Systems for Data Science : Milestone 2

Aiday Marlen Kyzy
Sciper : 283505

April 2021

1 Introduction

In this project I have worked on Milestone 2 for the course Systems for Data Science. In the table below you can find the approximate running times for the 3 scala files. The times are rounded up to the nearest minute, in practise the running times are slightly different.

similarity.Predictor.scala	6 mins
knn.Predictor.scala	4 mins
recommender.Recommender.scala	3 mins

2 Similarity-Based Predictions

[Question 2.3.1]
The first question was to compute the prediction accuracy using the improved baseline method introduced in the project. I did not use the pseudo-code suggested in the project. The cosine-based MAE found is : 0.5927119957743079. The difference between the cosine-based MAE and the baseline MAE given in the problem is : -0.17418800422569214. Since the difference is negative this means that the cosine-based prediction has a better prediction accuracy than the baseline which is given.

[Question 2.3.2]
The Jaccard coefficient between a set user u and a user v can be defined as follows. Here the sets I_u and I_v indicate the set of movies seen by user u and v.

$$J(I_u, I_v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \tag{1}$$

The Jaccard MAE is : 0.47298273387222567. The difference between the Jaccard and the cosine-based MAE is : -0.11972926190208222. Since this difference is negative this means the Jaccard similarity provides a slightly better prediction accuracy than the cosine-based similarity.

[Question 2.3.3]
Suppose that we calculate one similarity for each pair of users u, v. Then theoretically we would compute nCr(943, 2) = 444153 similarities on the ml-100k data set since there are 943 users. Due to my specific implementation of the cosine based-similarity however, I actually compute the similarities for all $(user_1, user_2)$, and $(user_2, user_1)$ pairs where $user_1$ and $user_2$ have seen at least one movie in common. The reason I compute the similarities in both orders is that this is later simplifies the calculations of the user specific weighted sum deviations. Hence the number of similarities I actually compute is : 141183. In the case of Jaccard similarities, since I use a cartesian join, I calculate 444153 similarities.

[Question 2.3.4]
In this problem, I have calculated for all user pairs u, v the number of movies rated in common. I have found the following statistics :

Statistic	Value
Maximum	263
Minimum	0
Mean	3
Standard Deviation	32

From the table above we can see that the maximum number of movies rated in common is 263.

[Question 2.3.5]
Theoretically, if we are to take into account all pairs of $s_{u,v}$ values without distinction on the order of u and v, then the total memory required to store the similarities will be : $444153 \cdot 8 = 3553224$ bytes. In general, if the number of users is U, then the memory requirement will be : nCr(U,2)·8.

In practise however, as explained before, due to my implementation of the algorithm, I had to calculate the similarities $s_{u,v}$ and $s_{v,u}$ for both orders of users u and v, where u and v have seen at least one movie in common. As such I take therefore the total number of similarities $s_{u,v}$ and I multiply that by 8 bytes, the number of bytes in a double, to get that the memory requirement for all the similarities is roughly $141183 \cdot 8 = 1129464$ bytes.

[Question 2.3.6]
In this question, I analyze the time to compute the prediction with the code I implemented. Meaning that here I don't calculate all possible $s_{u,v}$ but only the similarities for users having rated at least one movie in common, where I also make a distinction between $s_{u,v}$ and $s_{v,u}$. I perform 5 iterations as was asked in the project.

Statistic	Value
Maximum (ms)	434772.7
Minimum (ms)	117724.0
Mean (ms)	200401.48
Standard Deviation (ms)	117921.19

In the milestone 1, I reported an average of 3853401 microseconds for computing the baseline MAE. I can **not** compare the results above with the time in the first milestone, because here above I report the time just for computing the RDD of predictions, **not** for computing the final MAE, for which you have an additional map, reduce and count operations. However for the purpose of this question, I decided to also find the total time to compute the MAE, and I found a mean of around 20'000'000 microseconds. Therefore we can say that to compute the MAE in milestone 2 takes about 6 times more time than in milestone 1. This is to be expected as the baseline used in the second milestone is more complicated than in the first milestone and requires more calculations and more computation time.

[Question 2.3.7]
This time we consider the time required for computing only the similarities. We find the following statistics for 5 iterations :

Statistic	Time in microseconds
Maximum (ms)	341601.8
Minimum (ms)	92770.8
Mean (ms)	152033.08
Standard Deviation (ms)	95307.15
Average per $s_{u,v}$ (ms)	0.342
TimeSimilarity/TimePredict	0.758

From the table above we find as expected that the maximum, minimum, average and standard deviation are lower than in the table of question 2.3.6. The average time in microseconds per similarity calculation is 0.342, and the ratio of the time needed to compute the similarities versus the time needed to compute the predictions is 0.758. From this we can infer that the bulk of the computation time is concentrated in the similarity computations, as is expected.

3 Neighborhood-Based Predictions

[Question 3.2.1]
We find the following MAE values for the different values of k. From the table below we see that as the value of k increases, the MAE decreases. We find that for k equal to 800 and 943, the MAE values are equal, meaning that there is no change in the predictions past $k = 800$. Therefore to have a higher prediction accuracy, you should use a higher k value.

Value of k	MAE
10	0.6688162132564153
30	0.6626217380009035
50	0.648568699984258
100	0.6414906957479878
200	0.6451284761773858
400	0.5985431228947352
800	0.5927119957743079
943	0.5927119957743079

The lowest k with better baseline than the proposed MAE is for $k = 1$. Meaning that just by considering the first nearest neighbor in the prediction, we already get a lower MAE than 0.7669. The corresponding difference is : -0.19335007901692902.

[Question 3.2.2]
Suppose that we choose only the k nearest-neighbors for each user u. Then we consider $k \cdot U$ similarities, but since $s_{u,v} = s_{v,u}$ in an optimal scenario, then the minimum number of bytes required is : $\frac{1}{2} \cdot k \cdot U \cdot 8$. In practise however since I calculate both $s_{u,v}$ and $s_{v,u}$, I find the following memory requirements.

Value of k	Actual number of bytes
10	36688
30	109640
50	182080
100	361976
200	702232
400	1114336
800	1125792
943	1125792

As expected, when k increases, the memory required increases.

[Question 3.2.3]
Since $k = 1$, this means, that for each user u I store only one similarity $s_{u,v}$. According to the problem formulation, I store each similarity along with the values of u and v. Hence each such row is worth 24 bytes. On my computer I have 16.0 GB installed RAM, which is $16 \cdot 2^{30} = 17179869184$. Hence I can store 715827882 rows of the form $(u, v, s_{u,v})$.

[Question 3.2.4]
We investigate how varying k has an impact on the number of similarity values $s_{u,v}$ to compute. We find the following data where we set $k = 10, 30, 50, 100, 200, 400, 800, 943$.

Value of k	Number of $s_{u,v}$
10	4586
30	13705
50	22760
100	45247
200	87779
400	139292
800	140724
943	140724

As expected, as the value of k increases, the number of similarities computed increases, and one needs more storage space to store these similarities. In fact we notice that until $k = 200$ the increase in the number of similarities is roughly linear, and then it evens out. Past a value of 800, the number of computed similarities does not change.

[Question 3.2.5]
In this problem I use the Milestone 1 personal-data.csv to which I added additional ratings, because I did not have a sufficient number of ratings before. I get the following recommendations for k=30.

Id of movie	Movie title	Predicted rating
804	Jimmy Hollywood (1994)	3.7555
801	Air Up There, The (1994)	3.5268
797	Timecop (1994)	3.4786
803	Heaven & Earth (1993)	2.1948
799	Boys Life (1995)	2.6356

I get the following recommendations in the case $k = 300$.

Id of movie	Movie title	Predicted rating
804	Jimmy Hollywood (1994)	3.7026
797	Timecop (1994)	3.3850
801	Air Up There, The (1994)	3.2267
803	Heaven & Earth (1993)	2.5984
799	Boys Life (1995)	2.3249

As expected we get the same top 5 recommended movies in both cases albeit in a different order, even though the predicted ratings are slightly different. By looking at the top 3 recommended movies, in the case $k = 300$, the predicted ratings are lower than for $k = 30$. We observe the opposite behavior for the last 2 recommended movies. From this observation, we are not able to infer a generalization which would explain how the predictions vary as k increases.

Since I have used the personal-data.csv file from Milestone 1, to which I added more ratings, I can not compare the recommended movies in Milestone 1 and Milestone 2 effectively. However I still notice that in Milestone 1 my top 5 recommendations were movies with ID 50, 100, 174, 181, 258, and these are far from the recommendations I obtain above. In fact, the above recommendations are all surprisingly close in terms of movie ID. This actually makes sense since the users surveyed to create the database, probably gave ratings sequentially to movies close together in terms of movie ID. Therefore what likely happened is that my ratings have likely coincided with the ratings of a user having rated movies in the range [780, 810], and I am getting as recommendations, the movies highly rated by that user.

I also notice that the predicted ratings are fractional values no bigger than 4.0. This is different to the predictions in Milestone 1, where I got for the first 5 recommended movies a predicted rating of 5.0. This is probably the case because the predictions take into account the similarities with the k nearest neighbors which are fractional values, and therefore the predictions are themselves also fractional. These similarity values are also probably small values (smaller than 4.0) which cause the predicted ratings to fall below 4.0.

It remains surprising that my recommendations in both cases vary so much. From this I infer that including similarity coefficients in the predicted ratings, changed greatly the recommended movies. Since the difference between the cosine-based MAE and 0.7669, the base MAE given, is negative, we can also infer that the cosine-based predictions give better recommendations nonetheless than the prediction method from Milestone 1.

4 Conclusion

This second milestone was an interesting project. In the beginning I implemented the suggested pseudo-code, and my code didn't run. By changing and personalizing the code, I was able to make it output results.