your account has been created, you can go to the Kaggle page and start downloading the data. Further information are provided in the Kaggle page and below.

Be aware that the Kaggle ID that you choose will later be used in your report.

You should develop your algorithms in **Python**, and we offer TA support in Python. For your convenience, we will provide Python-code (in Python 3.6) that reads in all data, runs a very simple prediction algorithm, and writes the results into a submission file.

Files and data

Users.csv: Information about the users

This file contains four columns: (1) a column indicating the index of the user; (2) a column indicating whether the user is male or female; (3) a column indicating the age of the users (if known); and (4) a column containing a number that indicates the profession of the user.

Movies.csv: Information about the movies

This file contains three columns: (1) a column indicating the index of the movie; (2) a column containing the year the movie was released; and (3) a column containing the title of the movie.

Ratings.csv: User-movie ratings

This file contains three columns: (1) a column indicating the index of the user who gave the rating; (2) a column indicating the index of the movie that was rated; and (3) a column containing the rating that the user gave to the movie on a scale from 1 to 5.

Predictions.csv: List of user-movie ratings that needs to be predicted
The file contains two columns: (1) a column indicating the index of the user who gave the rating and (2) a column indicating the index of the movie that was rated. For these user-movie combinations, your algorithm needs to predict the ratings. These predicted ratings need to be written into the submission.csv file.

Submission.csv: Example of a submission file

The file that you need to write contains two columns: (1) a column indicating the corresponding row in predictions.csv ranging from 1 to 90,019 and (2) a column indicating the predicted ratings. Note that this file uses a comma as column separator (all other files use a semi-colon instead); this is the Kaggle default.

Algorithm performance

To be able to gauge how well your algorithm works, we have held out a total of 90,019 user-movie ratings. You are supposed to **make predictions for these ratings**; your predictions will be compared with the true user-movie ratings. To this end, we will measure the **root mean squared error** (RMSE) of your algorithm. Given that P are your predicted values and O are your observed values the RMSE of the recommendation algorithm is computed as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (P_i - O_i)^2}{n}}$$

While developing your algorithms, you can measure their RMSE by writing your predictions to a CSV-file and uploading the resulting CSV-file to the Kaggle-in Class website. The website computes the RMSE of your algorithm on a random subset of the 90,019 held-out ratings, and puts your result on the public leaderboard. This allows you to compare your progress with that of other students. **No more than 15 submissions can be done per day to avoid overfitting.**

The final score of your algorithm will be computed after the end of the competition based on the predicted ratings that were not used to compute the score on the leaderboard. This procedure is chosen because it ensures that final scores are not the result of overfitting.

Step 1 Description

Exercise 0: Running python code

For these exercises, you will not make use of a Jupyter notebook to run your code. Instead, you will directly edit and run python files. There are two ways to proceed: 1. Use an IDE (Integrated Development Environment). You can create a free student account with JetBrains to download PyCharm for free: https://www.jetbrains.com/student/ 2. Edit your code in your favorite text editor (e.g. Notepad, Sublime Text, Notepad++) and run the code from the command line. We will explain this method below.

Virtual Environments

In any case, we will make use of virtual environments. Virtual environments give you a way to maintain dependencies and Python versions for multiple projects.

Say, for example you have been working on *Project A*, which is written in Python 2.7. You want to start on a new project, *Project B*, using Python 3.7. If you upgrade your Python installation to 3.7, you might not be able to run *Project A* on your computer anymore, unless you downgrade your Python installation. This is where environments come in: for each project, you can define a Python virtual environment. The environment defines which Python version to use (by storing the path to the corresponding Python installation) and what packages are available. Within each environment, you can manage the installation with a package manager, like conda or pip.

For this project, we will need an environment with Python 3.6, numpy and pandas.

NOTE: you are not allowed to use use any libraries, besides numpy and pandas.

Running python files

Now that you have an understanding of environments, let's create and use an environment. We will make use of conda to manage your environments. You should already have conda installed on your machine, as you used it for the previous assignments. Open up your terminal or command prompt and follow along!

1. To create a new environment, type:

$ conda create -n recommenders python=3.6 Press Enter, and answer y when prompted.

-n recommenders tells conda that the **n**ame of your environment should be *recommenders*, python=3.6 tells conda to use Python 3.6.

2. To activate the environment, type:

$ conda activate recommenders Press Enter.

3. You are now working inside the *recommenders* environment. To install numpy and pandas into this environment, type:

$ conda install numpy pandas Press Enter, and answer y when prompted.

4. You can now run the example python code using the command:

$ python code_template2021.py Press Enter.

- If you would like to deactivate the environment you are currently using, type:

$ conda deactivate This will deactivate the current environment and return you to the base environment. This is your default environment, which you have most likely used for all the previous assignments. Actually, you can make use of the base environment to run the example python code as well: it already has numpy and pandas installed.

- If you want to know what environment you are in, type:

$ conda info or $ conda list The latter will show which environment is active and what packages are installed in the environment.

Exercise 1: Collaborative Filtering

For this exercise you will create a recommender system, which predicts ratings for netflix movies on the dataset available on Kaggle.

Apply collaborative filtering to make your first predictions. You may use any distance metric you want to find the similarity between the users. It is recommended to

1. First create an user/movie matrix containing all of the ratings.

2. With this matrix you can compute the utility matrix containing the similarities between the users.

3. You can use these similarities to predict the ratings given in the Predictions file (*Predictions.csv*).

Exercise 2: Latent Factors

For this exercise you will make use of *Matrix Factorization* in order to find the latent factors of the rating matrix you created for exercise 1. You can use *Singular Value Decomposition* to create the decomposed matrices which you can use to make movie rating predictions using the dot product and average user ratings. This leaves a matrix with the predicted ratings for all users and movies.

Step 2 Description

From here the idea is to try out more different algorithms to improve your predictor. The following criteria will be used to assess the report part about the predictor in step 2 (see also the rubric):

1. The variety of algorithms you have tried
2. The creativity of the solutions you have implemented
3. The level of understanding of the algorithms you display in your report

Assessment

The challenge requires each participating **team of 2 students** to write a **2 page** report about his/her predictors (excluding a title page and the references). This report will be graded based on the **provided rubric.** The grade constitutes **35% of the final grade** of the course. The passing grade is 5.0.

Please note that **only your report is evaluated and not ranking on Kaggle**. If you implemented an algorithm in step 2 that turned out not to work as expected, please make sure to describe this algorithm in your report as well. If you have developed ideas about why certain algorithms do work and why other algorithms do not, please explain these ideas in your report as well.
We will have a look at your code though to verify whether the claims in your report correspond to your code.
Also, **please make sure to report your Kaggle name** so that it is clear which entry on the leaderboard your predictor corresponds to.

Project reports and code should be made in teams of 2. The cover sheet of your report should include your name, kaggle ID and student number.

Submit **1 zip file** consisting of your report and the code in the corresponding assignment on Brightspace. Deadline is on **22nd January 2021, 23:59.**
Late submissions are not possible.