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

Name: xxx
Sciper: xxx
Email: xxx
Name: xxx
Sciper: xxx
Email: xxx

February 28, 2022

1 Motivation: Movie Recommender

(No Q)

2 Proxy Problem: Predicting Ratings

(No Q)

3 Baseline: Prediction based on Global Average Deviation

3.1 Questions

Implement the previous prediction methods using Scala's standard library, without using Spark.

$$p_{u,i} = \bar{r}_{u,\bullet} + \bar{\hat{r}}_{\bullet,i} * scale((\bar{r}_{u,\bullet} + \bar{\hat{r}}_{\bullet,i}), \bar{r}_{u,\bullet}) \tag{1}$$

B.1 *Compute and output the global average rating ($\bar{r}_{\bullet,\bullet}$), the average rating for user 1 ($\bar{r}_{1,\bullet}$), the average rating for item 1 ($\bar{r}_{\bullet,1}$), the average deviation for item 1 ($\bar{\hat{r}}_{\bullet,1}$), and the predicted rating of user 1 for item 1 ($p_{1,1}$, Eq* **??***) using `data/ml-100k/u2.base` for training. When computing the item average for items that do not have ratings in the training set, use the global average ($\bar{r}_{\bullet,\bullet}$). When making predictions for items that are not in the training set, use the user average if present, otherwise the global average.*

B.2 *Compute the prediction accuracy (average MAE on `ml-100k/u2.test`) of the previous methods ($\bar{r}_{\bullet,\bullet}$, $\bar{r}_{u,\bullet}$, $\bar{r}_{\bullet,i}$) and that of the proposed baseline ($p_{u,i}$, Eq. **??**).*

B.3 *Measure the time required for computing the MAE for all ratings in the test set (`ml-100k/u2.test`) with all four methods by recording the current time before and after (ex: with `System.nanoTime()` in Scala). The duration is the difference between the two.*

Include the time for computing all values required to obtain the answer from the input dataset provided in the template: recompute from scratch all necessary values even if they are available after computing previous results (ex: global average $\bar{r}_{\bullet,\bullet}$). Also ensure you store the results in some auxiliary data structure (ex: `Seq[(mae, timing)]`) as you are performing measurements to ensure the compiler won't optimize away the computations that would otherwise be unnecessary.

For all four methods, perform three measurements and compute the average and standard-deviation.

In your report, show in a figure the relationship between prediction precision (MAE) on the x axis, and the computation time on the y axis including the standard-deviation. Report also the technical specifications (model, CPU speed, RAM, OS, Scala language version, and JVM version) of the machine on which you ran the tests. Which of the four prediction methods is the most expensive to compute? Is the relationship between MAE and computation linear? What do you conclude on the computing needs of more accurate prediction methods?

4 Spark Distribution Overhead

4.1 Questions

Implement $p_{u,i}$ using Spark RDDs. Your distributed implementation should give the same results as your previous implementation using Scala's standard library. Once your implementation works well with `data/ml-100k/u2.base` and `data/ml-100k/u2.test`, stress test its performance with the bigger `data/ml-25m/r2.train` and `data/ml-25m/r2.test`.

D.1 *Ensure the results of your distributed implementation are consistent with **B.1** and **B.2** on `data/ml-100k/u2.train` and `data/ml-100k/u2.test`. Compute and output the global average rating ($\bar{r}_{\bullet,\bullet}$), the average rating for user 1 ($\bar{r}_{1,\bullet}$), the average rating for item 1 ($\bar{r}_{\bullet,1}$), the average deviation for item 1 ($\bar{\hat{r}}_{\bullet,1}$), and the predicted rating of user 1 for item 1 ($p_{1,1}$, Eq **??**). Compute the prediction accuracy (average MAE on `ml-100k/u2.test`) of the proposed baseline ($p_{u,i}$, Eq. **??**).*

D.2 *Measure the combined time to (1) pre-compute the required baseline values for predictions and (2) to predict all values of the test set on the 25M* *dataset, `data/ml-25m/r2.train` and `data/ml-25m/r2.test`. Compare the time required by your implementation using Scala's standard library (**B.1** and **B.2**) on your machine, and your new distributed implementation using Spark on `iccluster028`. Use 1 and 4 executors for Spark and repeat all three experiments (predict.Baseline, distributed.Baseline 1 worker, distributed.Baseline 4 workers) 3 times. Write in your report the average and standard deviation for all three experiments, as well as the specification of the machine on which you ran the tests (similar to B.3).*

As a comparison, our reference implementation runs in 44s on the cluster with 4 workers. Ensure you obtain results roughly in the same ballpark or faster. Don't worry if your code is slower during some executions because the cluster is busy.

Try optimizing your local Scala implementation by avoiding temporary objects, instead preferring the use of mutable collations and data structures. Can you make it faster, running locally on your machine without Spark, than on the cluster with 4 workers? Explain the changes you have made to make your code faster in your report.

5 *Personalized* Predictions

5.1 Questions

$$s_{u,v} = \begin{cases} \frac{\sum_{i \in (I(u) \cap I(v))} \hat{r}_{u,i} * \hat{r}_{v,i}}{\sqrt{\sum_{i \in I(u)} (\hat{r}_{u,i})^2} * \sqrt{\sum_{i \in I(v)} (\hat{r}_{v,i})^2}} & (I(u) \cup I(v)) \neq \emptyset; \exists_{i \in I(u)} \hat{r}_{u,i} \neq 0; \exists_{i \in I(v)} \hat{r}_{v,i} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$p_{u,i} = \bar{r}_{u,\bullet} + \bar{\hat{r}}_{\bullet,i}(u) * scale((\bar{r}_{u,\bullet} + \bar{\hat{r}}_{\bullet,i}(u)), \bar{r}_{u,\bullet}) \tag{3}$$

P.1 *Using uniform similarities of 1 between all users, compute the predicted rating of user 1 for item 1 ($p_{1,1}$) and the prediction accuracy (MAE on `ml-100k/u2.test`) of the personalized baseline predictor.*

P.2 *Using the the adjusted cosine similarity (Eq. **??**), compute the similarity between user 1 and user 2 ($s_{1,2}$), the predicted rating of user 1 for item 1 ($p_{1,1}$ Eq. **??**) and the prediction accuracy (MAE on `ml-100k/u2.test`) of the personalized baseline predictor.*

P.3 *Implement the Jaccard Coefficient[1]. Provide the mathematical formulation of your similarity metric in your report. User the jaccard similarity, compute the similarity between user 1 and user 2 ($s_{1,2}$), the predicted rating of user 1 for item 1 ($p_{1,1}$ Eq. **??**) and the prediction accuracy (MAE on `ml-100k/u2.test`) of the personalized baseline predictor. Is the Jaccard Coefficient better or worst than Adjusted Cosine similarity?*

[1] `https://en.wikipedia.org/wiki/Jaccard_index`

6 Neighbourhood-Based Predictions

6.1 Questions

N.1 *Implement the k-NN predictor. Do not include self-similarity in the k-nearest neighbours. Using $k = 10$, `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 ($p_{1,1}$), and make sure that you obtain an MAE of 0.8287 ± 0.0001 on `data/ml-100k/u2.test`.*

N.2 *Report the MAE on `data/ml-100k/u2.test` for $k = 10, 30, 50, 100, 200, 300, 400, 800, 942$. What is the lowest k such that the MAE is lower than for the baseline (non-personalized) method?*

N.3 *Measure the time required for computing predictions (without using Spark) on `data/ml-100k/u2.test`. Include the time to train the predictor on `data/ml-100k/u2.base` including computing the similarities $s_{u,v}$ and using $k = 300$. Try reducing the computation time with alternative implementation techniques (making sure you keep obtaining the same results). Mention in your report which alternatives you tried, which ones were fastest, and by how much. The teams with the correct answer and shortest times on a secret test set will obtain more points on this question.*

7 Recommendation

7.1 Questions

R.1 *Train a k-NN predictor with training data from `data/ml-100k/u.data`, augmented with additional ratings from user "944" provided in `personal.csv`, using adjusted cosine similarity and $k = 300$. Report the prediction for user 1 item 1 ($p_{1,1}$).*

R.2 *Report the top 3 recommendations for user "944" using the same k-NN predictor as for **R.1**. Include the movie identifier, the movie title, and the prediction score in the output. If additional recommendations have the same predicted value as the top 3 recommendations, prioritize the movies with the smallest identifiers in your top 3 (ex: if the top 8 recommendations all have predicted scores of `5.0`, choose the top 3 with the smallest ids.) so your results do not depend on the initial permutation of the recommendations.*