Final Project

Yangye Zhu

1 Introduction

The *Pearson correlation coefficient* for users a and b [1] is

$$P_{a,b} = \frac{\sum\limits_{i=1}^{m}(r_{a,i} - \bar{r_a})(r_{b,i} - \bar{r_b})}{\sqrt{\sum\limits_{i=1}^{m}(r_{a,i} - \bar{r_a})^2 \sum\limits_{i=1}^{m}(r_{b,i} - \bar{r_b})^2}} \tag{1}$$

A test dataset is created as below (Table 1) to illustrate the computation of this coefficient. The threshold for number of common movies is set to be 6.

Table 1: Test Data

userID	**(movieID, rating)**
1	(1,1) (2,4) (3,3) (7,1) (8,3) (9,4) (10,5)
2	(1,5) (2,5) (3,2) (4,3) (5,4) (6,2)
3	(1,2) (2,4) (3,5) (4,5) (5,2) (6,3)

For userID 1, the number of common movies with the other two users is 3, less than 6, so it has no correlated user. For userID 2, the maximum coefficient in absolute value must be $P_{2,3}$, and we can use Equation 1 to calculate it.

$$\begin{aligned} P_{2,3} &= \frac{(5-3.5)(2-3.5)+(5-3.5)(4-3.5)+(2-3.5)(5-3.5)+(3-3.5)(5-3.5)+(4-3.5)(2-3.5)+(2-3.5)(3-3.5)}{\sqrt{((5-3.5)^2+(5-3.5)^2+(2-3.5)^2+(3-3.5)^2+(4-3.5)^2+(2-3.5)^2)^2}} \\ &= \frac{-4.5}{9.5} \\ &= -0.47 \end{aligned} \tag{2}$$

So the output file for this test dataset should look like:

```
1
2 (3, -0.47, 6)
3 (2, -0.47, 6)
```

To provide recommendations about movies that user a has not rated but is likely to enjoy, we calculate a *predicted rating* for movie i by user a according to the following expression [2].

$$p_{a,i} = \bar{r_a} + \frac{\sum\limits_{b=1}^{n}(r_{b,i} - \bar{r_b})P_{a,b}}{\sum\limits_{b=1}^{n} \mid P_{a,b} \mid} \tag{3}$$

2 Pseudocode

The algorithm for reading input file and computing the correlation coefficients is shown in Algorithm 1. Inside this algorithm, a function is called to return the Pearson correlation coefficient for two users, according to Equation 1. Similar algorithm can be applied to obtain the recommended movies according to Equation 3.

```
Data: A data file of movie ratings
Result: For each user the ncorrmax users who are most closely correlated
Initialization: a User class with data of a ratings map and a method of computing the PCC
correlation between a given user and itself;
Read the movie ratings data to populate the data of ratings map for each user;
for each user do
    for every other user do
        Call the method of User class to compute and store the data of correlations vector for
        each user
    end
end
```

Algorithm 1: Computation of the correlation coefficient

3 Comparison of Runtime

Table 2 compares the runtime of computing correlations using three kinds of programming language/style. Evidently, C++ is faster than Python, but in this case, C++ without OOP design is not significantly better in terms of performance. What really beats Python is C++ with OOP design. In addition, the algorithm is cleaner than that without OOP design. As seen in the algorithm section above, all we need to do is call the method of User class, making the code easy to read and understand. Another trivial observation is that if not using the -O3 option to compile the C++ code, it is actually much slower than Python. This might indicate that compiling optimization is crucial in achieving very good performance of C++ programming.

References

[1] CME211. Final project: Part 1. http://coursework.stanford.edu, November 15, 2013.

[2] CME211. Final project: Part 2. http://coursework.stanford.edu, November 24, 2013.

Table 2: Correlations Computation Runtime

Programming language/style	**Time cost (s)**
Python	33.3
C++ (not OOP)	14.9
C++ (OOP)	6.3