Programming Comps — January 16-19, 2007

1 Background

You graduated from Brown and got a job at FlatPix, a company that rents out computer games to their customers.[1] When returning a game, a customer must present his FlatPix-Card and is asked to rate the game between 1 (not so great) and 5 (spectacular). After two years, the company has a lot of data on which customer has rented what game and how (s)he liked it.

Now, FlatPix has decided to provide targeted advertisement, whereby the recommendation to rent a specific game shall be based on the ratings in the data base. The FlatPix informatics department was asked to provide an algorithm for this task.

2 Algorithm Description

The FlatPix algorithm design team has come up with the following algorithm to tackle the problem of forecasting the rating of a game by a customer. It is based on the idea that other users whose ratings of the jointly rented games correlate with yours allow us to infer a rating of a game that you do not know but that they have played and rated before. This is known in machine learning as "collaborative filtering".

2.1 Notation of Data

- For each customer c, your are given a list of previously rented games g with ratings $r_{cg} \in \{1, \ldots, 5\}$.
- For each customer c, denote by G_c the set of games that the user has rated, by $\oslash_c$ the average rating, and by σ_c the standard deviation of ratings.
- For each customer c and game $g \in G_c$, $\overline{r_{cg}}$ denotes the normalized rating.
- For each pair of customers $c \neq d$, denote by I_{cd} the set of games that both c and d have rated, and denote the correlation of their ratings by w_{cd}.
- For each customer c and game g, we denote the normalized forecast by $\overline{f_{cg}}$, and by $f_{cg} \in [1, 5]$ (a real number), we denote the forecasted rating.

[1] We invented this company of course. The data that you will work with as well as the idea for the assignment are based on the Netflix contest (see http://www.netflixprize.com/).

2.2 Normalization of Ratings

Naturally, different customers have their own idea of what a game rating 4 means. For some, a 4 is just an ok game, for others it may be the highest rating that they ever give. Also, some customers may like extreme ratings 1 and 5, whereas others rate everything between 2 and 4. Therefore, to make customer ratings comparable, your algorithm will first normalize the given ratings by computing $\oslash_c := \frac{1}{|G_c|}\sum_{g \in G_c} r_{cg}$, $\sigma_c := \sqrt{\frac{1}{|G_c|}\sum_{g \in G_c}(r_{cg} - \oslash_c)^2}$, and then setting

$$\overline{r_{cg}} := \frac{r_{cg} - \oslash_c}{\sigma_c}.$$

2.3 Customer Correlation

Next we compute a correlation factor between customers. For each pair of customers $c \neq d$, we set $I_{cd} := G_c \cap G_d$, and for all $c \neq d$ with $I_{cd} \neq \emptyset$, we set

$$w_{cd} := \frac{1}{|I_{cd}|} \sum_{g \in I_{cd}} \overline{r_{cg} r_{dg}}.$$

2.4 Rating of Games

Now, for each customer c and game $g \notin G_c$, we compute the normalized forecast

$$\overline{f_{cg}} := \frac{\sum_{d \mid I_{cd} \neq \emptyset,\ g \in G_d} w_{cd} \overline{r_{dg}}}{\sum_{d \mid I_{cd} \neq \emptyset,\ g \in G_d} |w_{cd}|}.$$

Finally, we project the normalized forecast rating back into the interval $[1, 5]$ by computing

$$f_{cg} := \overline{f_{cg}} \sigma_c + \oslash_c,$$

whereby we clip the interval so that all ratings below 1 are bumped up to 1, and all above 5 are set to 5. Note: this denormalization step should only be used to forecast a rating, but not to predict the n best games to recommend to a customer.

3 Your Task

It is your task to **implement this algorithm**. Do not make up your own approach to tackle the problem! If you do, you will fail the exam. The choice of algorithm is fixed, now all that is required is that you implement it. The following features must be supported by your implementation:

- Hand in **one** file named procom07.tar.gz containing all files of your code, a file named Readme.txt explaining in detail how to compile it, as well as a file named Description.pdf that describes your implementation in detail, including a justification for your choice of language and the structure of your code. **Do not include the test data in your tar.gz!**

- Your code must compile and run on standard department linux machines with 2GB main memory. If your program does not compile or run on a machine with those specifications, you will not pass the exam. When started, your executable will be provided with a command line argument containing the path name of the rating data files which are in the format as described in the Readme at "/u/sello/pub/FlatPix/PROCOM-07" and a parameter K (see below). Note: When parsing the input **you must follow the format instructions** as you find them in the said Readme file! Do not assume anything else that might only hold for the concrete input data that we provide. Also, take into account that the data with which the program is provided might not be in the specified format, in which case we want your code to stop with an appropriate **error message that makes it easy for the user** to find the problem in the data.

- Your executable should prompt the user with two options: Forecast the rating of game g by customer c (g, c as given by the user). This forecast is given by f_{cg} that should be computed **as needed**. Games for which the customer has provided a rating should be rated as such. The second option for the user is to ask for the n most interesting, unrented games of customer c (n, c as given by the user). This recommendation should be based on values $\overline{f_{cg}}$ which are also to be reported together with the corresponding game IDs. For each request, report the CPU-time of your program. Repeated requests of these two types should be possible, until the user enters "quit" which terminates the program. Make the user interface self-explanatory so that it is easy to use your program. Prompt the user with a hint what is going on when conducting lengthy computations during which your program may not respond.

- To save time when a request is made, your program should first compute and store data G_c, $\oslash_c$, σ_c, $\overline{r_{cg}}$, and r_{cg}. Also, initialize your data structures so that future computations can be carried out efficiently. All other data should **only be computed as needed**, whereby data that has been computed once should not be recomputed for at least the next K requests.

- Use efficient data structures to store and access the data. Save memory and time! In your Description.pdf, explain and justify the data structures you decided to use.

4 Evaluation Criteria

Your implementation will be evaluated based on **all** of the following criteria:

- Appropriate choice of language. Consider that FlatPix wants to use your program to send advertisement to almost half a million customers! Therefore, you must choose a programming language that is feasible for the task. The primary focus on ease of programming and maintainability is only okay within reasonable performance limits.

- Structure, readability, and maintainability of your code.
- Correctness, memory efficiency, and speed.

It is important that your work passes a critical threshold with respect to all criteria. A super-efficient code that is poorly documented and hardly maintainable will be regarded as a failure. So will a software engineering masterpiece that does not work as specified or that does not respond in reasonable time. Defintely do not code in assembler or shell scripts! And now: **Good luck!**