COEN 169 Web Information Management

Project II (100 points + 20 bonus points)

Deadline 1: First submission, 5:30 PM, Tuesday February 23rd
(Before class on Tuesday, Week 8)

Deadline 2: 11:59pm, Friday March 12th
(Friday, Week 10)

Introduction

In this project, you will develop different algorithms to make recommendations for movies.

You are free to choose *any programming language that you like* such as Python, Go, R, Java, C/C++, or Matlab. **You are only allowed to use features from the standard library.** In other words, you aren't allowed to use any packages that would require installation. (e.g. in Python, Collections would be acceptable, but NumPy would not be).

- I personally recommend you use either Python or Go.
    - Python is an excellent language for interviewing and is definitely worth learning
    - Go is an excellent modern systems language; it's a more reasonable version of C
    - When I took this class, I wrote this in Python-- nowadays I would write it in Go
- I personally recommend you stay away from C
    - Odds are this is the first language you learned, and the one you've written the most code in, but the small standard library will force you to write much more code
- I will not debug your code for/with you
    - I'm happy to go over how the concepts we discuss in class work.
    - You are going to be developing a fairly large system (at least for a class project) and as a result you will be the most qualified to understand and resolve bugs.
    - Consider how your system will work. A diagram of all the parts worth the time!
        - Although you are not graded on performance, even at this scale, a single-threaded, single-process system will be slow (especially if it's written in Python). Totally optional, but consider what parts could be parallelized. One obvious choice is the similarity calculations for k-similar-neighbors, but there are many potential speed-ups.
    - Consider how your code can be re-used. Classes and Functions are your friend.
    - This is an excellent project to put on your Github/resume-- take the time to make it a good example of your work.
- **Use source control!!!** (git)
    - Thank me later.
    - If you don't know what this is/how to use it, now is the time to learn.

Movie Recommendation System

The Training Data

The training data: a set of movie ratings by 200 users (userid: 1-200) on 1000 movies (movieid: 1-1000). The data is stored in a 200 row x 1000 column table. Each row represents one user. Each column represents one movie. A rating is a value in the range of 1 to 5, where 1 is "least favored" and 5 is "most favored". **Please NOTE that a value of 0 means that the user has not explicitly rated the movie**.

Please download the training data here: train.txt.

The Test Data

There are three test files: test5.txt, test10.txt and test20.txt.

[test5.txt] A pool of movie ratings by 100 users (userid: 201-300). Each user has already rated 5 movies. The format of the data is as follows: the file contains 100 blocks of lines. Each block contains several triples : (U, M, R), which means that user U gives R points to movie M. **Please note that in the test file, if R=0, then you are expected to predict the best possible rating which user U will give movie M.** The following is a block for user 276. (line 6545-6555 of test5.txt)

```
276 42 4      // user 276 gives movie 42 4 points.
276 85 2      // user 276 gives movie 85 2 points.
276 194 5     // user 276 gives movie 194 5 points.
276 208 5     // user 276 gives movie 208 5 points.
276 585 1     // user 276 gives movie 585 1 point.
276 4 0       // need to predict user 276's rating for movie 4
276 26 0      // need to predict user 276's rating for movie 26
276 33 0      ...
276 56 0
276 63 0
276 67 0
276 72 0
```

ATTENTION: Please make the prediction block by block: every time when you are making predictions for user U, please assume that you **ONLY** know the knowledge of the training data (train.txt) and the existing 5 ratings for this user. In other words, please DO NOT use the knowledge of any other blocks in the test file when making predictions. This will not be possible when running your algorithms against the test data.

The format of test10.txt and test20.txt is nearly the same as test5.txt, the only difference is that: in test10.txt, 10 ratings are given for a specific user; in test20.txt, 20 ratings are given for a specific user.

How to get the accuracy?

To get the accuracy of your predictions, please submit the predicted ratings to our online grading system. (For more information, please check the help page of our grading system.).
You can access the grading system via submit.html.

Hint: You can do some cross validations to pretest the performance of your algorithm. (that is to split the training data into your own training data and your own test data...)

Tasks

Your task is to design and develop collaborative filtering algorithms that predict the unknown ratings in the test data by learning users' preference from the training data.

Please complete the following experiments:

1. Submit your first run by the first deadline via submit.html. (10 points)

2. User-Based Collaborative Filtering Algorithms (40 points)

2.1 Implement the basic user-based collaborative filtering algorithms

Please implement two versions of the basic user-based collaborative filtering algorithm as the Cosine similarity method and Pearson Correlation method.

2.2 Extensions to the basic user-based collaborative filtering algorithms

Please implement the following two modifications to the standard algorithm (using Pearson Correlation): 1. Inverse user frequency; 2. Case modification.
In your report, include references (specific line numbers, snippets of code) to the most important code for this implementation.

3. Item-Based Collaborative Filtering Algorithm (20 points)

Please implement the item-based collaborative filtering algorithm based on adjusted cosine similarity.
In your report, include references to the most important code for this implementation.

4. Implement your own algorithm (15 points)

You can implement your own algorithm to improve your recommendation performance. The grading will be based on the performance (in terms of MAE) and novelty of the algorithm.
In your report, include references to the most important code for this implementation.
Please discuss the improvements you made and your reasoning.

5. Results Discussion (15 points)

Please provide the following information

Compare the accuracy of the various algorithms. Do you think your results are reasonable? How can you justify the results by analyzing the advantages and disadvantages of the algorithms?

Bonus:

Results Competition (20 points)

- 20 points will be assigned according to the performance of your recommendation system. These points are weighted according to this submission (points * weight of the assignment) and they will be added to this project and/or your lowest-scoring assignment/exam.
- I will also offer to take the highest-scorer to get lunch with me at Google (COVID/office restrictions permitting; very likely end of this year at the earliest). If the first person is not interested, I'll go down the ranked list of MAEs.

1: 20

2-3: 18

4-6: 16

7-10: 12

11-15: 10

16-18: 8

19-22: 6

23-25: 4

26+: 2

What to turn in

1. For the first deadline, you just need to submit your first run via the website: http://linux.students.engr.scu.edu/~psanchez/submit.html
   The grading only considers whether you submit or not, regardless of the performance of this submission. (Simply submit the format files if you must!)

2. For the second deadline, please submit a hard copy of the report of your experiments and your code to Camino.
   a. You will not be graded on the exact contents/structure/quality of your code, but it must be executable and properly implement the algorithm for each section:
      i. Basic User-Based CF (using pearson correlation) [40]
      ii. Item-Based (adjusted cosine similarity) [20]
      iii. Custom-Design [15]
   b. Please provide clear instructions for how to run your code in the different settings seen above.
      i. "set variable x=True and variables y,z=False to get Basic User-Based CF":
      ii. "execute my program as './movierecommender --mode=custom' to get my custom algorithm"
      iii. ..