Assignment 1
Recommender Systems

Wojtek Kowalczyk
wojtek@liacs.nl

20-09-2022

Introduction

In this assignment you will work with the *MovieLens 1M* dataset which can be fetched from `http://grouplens.org/datasets/movielens/`. 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. 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.

This assignment consists of two parts:

1. Building recommender systems based on the following algorithms:
   - **Naive Approaches**: the 5 formulas from slide 17 (lecture 2): the global average rating, the average rating per item, the average rating per user, and an “optimal” linear combination of the two averages (per user and per item), with and without the parameter γ.
   - The **UV matrix decomposition** algorithm as described in Chapter 9.4 of the textbook.
   - The **Matrix Factorization** with Gradient Descent and Regularization as described on slide 39 (lecture 2) and in the paper `gravity-Tikk.pdf` (the beginning of section 3.1).
2. Exploratory data analysis: using dimensionality reduction techniques to visualize your data.

1 Recommender systems

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.

1.1 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 with the "full" variant of linear regression (i.e., include the intercept parameter γ):

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

and a simplified variant (without the intercept γ):

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

In both cases you may use the scikit-learn package for linear regression: `sklearn.linear_model.LinearRegression`.

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 5 naive approaches: global average, user average, movie average and a linear combination of user and movie averages (with fall-back rules).

1.2 UV Matrix Decomposition

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

1.3 Matrix Factorization

Implement the "classical" Matrix Factorization algorithm. The implementation of this algorithm is relatively straightforward. Check the blog of S. Funk, `http://sifter.org/~simon/journal/20061211.html` to understand the key ideas behind this algorithm. However, you should implement a slightly more complication version of this algorithm which 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.`

Finally, 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.

Tip: Once you successfully complete your matrix factorization procedure save the resulting feature matrices (for both users and movies) - they are useful and they are going to be needed for the second part of the assignment.

2 Data visualization

Visualizing the underlying patterns in your data is an essential skill in numerous occupations. Since datasets often have dozens of separate features, dimensionality reduction techniques can provide valuable insights on how your data is distributed and whether it is clustered in some potentially meaningful manner.

In the first part of this assignment you used the Matrix Factorization algorithm to produce artificial features that describe your movies and users. Not only can these features be used to build recommender systems but also to compare or cluster users and movies.

In this task you should produce a Jupyter notebook with the following content:

1. Use **T-SNE** and **U-MAP** and **PCA** algorithms to visualize your movie and user features in 2D (*sklearn, umap, matplotlib, seaborn* libraries). Read about these algorithms, try to understand on how they work under the hood and write a short summary on them. Additionally, try to qualitatively describe the differences in the output of these algorithms.

2. Try different labeling/coloring schemes for your datapoints. For example you can label your movies based on genre or year of release and see whether any meaningful clustering is present. You should do the same for users (age/gender visualization).

The Report

In your report describe experiments that you've performed (including values of various parameters that you've used in your simulations), the accuracies achieved (both on the training and the test data) and the actual run time. To make sure that your results are reproducible, explicitly initialize random seeds to your favourite values. (There are two places where randomness plays a role: splitting data into folds and initialization of factors in the Matrix Factorization approach.)

What can you say about the required time and memory of the implemented algorithms? How quickly the required CPU-time and memory would grow with the number of movies, M, the number of users, U, and the number of ratings, R? Informally, the "required time" is proportional to the number of steps a program has to execute and the "required memory" is proportional to the number of bytes that are used by a program. Traditionally, time and memory requirements are expressed with help of the "big O" notation, see `en.wikipedia.org/wiki/Big_O_notation`. For example, multiplying an $M \times N$ matrix by an $N \times K$ matrix requires $O(MNK)$ time and $O(MK)$ memory. Indeed, the result of such a multiplication is an $M \times K$ matrix, and calculating any entry of this matrix requires N multiplications and $N - 1$ additions. Formulate your answer to this question using the "big O" notation!

When estimating the amount of memory needed to run your algorithms, keep in mind that in "real life" the training sets might be huge! Fortunately, we don't have to load these sets into RAM–it is sufficient to process every rating one by one, perhaps several times. For example, to calculate the global mean we only need to scan the data once, summing up all the ratings (one float, Sum) and counting them (one integer N). The global mean is then S/N, so we need memory to store only 2 numbers, i.e., the required memory is $O(1)$.

The submission procedure

This assignment should be made in groups of size 3. To submit the assignment, you and your teammates must be enrolled in a practical group on Brightspace. You are allowed to make as many submissions as you want before the deadline; we only store your most recent submission. We will run a plagiarism check on your report and code, so make sure not to copy work from other students or websites!

Your *final submission* must be a **ZIP file** containing the following two things:

- A folder "*part1*", containing all your Python code for the first part of the assignment in one of the following formats:
  - .py scripts along with README.txt with instructions on how to reproduce your results.
  - A Jupyter notebook
- The data visualization (or exploratory analysis) part of the assignment should be contained in a Jupyter notebook named *'data_visualisation.ipynb'*.
- Your report in PDF format, called "*report.pdf*"

The deadline for this assignment will be announced shortly on the Brightspace.