Assignment 1
Recommender Systems

Wojtek Kowalczyk
wojtek@liacs.nl

2-10-2023

Introduction

In this assignment you will work with the *MovieLens 1M* dataset which can be fetched from `https://grouplens.org/datasets/movielens/1m/`. This set contains about 1.000.000 ratings given to about 4.000 movies by about 6.000 users. Additionally, some information is provided about movies (genre, title, production year) and users (gender, age, occupation). Your task is to implement in Python several recommendation algorithms that have been discussed during the course, and estimate their accuracy with the Root Mean Squared Error, RMSE, and the Mean Absolute Error, MAE. Additionally, you will have to visualize vector representations of users and movies (generated by the matrix factorization algorithms) with PCA, t-SNE and UMAP algorithms.

To make sure that your results are reliable use 5-fold cross-validation. In other words, split your data set at random into 5 equal size pieces and use each of this piece as a test set, while training the model on the remaining 4 pieces. The average error of these five models (measured on the 5 test sets) is a reliable estimate of the accuracy of the (hypothetical) final model that is trained on the whole data set.

It may happen that some users or movies that appear in a test set have not occurred in the training set. How will you handle such cases?

The algorithms that you should implement are:

- **Naive Approaches:** the 5 formulas from slide 17: the global average rating, the average rating per item, the average rating per user, and an "optimal" linear combination of the two averages (with and without the bias term γ).
- **The UV matrix decomposition** algorithm as described in Chapter 9.4 of the MMDS textbook.
- **The Matrix Factorization** with Gradient Descent and Regularization as described on the slide 39 and in the paper `gravity-Tikk.pdf` (the beginning of section 3.1). Note that this algorithm requires tuning the regularization factor λ!

Details

Cross-validation

We are interested in the accuracy of recommender systems on the data that was not used in the training process. Therefore, you are required to apply the 5-fold cross-validation scheme. It means that you should split your available data, at random, into 5 parts of more or less equal sizes and develop 5 models for each combination of 4 out of 5 parts. Then, each model should be applied to the part that was not used in the training process. In this way you will generate 5 different estimates of the accuracy; their average is considered to be a good estimate of the error on the future data. You may compare your results to the results reported at `http://mymedialite.net/examples/datasets.html`. We advice you to start with applying the cross-validation scheme to the simplest recommender: the overall average score. Write a script (or a function) that splits the data (at random) into 5 folds, constructs 5 models (each one consisting of just one number: the average score) and applies these models to the training and test sets, generating predictions and calculating errors. Finally, average errors over the training sets and over the test sets to get estimates of the accuracy of your recommender, both on the training and the test sets. Repeat this process for every other recommendation algorithm. To make sure that your results are reproducible, when splitting the data in 5 folds, explicitly set the random seed to a specific value.

Naive Approaches

The "average rating" recommender requires no further explanation. However, when building models for "average user rating" or "average movie rating" you must take into account that during the sampling process some users or some movies might disappear from the training sets – all their ratings will enter the test set. To handle such cases, use the "global average rating" as a fall-back value.

Concerning the linear regression models, experiment only with the "full" variant of linear regression (i.e., include the γ parameter):

$$pred = \alpha \cdot avg_{user} + \beta \cdot avg_{movie} + \gamma$$

You may use here the numpy `np.linalg.lstsq` function.

Additionally, improve predictions by rounding values bigger than 5 to 5 and smaller than 1 to 1 (valid ratings are always between 1 and 5). Which fall-back values will you use when user or movie average rating is not available? Describe it in your report!

Thus there are 4 naive approaches: global average, user average, movie average and a linear combination of the three averages (with fall-back rules).

UV Matrix Decomposition

Implement the algorithm as described in Section 9.4 of the MMDS textbook `http://infolab.stanford.edu/~ullman/mmds/ch9.pdf`

Matrix Factorization

The implementation of this algorithm is relatively straightforward. Check the blog of S. Funk, `http://sifter.org/~simon/journal/20061211.html`. Note that you should implement the

algorithm that is described in the `gravity-Tikk.pdf` paper – their version of the algorithm is better than the one proposed by S. Funk.

Keep in mind that running the Matrix Factorizaton algorithm may take hours (a single pass through the training set could take a couple of minutes). Therefore, instead of running multiple runs in search for optimal parameters, run at least one experiment with the parameters that are reported on the MyMedialite website:

`num_factors=10, num_iter=75, regularization=0.05, learn_rate=0.005`.

Optionally, as you probably work with computers that have multi-core CPUs think about running several experiments in parallel - for example, the 5-fold cross validation can be distributed along 5 independent threads.

Once you successfully completed your matrix factorization experiments, save the feature matrices (both for users and movies) - you will have to use them for 2-d visualizations.

Data visualization

The matrices of features which characterize users and movies can now be used for visualization. Apply the most popular dimensionality reduction algorithms: **PCA**, **t-SNE**, and **UMAP** to visualize these matrices. For this purpose use available python libraries like *sklearn*, *umap*, *matplotlib*, *seaborn*, etc.

Use different labeling/coloring schemes for your data points. For example, you can label your movies based on genre or year of movie release to see whether any meaningful clustering is present. You should do the same for users, focusing on their age or gender. These movies and users attributes can be fetched from `https://files.grouplens.org/datasets/movielens/ml-1m-README.txt`.

The Report

Submit your work in the form of two Jupyter notebooks that correspond to two parts of the assignments: recommender systems and data visualization.

The deadline for this assignment can be found on BrightSpace.