Programming for Data Analytics Project Report: Recommender system competition

1. Description of the project

The project consists of a Kaggle competition, based on the prediction and recommendation of a list of 10 movies for a set of test users. The training datasets are potentially two:

- A dataset "*train-PDA2019.csv*" that contains around 700000 ratings for 5690 users
- A dataset "*content-PDA2019.csv*" that contains information about 1824 movies (visual, tag, genre features, ecc)

The final goal is therefore the implementation of a recommender systems that recommends a list of top 10 movies for each user in the test set. In this report I will present the solution I have chosen to get the most accurate recommendation possible.

2. First try: Surprise

The first experiment I did to face the challenge was based on the choice of a Python library with a set of recommending algorithms, the *Surprise* library. In this early phase of the project I focused on the **collaborative-filtering** technique, which makes recommendations based on the content preferences of similar users.

Before implementing the algorithm and perform the recommendation for the set of test users, I have applied a first comparison between the most used algorithms of the library, in order to choose the most performing ones. To achieve this goal and compare the recommenders I have split the training dataset and performed 5-fold cross validation of the following algorithms:

- KNN-Basic
- KNN-Baseline
- SVD
- SVDpp

At the end of the cross validation, I choose the best two recommenders in term of the best accuracy (based on Root Mean Square Error RMSE and Mean Absolute Error MAE), which resulted to be SVD and SVDpp.

Therefore, I have applied these two algorithms to recommend a list of top 10 movies for each test users, but the score of the recommendation was not satisfying (0.02761 and 0.3460 respectively). For this reason, I have decided to opt for another solution, as I will explain in the next paragraph.

3. Second try: Graphlab

As second and final solution, my choice fell on a second library: ***Graphlab*****,** which provides a variety of recommender models and algorithms in order to perform recommendation. For the purpose of the project, I have developed two scripts in python, one responsible for evaluating the most

suitable recommendation model for the training set, and the other for the implementation of the algorithm and creation of the top 10 movies recommendation csv file. The two scripts are available in my personal GitHub profile and are explained more in details in the following sections, providing also the links where the source code could be found.

Note: Graphlab is not supported on the most recent versions of Python, therefore the code should run on Python 2.7.

Comparison of the models: *ModelComparison.py*

As previously anticipated, the first script developed is responsible of performing an evaluation and comparison of the various algorithms available in the library to make recommendations. For seeing the source code of the script, click this link.

To perform the evaluation and comparison, I have randomly split the training set of users in two samples (test and set) and then applied 8 different type of algorithms, taking into consideration some changes in term of parameters. The algorithms taken into account are summarized in the following list:

- **R1**: basic algorithm + additional information of the movies
- **R2**: basic algorithm
- **R3**: factorization recommender
- **R4**: factorization recommender + additional information of the movies
- **R5**: ranking factorization recommender
- **R6**: ranking factorization recommender + additional information of the movies

For the evaluation, the criterion taken into account is RMSE (Root Mean Square Error) and this is the summary of the comparison:

Algorithm	RMSE
R1	1.0388622783137575
R2	1.093274792858811
R3	1.0230625430757982
R4	0.9845947037200079
R5	0.8401678373158883
R6	1.0402263251840038

Since the lower the RMSE, the more accurate a recommendation is on average, I have chosen the pure ranking factorization recommender. As a further inspection, I added two experiments on the ranking factorization recommender, trying to change the default number of latent factors (default = 32) taking into account during the training phase of the algorithm:

- **R7:** ranking factorization recommender + number of latent factor = 40
- **R8:** ranking factorization recommender + number of latent factor = 28

And the result is:

Algorithm	RMSE
R7	0.9969257141848981
R8	0.987663504525894

which shows a higher RMSE. As a conclusion, the chosen algorithm is the ranking factorization recommender using 32 number of latent factors to learn from.

Recommendation: *ProjectGraphLab.py*

The second script is responsible for the actual implementation of the algorithm on the given training test of users and the consequent creation of a list of top 10 movies for each test user. The source code can be found clicking on this link.

The chosen algorithm basically learns latent factors for each user and item and uses them to make rating predictions. First, the recommender model is trained based on the *train-PDA2019.csv* file that contains 700000 rating for 5690 users. After the model training, the list of test users is obtained and for each of them the model recommends a list of 10 movies.

The result is a SFrame object (similar to a pandas dataframe) with four columns:

- ID of the user
- ID of the item
- Predicted rating
- Ranking (integer between 1 and 10 since the algorithm is called after setting k = 10, i.e., after specifying that we are interested in the best 10 movies)

The last lines of the script are responsible for processing the resulted SFrame to create a suitable pandas dataframe and export it in a csv file: each user is grouped together and the top 10 movies ranking are summarized in a string using the *join* function to the list object.

The resulted recommendation obtained the final score of 0.06928.

Final considerations

After the submission of the previously descripted recommendation, I have tried other different solutions, including hybrid recommender systems that combine content-based and collaborative filtering, item similarity recommendation based on cosine similarity of movies, cleaning and pre-processing of the given dataset.
However, the score obtained by this techniques were lower than the score obtained by the ranking factorization recommender algorithm provided by the GraphLab libray, which represents therefore the chosen and final solution.