Aviv Glick
Cosi 105b

February 9, 2015
Professor Salas

Github repo: https://github.com/gaviv19/PA2
CodeClimate report: https://codeclimate.com/github/gaviv19/movies-2

Algorithm

I used the root mean square of the error as my primary estimator, to identify my expected prediction error. I have tried many different combinations of independent variables in my algorithm, such as the average rating of a movie, a popularity of a movie (an algorithm I implemented in the class Movies1), the average of ratings user u gives to all users, and even by using the rating given to movie m by another user, who, by using the most_similar(u) method in class Movies1, was the user with most similar taste in movies to user u. I combined all the variables and checked my results by trying several algorithms, which include different number of variables at a time with different variations of calculations, factors, and averages, trying to minimize the rms. I was surprised that basically all variables but one were harming my result, and the smallest rms was achieved by using mainly the popularity value, according to the popularity method I developed in Movies1. I further minimized rms by adding a fraction of the average rating the user gave to all movies, but the addition is negligible. All in all, my algorithm is very (very) simple:

Predicted rating = popularity of movie m + 0.01*(ave. ratings by user u)

The popularity of a movie is as follows: (a + 2^r) - 1, where:
a = the movie's average rating
r = the ratio of the number of users who have rated the movie to the total number of users (always between 0 - 1).

Thus, the popularity scale is 1 to 5.

In addition, I was extremely surprised to discover that the rating given to m by the user most similar to u did not provide a good prediction: After further enhancing my algorithm by checking the number of movies user u and his most similar user both rated (to verify they have rated “many” movies together, thus confirming the validity of this factor as prediction estimator), I was not surprised to discover that on average, most similar users (according to my most_similar(u) algorithm), did not share many rated movies, making its value useless. I then understood that I might see different results by changing the most_similar(u) algorithm (changing the definition of “most similar users”), and then I might be able to further improve my predicted ratings.

Analysis

For k = 20000, you get:
average prediction error 0.0876887279456622
root mean square error of the prediction: 0.9785517316470921
standard deviation of the error: 3.592491194392531

Aviv Glick
Cosi 105b

February 9, 2015
Professor Salas

Using the Time.now command, the whole process takes:

2.87434 seconds	for	k=20,000
1.406443 seconds	for	k=10,000
0.749403 seconds	for	k=5,000

Looks like my algorithm's time complexity is linear.