CS-449 Project Milestone 2: Optimizing, Scaling, and Economics

Name: Emna FENDRI
Sciper: 297286
Email: emna.fendri@epfl.ch
Name: Douglas Bouchet
Sciper: 287906
Email: douglas.bouchet@epfl.ch

May 19, 2022

The technical specifications of the machine on which we ran the tests for this project.

- Model: MacBook Pro
- CPU speed: 2,3 GHz Intel Core i5 quad-core processor
- RAM: 8 Go 2133 MHz LPDDR3
- OS: macOS Big Sur Version 11.6
- Scala language version: 2.11.12
- JVM version: 1.8.0
- sbt version: 1.4.7

General Notes:

- In this context, as we do not include self-similarity in the k- nearest neighbours we choose to set all the self-similarities to 0.
- We approximate results to 10^{-5}.

1 Optimizing with Breeze, a Linear Algebra Library

BR.1 Reimplement the kNN predictor of Milestone 1 using the Breeze library and without using Spark. Using $k = 10$ and `data/ml-100k/u2.base` for training, output the similarities between: (1) user 1 and itself; (2) user

1 and user 864; (3) user 1 and user 886. Still using $k = 10$, output the prediction for user 1 and item 1 ($p_{1,1}$), the prediction for user 327 and item 2 ($p_{327,2}$), and make sure that you obtain an MAE of 0.8287 ± 0.0001 on `data/ml-100k/u2.test`.

Answer:

Using k = 10 on `ml-100k/u2.base` for training and `ml-100k/u2.test` for testing:

- similarity(1,1) = 0
- similarity(1,864) = 0.24232
- similarity(1,886) = 0
- prediction user 1 item 1: 4.31909
- prediction user 327 item 2: 2.69941
- MAE: 0.82872

BR.2 Try making your implementation as fast as possible, both for computing all k-nearest neighbours and for computing the predictions and MAE on a test set. Your implementation should be based around `CSCMatrix`, but may involve conversions for individual operations. We will test your implementation on a secret test set. The teams with both a correct answer and the shortest time will receive more points.

Using $k = 300$, compare the time for predicting all values and computing the MAE of `ml-100k/u2.test` to the one you obtained in Milestone 1. What is the speedup of your new implementation (as a ratio of $\frac{average\ time_{old}}{average\ time_{new}}$)? Use the same machine to measure the time for both versions and provide the answer in your report.

Also ensure your implementation works with `data/ml-1m/rb.train` and `data/ml-1m/rb.test` since you will reuse it in the next questions.

Answer:

Using $k = 300$ on `ml-100k/u2.base` for training and `ml-100k/u2.test` for testing :

- Timing from Milestone 1 :
  - Average timing: 21.686 seconds
  - stddev: 7.794 seconds
- Timing from Milestone 2 :
  - Average timing: 7.519 seconds
  - stddev: 0.433 seconds

Finally, this represents a **speedup** of $\frac{21.686}{7.519} = 2.884$.
Note:

We were having a java heap error during the similarity matrix computation when training on the `data/ml-1m/rb.train` data set. So we had to replace the standard breeze matrix multiplication by a different (and slower) implementation. Before making this change, we were having an average run time of 4.9 seconds on the `ml-100k/u2.base` data set, which induced a speed up of $\frac{21.686}{7.519} \sim 4.5$

2 Parallel k-NN Computations with Replicated Ratings

EK.1 Test your parallel implementation of k-NN for correctness with two workers. Using $k = 10$ and `data/ml-100k/u2.base` for training, output the similarities between: (1) user 1 and itself; (2) user 1 and user 864; (3) user 1 and user 886. Still using $k = 10$, output the prediction for user 1 and item 1 ($p_{1,1}$), the prediction for user 327 and item 2 ($p_{327,2}$), and make sure that you obtain an MAE of 0.8287 ± 0.0001 on `data/ml-100k/u2.test`

Answer:

Using k = 10 on `ml-100k/u2.base` for training and `ml-100k/u2.test` for testing:

- similarity(1,1) = 0
- similarity(1,864) = 0.24232
- similarity(1,886) = 0
- prediction user 1 item 1: 4.31909
- prediction user 327 item 2: 2.69941
- MAE: 0.82872

EK.2 Measure and report the combined *k-NN* and *prediction* time when using 1, 2, 4 workers, $k = 300$, and `ml-1m/rb.train` for training and `ml-1m/rb.test` for test, on the cluster (or a machine with at least 4 physical cores). Perform 3 measurements for each experiment and report the average and standard-deviation total time, including training, making predictions, and computing the MAE. Do you observe a speedup? Does this speedup grow linearly with the number of executors, i.e. is the running time X times faster when using X executors compared to using a single executor? Answer both questions in your report.

Answer:

By running `distributed.exact` on the cluster with `data/ml-1m/rb.train` for training and `data/ml-1m/rb.test` for test , with k = 300 and computing the mean and standard deviation over 3 runs, we obtained the following results on Table 1:

Number of workers	Average (seconds)	stddev
1	**278**	**0.588**
2	**172**	**1.938**
4	**120**	**2.134**

Table 1: Total timings including training, making predictions and computing MAE of Parallel(exact) 300-NN on `1m` data set.

We can clearly observe a speedup as we use more workers.

From these timings, we can see that using 2 executors is ~ 1.61 times faster than 1 executor, and using 4 executors is ~ 1.43 times faster than 2 executors or ~ 2.32 times faster than 1 executor.

This is not really linear in the number of executor, as we cannot say that using 4 executors is 2 times faster than 2 executors or 4 times faster than 1 executor. However we can see that we tend to a linearity in the number of executor, but with a constant of 1.52. i.e using 2N executors is ~ 1.52 times faster than using N executors.

The fact that it's not perfectly linear can be explained by the fact that not all the computations are parallelized, i.e only the similarity computation is. This bound the speed up of code execution to only a part of it.

3 Distributed Approximate k-NN

AK.1 Implement the approximate k-NN using your previous breeze implementation and Spark's RDDs. Using the partitioner of the template with 10 partitions and 2 replications, $k = 10$, and `data/ml-100k/u2.base` for training, output the similarities of the approximate k-NN between user 1 and the following users: $1, 864, 344, 16, 334, 2$.

Answer: Using k = 10 on `ml-100k/u2.base`, with 10 partitions and 2 replications:

- similarity(1,1) = 0
- similarity(1,864) = 0
- similarity(1,344) = 0.23659
- similarity(1,16) = 0
- similarity(1,334) = 0.19282
- similarity(1,2) = 0

AK.2 Vary the number of partitions in which a given user appears. For the `data/ml-100k/u2.base` training set, partitioned equally between 10 workers, report the relationship between the level of replication (1,2,3,4,6,8) and the MAE you obtain on the `data/ml-100k/u2.test` test set. What is

the minimum level of replication such that the MAE is still lower than the baseline predictor of Milestone 1 (MAE of 0.7604), when using $k = 300$? Does this reduce the number of similarity computations compared to an exact k-NN? What is the ratio? Answer both questions in your report.

Answer: The results are summarized in the table 2. We set k = 300, trained on `ml-100k/u2.base` and the MAE is obtained on `ml-100k/u2.test`. We run this locally on our device using 4 workers.

Level of replication	MAE	Number of similarity computations	Ratio
1	0.8090	89 845	0.101
2	**0.7587**	**356 924**	**0.401**
3	0.7457	801 563	0.901
4	0.7410	1 425 050	1.602
6	0.7391	3 202 522	3.601
8	0.7391	5 694 792	6.404

Table 2: MAE and ratio of number of similarities computed compared to an exact 300-NN, on the `100k` data set.

From the table, we see that with 2 replications we get an MAE of $0,7587$. Therefore, we need to have at least **2 replications** such that the MAE is still lower than the baseline predictor of Milestone 1 (MAE of $0,7604$), when using k = 300.

We also report in the table the **total number of similarity computations needed for each level of replication**. We compute this as follows:
Assume there are p partitions and let n_i be the number of users assigned to partition i. Then the number of similarity computations in partition i is $(n_i)^2$, this is because we compute the similarity for each ordered pair of users that are assigned to partition i. Hence, the total number of computations T is given by

$$T = \sum_{i=1}^{p} (n_i)^2.$$

Following the same reasoning, the number of similarity computations in an exact k-NN is N^2 where N is the total number of users.

The **ratio** is then computed as : $\frac{T}{N^2} = \frac{T}{889249}$, where $N = 943$ when training on `ml-100k/u2.base`.

With **2 replications** we indeed **reduce** the number of similarity computations compared to an exact k-NN with a factor of roughly $\frac{889249}{356924} = 2,491$.

AK.3 Measure and report the time required by your approximate *k-NN* implementation, including both training on `data/ml-1m/rb.train` and computing the MAE on the test set `data/ml-1m/rb.test`, using $k = 300$ on 8 partitions with a replication factor of 1 when using 1, 2, 4 workers. Perform each experiment 3 times and report the average and standard-deviation. Do you observe a speedup compared to the parallel (exact) k-NN with replicated ratings for the same number of workers?

Answer: The results of the measurements are reported on Table 3.

Number of workers	Average (seconds)	stddev
1	164.586	5.055
2	135.108	5.787
4	132.507	3.535

Table 3: Total timings including training, making predictions and computing MAE of Approximate 300-NN on `1m` data set (8 partitions, replication factor of 1).

We can see that for 1 worker, the improvement of approximate k-NN ($164sec$) compared to exact k-NN ($278sec$) is significant ($\sim \frac{278}{164} = 1.69$ times faster).
However, whereas in the exact case the runtime was $\sim$ linearly decreasing with the number of workers, we can see here that it's clearly not the case. Using 2 workers instead of 1 only speeds up the computation by $\sim 20\%$. And using 4 workers instead of 2 brings almost not improvement. In the end, with 4 workers, the exact k-NN ($120sec$) is actually slightly faster than the approximate k-NN ($132sec$).

4 Economics

Implement the computations for the different answers in the Economics.scala file. You don't need to provide unit tests for this question, nor written answers for these questions in your report.