CS-449 Project Milestone 1: Personalized Recommender with k-NN

Name: Adda Julien
Sciper: 282 449
Email: julien.adda@epfl.ch
Name: Epple Pascal
Sciper: 287 340
Email: pascal.epple@epfl.ch

March 25, 2022

For reading comprehension, most of the presented computations were rounded down to four digits precision. The complete results can be found in the appropriately generated .json *files.*

1 Motivation: Movie Recommender

(No Q)

2 Proxy Problem: Predicting Ratings

(No Q)

3 Baseline: Prediction based on Global Average Deviation

3.1 Answers

B.1 The computed values are reported in the following table:

Table 1:

$\bar{r}_{\bullet,\bullet}$	3.5264625
$\bar{r}_{1,\bullet}$	3.6330
$\bar{r}_{\bullet,1}$	3.8882
$\hat{\bar{r}}_{\bullet,1}$	0.3027
$p_{1,1}$	4.04681

B.2 The computed results are the following:

Table 2:

Prediction method	Average MAE)
GlobalAvgMAE ($\bar{r}_{\bullet,\bullet}$)	0.9489
UserAvgMAE ($\bar{r}_{u,\bullet}$)	0.8383
ItemAvgMAE ($\bar{r}_{\bullet,i}$)	0.8206
BaselineMAE ($p_{u,i}$)	0.7604

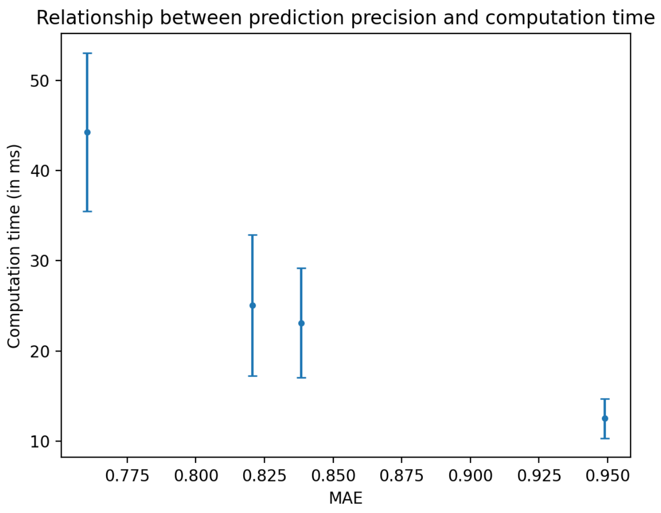
B.3 We thrice measured the time needed in order to compute the average MAEs and got the following results:

Table 3:

Prediction method	average time (in ms)	standard deviation (in ms)
$\bar{r}_{\bullet,\bullet}$	12.5147	2.2027
$\bar{r}_{u,\bullet}$	23.1030	6.0636
$\bar{r}_{\bullet,i}$	25.0750	7.8336
$p_{u,i}$	44.2814	8.7918

The baseline prediction method is definitely the most time consuming. This is not a surprise as it is the method out of the four involving the most computations. The price we pay in computation time is directly compensated by a significantly better average MAE, as can be seen in Table 2.

Figure 1:



We notice that there seems to be a quadratic relationship between prediction precision and computation time. We can conclude that more accurate prediction methods definitely need more computation time. As the relationship is quadratic, we can clearly see the need to make the computations in an as efficient as possible way, and the use of Spark and MapReduce comes in handy to do so.

Technical specifications:

- Model : MacBook Pro (2019)
- CPU Speed : 2.4 GHz
- RAM : 16 Go
- OS : macOS Monterey
- Scala language version : 2.13.8
- JVM version : 1.8.0_272

4 Spark Distribution Overhead

4.1 Answers

D.1 Running the code locally, the following values were computed:

Table 4:

$\bar{r}_{\bullet,\bullet}$	3.5264
$\bar{r}_{1,\bullet}$	3.6330
$\bar{r}_{\bullet,1}$	3.8882
$\bar{\hat{r}}_{\bullet,1}$	0.3027
$p_{1,1}$	4.0468

The prediction accuracy of the proposed baseline is the following:

$$AvgMAE = 0.7604$$

As expected, we get the same results as for questions B1 and B2. This ensures our implementation using Spark RDDs is working properly.

D.2 We encountered some issues when downloading our files onto the cluster and therefore can't answer this question.

5 ***Personalized*** Predictions

5.1 Answers

P.1 Using uniform similarities, the following values were computed:

Table 5:

$p_{1,1}$	4.0468
MAE	0.7604

These results are similar to the results in Tables 1 and 2, which is not a surprise as using uniform similarities boils down to the baseline prediction method.

P.2 Using the adjusted cosine similarity, we computed the following values:

Table 6:

$s_{1,2}$	0.0730
$p_{1,1}$	4.0870
MAE	0.7372

P.3 The Jaccard Coefficient is a similarity method based on the common rated movies between two users. The mathematical formulation is the following:

$$s_{u,v} = \begin{cases} \frac{|Items(u) \cap Items(v)|}{|Items(u) \cup Items(v)|} & Items(i) \cup Items(j) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

where $Items(u)$ denotes the items user u rated.

Notice that this metric takes values in $[0, 1]$, is symmetric and does not take the actual ratings both users gave to the movies into account.
With this new metric implemented, we computed the following values:

Table 7:

$s_{1,2}$	0.0318
$p_{1,1}$	4.0982
MAE	0.7556

6 Neighbourhood-Based Predictions

6.1 Answers

N.1 These are the values we computed using $k = 10$:

Table 8:

$s_{1,1}$	0
$s_{1,864}$	0.2423
$s_{1,886}$	0
$p_{1,1}$	4.3190
MAE	0.8287

N.2 The following table contains the computed values:

Table 9:

k	10	30	50	100	200	300	400	800	943
MAE	0.8287	0.7810	0.7618	0.7466	0.7400	0.73915	0.73912	0.7392	0.7372

As a reminder, for the baseline predictor we managed to get an average MAE of 0.7604. From that, we conclude that $k = 100$ is the lowest k such that the MAE of the k-NN method is lower than for the baseline method.

N.3 Several techniques were implemented to decrease our mean time :

– replace map by fold left when computing the mean

– use closures in functions as much as possible (calculate in advance values)

– take advantage of the symmetry of the similarity function. Rather than computing every similarity(u,v), we compute half of the values and set similarity(u,v) = similarity(v,u)

– not compute the similarity (u,v) when u == v as we know that this value is equal to 0.0.

- when finding the k biggest similarities per user, rather than sorting the array, reversing it and taking the k first elements, we sort and drop the length_of_array-k first elements (we don't need to reverse).
- for k similarity : store our pre-computed values in Maps rather than arrays of arrays. This prevents us from storing the similarity(u,v) that are equal to 0.0 (all the similarity that were not in the top k biggest). By using the getOrElse method we can return 0.0, if the key (u,v) is not in the Map.

For k = 300 and with 5 measurements, our mean time on the data set `data/ml-100k/u2.test` is $27208.6293ms$ and the std is $1427.7720ms$. Our first value for timing was several hours, and it gave the same output as now.

7 Recommendation

7.1 Answers

R.1 The following value was computed:

$$p_{1,1} = 4.1322$$

R.2 The top 3 recommendations for user "944" are:

"Maya Lin: A Strong Clear Vision (1994)"
"Great Day in Harlem"
"Prefontaine (1997)"