11-741/11-641/11-441 – Machine Learning for Text Mining **Due: Feb 25, 2020, 11:59pm**

Instructor: Yiming Yang

# Homework 2: Collaborative Filtering

This assignment will give you experience in implementing a collaborative filtering system for movie ratings. We will be using a subset of the data provided by Netflix. The assignment consists of three major parts:

- A program that predicts movie recommendations for a user
- Experiments with the program
- A report describing your program, the experiments, and your conclusions.

## 1 The Dataset

We will use a sample of the Netflix Prize dataset. We have gotten permission from Netflix to re-distribute this data for classroom purposes, and **we need to adhere to the following guidelines**:

- You MAY NOT distribute the Netflix Prize subset to anyone outside of the class.
- You MUST remove the dataset from your computer when the class is over.
- You MUST adhere to the guidelines in the netflix-README file provided by Netflix. [The files training_set.tar and movie_titles.txt mentioned in the netflix-README have been processed for you in the HW data zip file.]

For details and background of this dataset or the Netflix Prize competition, see:

- a short paper on the Netflix Prize and dataset written by the people at Netflix running the competition
- proceedings from a scholarly conference (KDD-Cup 2007) also using Netflix data

## 2 The Program

Your program is responsible for doing the following:

1. Reading in the data from the Netflix Prize subset
2. Producing a prediction for the rating of each user-movie pair in the development/test file
3. Writing those predictions to a log file using the specified format. The format is described in a README – see the handout file for details.

You may want to pre-process the data to make it easier for your program to perform certain functions. For example, in the original data format it is somewhat difficult to look up all the ratings by a given user. It is perfectly OK to do some pre-processing on the dataset and probably a good idea since we're planning on experimenting with a variety of rating algorithms. How you pre-process the data is up to you. In the report, please document anything you do to pre-process the dataset. For more details on making a rating prediction, see the experiments section.

Please recall the importance of imputation when performing collaborative filtering. **Empty cells should not be ignored!** As seen in class, there are several approaches to imputation – we will be asking you to use **Option 2 from the lectures slides**[1].

# 3 Corpus Exploration

Once again, we will be having you examine the corpus for some statistics on the **training set**. Please refer to the report template for specific details.

**Hint:** For the total number of movies and users, you only need to count those actually exist in the training set, i.e., the total number might be less than the maximum ID.

# 4 Input/Output

The data for this assignment can be found in the **data** directory of the handout. It consists of the training, development, and test sets. A README is included with the data; you should read this file before beginning your implementation. It will describe all of the formatting in this assignment, for both input and output.

We have provided a standard evaluation program for you to use. This program computes a single evaluation metric typically used in collaborative filtering, Root Mean Squared Error (RMSE). RMSE is calculated as follows:

```
E = 0
For each rating
        E = E + (true_rating - rating)^2
RMSE = sqrt(E / num_ratings)
```

You may test your system's performance on the development set gold-standards using the script provided in the **eval** directory of the handout or you could evaluate it using the Gradescope. We will not provide any feedback on the test set until after the homework deadline.

# 5 The Experiments

You have to implement several different rating-prediction algorithms. The first two experiments are based on the memory-based and item-based approaches seen in lecture. The third experiment will ask you to consider rating bias of users and items. In the final experiment, you will be asked to implement an algorithm based on matrix factorization.

## Experiment 1. User-user Similarity

Do the imputation using Option 2 taught in class (i.e. subtract 3 from each of the non-empty cells).
The prediction is based on user-user similarities. A description of the algorithm is as follows:

Input: a movie, user pair: `p = (movieA, userA);` `Output`: a rating `r` in the range `[1,5];`

1. Find the `k` nearest-neighbors of `userA` using the dot product similarity and cosine similarity.
2. Produce a prediction of the rating for `p` based on the `k` nearest-neighbors of `userA` using both the mean movie rating and the weighted mean movie rating. **Remember to adjust your prediction to be in the range [1,5].**

[1] Keep all the empty cells unchanged, but subtract 3 from each non-empty cells.

Sort the nearest-neighbors with the same similarity in terms of the `UserID`. That is, you **use UserID for tie-breaking**. In Step (2), you need to implement two rating methods:

- The mean rating is the average rating of this movie among the neighbors
- The weighted mean rating for a movie among the neighbors uses the similarity measure from step (1) as the weight. Do not forget to include a normalization factor.

  **Hint:** Use absolute values when computing the divisor of the weighted sum, i.e., $\frac{w_i}{\sum_{i \in knn} |w_i|}$.

## Experiment 2. Movie-movie Similarity

Again, do the imputation using Option 2 taught in class. The prediction is based on item-item associations (item-based CF). A description of the algorithm is as follows:

Input: a movie, user pair: `p = (movieA, userA);` Output: a rating `r` in the range `[1,5]`

1. Compute the matrix $A$ of item-item associations ($A = X^T X$) using the dot product similarity and cosine similarity.[2]
2. At testing time, compute the k-nearest neighbors of `movieA` using the matrix $A$.
3. Produce a prediction of the rating for `p` based on the `k` nearest-neighbors of `movieA` using both the mean user rating and the weighted mean user rating (see the slide, Item-based CF at Testing Time, in the first lecture on CF for details). **Remember to adjust your prediction to be in the range [1,5].**

   **Hint:** Use absolute values when computing the divisor of the weighted sum, i.e., $\frac{w_i}{\sum_{i \in knn} |w_i|}$.

## Experiment 3. PCC-based methods

It is often wise to consider that different movies are inherently more popular than others (i.e. have a higher average rating) and different users are more generous than others (i.e. give a higher average rating). If we would like to take this into account with our rating prediction step, we need to account for this movie- or user-bias. Just like Pearson's correlation coefficient takes into account the movie or user mean rating when calculating similarity between movies or users, we can normalize for this rating bias when making the prediction. In this experiment, your implementation must apply user-rating and/or movie-rating standardization. We are leaving the exact formulation of this standardization as an exercise for you in this assignment. You can implement the standardization based on either Experiment 1 or Experiment 2. **Please include the details of your implementation in your report.**

## Experiment 4. Matrix Factorization

In this experiment, you are going to implement Probabilistic Matrix Factorization (PMF) as an alternative to the nearest-neighbor-based approaches. While there exists many variants of PMF, for this homework assignment your objective function should be the same as eq. (4) in the original PMF paper:

Ruslan Salakhutdinov and Andriy Mnih. "Probabilistic matrix factorization." Advances in Neural Information Processing Systems (NIPS). 2007.

We give you the freedom of choosing the optimization algorithm. E.g., you may use any one of Gradient Descent (GD), Stochastic Gradient Descent (SGD) or Alternating Lease Squares (ALS). It's okay to use optimizers and automatic differentiation facilities provided by PyTorch or Tensorflow. You are going to report the performance of PMF on the development set with different inner dimensions (number of latent

[2]Note that you are not required to explicitely compute the whole matrix.

factors). For each given inner dimension, it is your responsibility to tune the model hyperparameters and the learning rate to ensure that your algorithm has correctly converged to a meaningful local minima.

**You must implement Experiments 1 and 2 using the exact algorithm described above** as it is easier for us to verify your implementation. For Experiment 3, you can implement an algorithm based on either Experiment 1 or Experiment 2, but please document the approach you choose. For Experiment 4, please document your optimization algorithm for PMF. You must conduct all experiments with your program and report the "wall clock" running time and the RMSE as computed by the evaluation script. **An experiment may take from several minutes to several hours depending on your implementation, so do NOT wait until the last minute to start your experiments**.

# 6 Handout

A handout.zip is provided to you with all the required files.

```
handout.zip
├── data
│   └── [Netflix data]
├── eval
│   └── [evaluation scripts and dev labels]
├── src
│   └── utils.py
└── HW2-Template.docx
```

We provided some sample I/O utils in **src/utils.py**, but it's not required to follow the provided sample.

# 7 Grading

1. **[80 pts]** Report: You must describe your work and your analysis of the experimental results in a written report. A report template is provided. Please address all the questions in this template report within the specified section. Please do not change the section headings. Do not list your source code in the report. Please make it easy for the TA to see how you have addressed each of the requirements described for each section.

   Please note that the quality of your report is important; so please allocate a sufficient amount of time to writing your your report. An ideal report should be clearly written, well organized, and sufficiently detailed. In general, one sentence answers or analyses are not considered to be sufficiently detailed. The TA reserves the right to deduct up to 20% of the grade based on the quality of the report.

2. **[10 pts]** Test set performance: You need to include the ratings on the test dataset using your best algorithm; use your best judgment as to which system is the best (development set performance is typically a good indicator). The format is the same as required for evaluating with the provided script. Please submit your predicted ratings on the test dataset to the Gradescope (please name predicted ratings for test dataset as **test-predictions.txt**, predicted ratings for dev dataset as **dev-predictions.txt**). You score will be based on the following criteria on the **test leader board**.

| RMSE | Score |
|---|---|
| $0.0000 \leq x \leq 0.9300$ | 10 pts |
| $0.9300 < x \leq 0.9400$ | 9 pts |
| $0.9400 < x \leq 0.9500$ | 8 pts |
| $0.9500 < x \leq 0.9600$ | 7 pts |
| $0.9600 < x \leq 0.9800$ | 6 pts |
| $0.9800 < x \leq 1.0000$ | 5 pts |
| $1.0000 < x \leq 1.0500$ | 4 pts |
| $1.0500 < x \leq 1.1000$ | 3 pts |
| $1.1000 < x \leq \infty$ | 2 pts |

**Hint:** Although we made several restrictions in the experiments section (such as requiring Option 2 imputation in Exp 1 & 2 or using eq. (4) for PMF), it's okay to use different variants in final prediction for leaderboard. But you must document such choices in the report.

3. **[10 pts]** Source Code: Your submission must include all of your source code, and any files necessary to make and run your source code. Your source code must run within our testing service, so please check that it does before you make your final submission. The TA will look at your source code, so make sure that it is legible, has reasonable documentation, and can be understood by others. Note: although code documentation is worth only 10%, failing to submit source code will result in a zero grade.

# 8 Restrictions

By default, the code will be tested on Ubuntu Linux 18.04 and Python 3.6.7, please make sure your code can run successfully under such environment. If you plan to use a different version please document the choice in a README.txt file. If you plan to use other programming languages, please discuss with TAs in advance.

1. You should not include datasets in the submission. TAs will run ./hw2.sh in the **src** directory. Your program should assume the data files are stored in **src/data** and generate all the experiment results automatically. In addition, it should generate **test-predictions.txt** in the **src** directory for test predictions. (It's okay that the generated files do not match the submissions exactly on the leaderboard due to uncontrollable randomness, but you should use the same algorithm to generate the files.)
2. Please test the script to verify that you include everything necessary to run it. You must write all of the software yourself.
3. It's okay to use scientific packages such as Numpy and Scipy for matrix multiplication, but you must write the algorithm by yourself. For the PMF, it's okay to use automatic differentiation facilities provided by PyTorch or Tensorflow. If you are unsure whether a particular API could be used, ask about it on Piazza. You must include the package dependency files such as requirements.txt or environment.yml if you used an package.

# 9 Submission Checklist

The report and the source code should be submitted separately.

For the source code, please compress all the following files into a .zip file for submission.

1. All source code, script and data in **src** folder
2. **dev-predictions.txt** and **test-predictions.txt** for online testing. Note that their location is different from the ones generated by **hw2.sh**. The test set will be used for grading.

3. (Optional) README

Folder structure:

```
HW2.zip
├── dev-predictions.txt
├── test-predictions.txt
└── src
    ├── hw2.sh
    ├── source files
    └── data
        └── [empty]
```

Do **not** actually upload the **data** directory, but your program should assume the dataset exist under that directory when hw2.sh is executed.

# 10 Frequently Asked Questions

- Can we load the data into some sort of database?

  Yes. Whatever makes it easiest for you. This subset was created specifically so that you could keep the entire dataset in memory, but if you're more comfortable with a database go right ahead. As long as your code runs on our system, this is not an issue.

- Wow, is this slow! Is there anything I can do to speed up the computation?

  Yes! Here are some tips:

  1. The dataset was constructed so that you could load it all into memory. If you're constantly reading it off of disk, consider using efficient in-memory data structures instead. If you find the similarity matrix is too big to fit in the memory, consider using the sparse matrix representation.
  2. You don't need to compute every movie-movie or user-user similarity. You're just concerned about the movies and users in the query file.
  3. It's also a good idea to cache similarities to avoid computing them more than once. All similarity measures should be symmetric, so sim(a, b) = sim(b, a).

- What about the dates in the training data? Do we need to use those?

  No, you don't need to look at those at all. They are a part of the original data, but will not be used for this assignment.

- A small subset of users/items do not occur in the training set. How should we account for this?

  If a user (item) does not occur in the training data, then the user is basically represented by an empty (all zero) vector. It will not have any specific nearest neighbor under either the dot product or cosine similarity. And you also could not compute a prediction for them using PMF. For these cases, you can pick any reasonable method of handling such cases; for instance, if you are asked to predict how an unknown user would rank movieA, you can use the average score of movieA across all users in the training data. Please document such choices in your report.