Programming Assignment 1

Collaborative Movie Recommendation

Tulio Criscuolo
Computational Mathematics Student
Universidade Federal de Minas Gerais

1. INTRODUCTION

According to [2], recommendation algorithms uses users's interests that can be obtained explicitly (rating) or implicitly (purchase) to recommend items or generate a list of items for their costumers. As described in [4] , Collaborative filtering (CF), is an algorithm for recommendation that bases its prediction and recommendations on users past ratings and behaviour in the system and it's core assumption is that if users showed similar preference about some items, then they will likely agree about other items.

In this assignment, a baseline and three classical memory-based approach for CF were implemented user-based, item-based and matrix factorization plus a meta algorithm combining their result with the goal to make prediction about users preference about movies.

2. ALGORITHMS

2.1 Baseline

The first algorithm implemented for this assignment was a baseline presented in [4]. The baseline algorithm was useful for pre-processing and normalizing the data for more elaborated algorithm and predicting ratings for some cases where a user or item is unknown from the training set. The baseline prediction for user u and item i will be denoted as b_{ui} is given by the following expression:

$$b_{ui} = \mu + b_u + b_i \tag{1}$$

Where μ is the global average rating and b_u and b_i are users and items bias, which measures how much the user or item rating deviates from the average. Let r_{ui} be the rating of user u to item i, I_u the set of items rated by u, U_i the set of users who rated item i, and β_u, β_i a dumping value for user u and item i then user and item bias are given by:

$$b_u = \frac{1}{|I_u| + \beta_u} \sum_{i \in I_u} (r_{ui} - \mu) \tag{2}$$

$$b_i = \frac{1}{|U_i| + \beta_i} \sum_{u \in U_i} (r_{ui} - b_u - \mu) \tag{3}$$

The interpretation for the baseline is given by [3], for example if the global rating mean μ is 7.3 and a user u is a soft rater and rates 1.2 above the average and the item i is a criticised item and is rated 1.0 lower than the average. Thus, the baseline prediction for user u to item i will be 7.5.

To build the baseline model it is necessary to compute the global rating mean which is $O(t)$, t is the number of triple $(user, item, rating)$, plus two linear steps $O(m)$ and $O(n)$ to compute user and item bias, where n and m are the number of user and item, thus the baseline complexity is $O(t + n + m)$. Assuming that the number of triples is much bigger than the number of items and user the baseline complexity is $O(t)$.

2.2 User Based Collaborative Filtering

The user based CF, was first introduced by [5] and it is also known as $k - NN$ (k nearest neighbours). The idea behind the user-based CF is that users whose rating behaviour in the past is similar will have a similar rating behaviour in the future. For example, to predict a user u preference to an item i, denoted as p_{ui}, first it is necessary to build a neighbourhood for user u using some similarity metric and next the information from this neighbourhood is aggregated to make the prediction.

In this assignment, the similarity metric chosen for User Based CF was the Pearson's correlation. The Pearson's correlations, suffers from the problem of giving users with a few common rating a high similarity, to solve this problem a threshold for a minimum of number of items in common was used. Let I_u be set of items rated by the user u and $\bar{r}_u$ is the user u rating mean, the Pearson's similarity $s(u, u')$ between two user is given by:

$$s(i, i') = \frac{\sum_{i \in I_u \cap I_{u'}} (r_{ui} - \bar{r}_u)(r_{u'i} - \bar{r}_{u'})}{\sqrt{\sum_{i \in I_u \cap I_{u'}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_{u'}} (r_{u'i} - \bar{r}_{u'})^2}} \tag{4}$$

After building a neighbourhood for each user using the Pearson's similarity, the information from their neighbour is combined to make a prediction p_{ui}, for user u to item i. This was done by computing the weighed average and a normalization to a $z - score$ given by the following formula:

$$p_{ui} = \bar{r}_u + \sigma_u \frac{\sum_{u' \in N} s(u, u')(r_{u'i} - \bar{r}_{u'})/\sigma_{u'}}{\sum_{u' \in N} |s(u, u')|} \tag{5}$$

Where, σ_u is the user u standard deviation, N is the set of neighbour's of u and $s(u, u')$ is the similarity between users u and u'.

There are several parameters to improve the User Based CF, i.e the number of neighbours to consider, a similarity threshold and the number of rating that a user has to have to be considered to have a neighbourhood also know as a cold start. To improve the user-user accuracy a few values for those parameters were tested manually.

The time complexity to build a neighbourhood for all users is $O(n^2m)$, n is the number of users and m is the number of items, because for each pair of user it has to compute the similarity between them which is linear in the number of item. The time complexity to compute the weighed average to make a prediction is $O(km)$, where k is the number of neighbour considered.

2.3 Item Based Collaborative Filtering

Even though user-user CF works well, in a real recommendation system it suffers from scalability problems, it is expensive to compute a user neighbourhood on-line and the user neighbourhood is not stable to be pre-compute off-line. Item-item CF, was first used by Amazon [1]. Overall, item-item CF, is similar to user-user CF, but instead of using similarities between users to predict how much a user will like an item, item-item CF uses the idea that items are similar if they have a similar rating pattern and that users will show similar preference for similar items.

Item-item CF first builds a neighbourhood N for each item i, which is more stable than users neighbourhood, next to make a prediction p_{ui} of how much user u will like item i, it computes an weighed average of the k most similar item to i that were rated by u. In practice the item neighbourhood N is pre-computed off-line and only the top n most similar item to i is maintained in N, usually $n >> k$. The item similarity $s(i, i')$ between item i and i' used in this assignment was the cosine and the aggregation rating p_{ui} was computed using an weighed average, both expression are given by the equations below.

$$s_{i,i'} = \frac{r_i^T r_{i'}}{||r_i||_2||r_{i'}||_2} \tag{6}$$

$$p_{ui} = b_{ui} + \frac{\sum_{i' \in N} s(i,i')(r_{ui'} - b_{ui})}{\sum_{i' \in N} |s(i,i')|} \tag{7}$$

Where $r_i^T r_{i'}$ and $||r||_2$ is the dot product between ratings of items i and i' and the l^2 norm of the vector r respectively, note that a missing rating is considered to be zero, b_i is the bias of item i, which is computed using the baseline algorithm and b_{ui} is the baseline prediction for user u to item i. To improve the item-item CF accuracy, a few extra parameters were added, i.e minimum similarity threshold, minimum number of users who rated the item and maximum number of neighbour. A few values for those parameters were tested manually.

To build a neighbourhood for all item it is necessary to compute the dot product between all pair of item, which is linear in the number of users, thus it is necessary $O(m^2n)$ operations where m and n are the number of item and users.

2.4 Dimensionality Reduction

On the previous described algorithms, users and items are represented in a high dimensional space where a user $u \in R^{|I|}$ and a item $i \in R^{|U|}$, where I and U is the set of item and user respectively. As it can be seen, a user and item belongs to a high dimensional space with different dimension and it has lots of zero entries. The idea of dimensionality reduction is to represent users and items in a k dimensional space where we can compare user and item. There are a few dimensionality reduction methods, principal component analysis (PCA), probabilistic matrix factorization and singular value decomposition (SVD), in this assignment the SVD was used to represent users and items in R^k.

The SVD used in this assignment is a simple version presented in [3], which does not consider time or neighbourhood information and has some modification from the classical SVD defined in mathematics which needs a matrix to have all it' entries. The idea is that we can represent a user u and item i as a feature vector $p_u, q_i \in R_k$, and make a prediction $\bar{r}_{ui}$ by computing:

$$\bar{r}_{ui} = p_u^T q_i + \mu + b_u + b_i \tag{8}$$

The parameters μ, b_u and b_i are the global mean, user bias and item bias respectively. Let R be the set containing all ratings and λ be the learning rate the goal is to minimize the following objective function.

$$\sum_{(u,i) \in R} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(b_u^2 + b_i^2 + ||q_u||^2 + ||p_i||^2) \tag{9}$$

There are two common approach to minimize this objective function, one is the stochastic gradient descendant and the other is the alternating least square. In this assignment the stochastic gradient descendant was used, it needs a learning rate λ which indicates how much we are going to move in the gradient direction and regularization value η to avoid over fitting, the algorithm works as follows:

for all $(u, i) \in R$ **do**
  $d_{ui} \leftarrow r_{ui} - p_u^T q_i$
  $\Delta p_u \leftarrow \eta(d_{ui} p_u - \lambda q_i)$
  $\Delta q_i \leftarrow \eta(d_{ui} q_i - \lambda p_u))$
  $p_u \leftarrow p_u + \Delta p_u$
  $q_i \leftarrow q_i + \Delta q_i$
end for

First, all p_u and q_i are assigned to an arbitrary number (not zero), e.x 0.1, then at each iteration it uses the gradient of the objective function to update each user and item vector, this procedure is done until SVD converges to a local optimal solution. A few modifications were made so that the SVD would converge to a reasonable solution in a shorter time, e.g it does not wait the SVD to converge to a local optimal solution instead it uses a maximum number of iterations and rather than using all pairs $(u, i) \in R$ it randomly selects a pair $(u, i) \in R$.

Let n and m be the number of user and item, for each iteration the SVD needs $O(k)$ operations to compute the dot product between user u and item i and update the vectors p_u and q_i, to acess the user and item bias is $O(1)$, since it was pre computed by the baseline which is linear in the number of pairs (user,item). Since it runs for c (c is a big number) iterations the SVD complexity is $O(ck)$.

3. GENERAL OBSERVATIONS

3.1 Dataset

The training data used in this assignment was a csv file containing 34,105 users and 19,498 items and 336,672 triple of $(user, item, rating)$. The test data contains 77,276 pairs of $(user, item)$ for prediction, predicted values for the test data were automatically evaluated by submitting it predictions to Kaggle. Since, Kaggle accepted only two submissions per-day the training data was randomly split in two smaller training and test data-set for an off-line evaluation

each one containing 70% and 30% of the original training set data.

3.2 Evaluation Criteria

The evaluation metric used was the Root Mean Squared Error (RMSE). The RMSE, is commonly used in recommender systems literature, let r_i and p_i be the actual and predicted value for the $i-th$ pair $(user, item)$ and N be the number of predictions made, then RMSE is given by:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - p_i)^2} \tag{10}$$

Smaller RMSE value indicates better prediction.

3.3 Implementation decisions

All the algorithms described in section 2, were implemented and tested using the programming language C++ 11 compiled using gcc 4.8.4 on a Ubuntu 14.04 machine with quad core processor Intel $i5$ with 3.20Ghz. The implementation is object oriented, in other words there is one object for each CF algorithm and two more objects one to represent the training/test data and one to represent the matrix of ratings.

4. EVALUATION

4.1 Empirical

Even though, the baseline algorithm is simple and easy to implement it had a good performance, reaching a low RMSE (less than 1.6 RMSE on Kaggle!) in a short running time (less than one second), the best values found for β_u and β_i were 5 and 2 respectively, those values were found manually by changing it arbitrarily.

The user-user CF, was a bit more complicated to find reasonable parameters than the baseline. The parameter which had the most impact on the user-user CF result was the minimum number of rating that a user has to have to have to apply user-user CF, for example users with less than 5 ratings were not stable enough such that user-user CF would not make a good prediction, therefore for those users the algorithm just ignored them and returned the baseline prediction. Others parameters such as the minimum similarity threshold and number of neighbour also showed a significant impact on the algorithm performance, a few arbitrary number were tested, and it was possible to observe that with a low similarity threshold and a high number of neighbour the accuracy decreased. A reasonable set-up for user-user was a minimum similarity threshold of $1e^{-4}$, at most 50 neighbour and at least two neighbour and three rating to apply user-user CF.

For item-item CF, the parameter tuning followed similar steps to user-user CF. The main parameter was the minimum number of rating that an item has to have to apply the algorithm, if it has a number of rating smaller than the threshold the baseline prediction is used. The similarity threshold and number of neighbour's for item-item CF also had a significant impact on the algorithm accuracy, if the similarity threshold was set too high an item would not have a neighbour or if the threshold was too low a lot of items were added to the item neighbourhood adding a lot of noise. The best set-up found for item-item was a similarity threshold of $1e^{-4}$, at most 40 neighbour, at least 3 ratings to be considered for item-item and two items are similar if they were rated by at least two users in common.

Algorithm	RMSE	Execution time
Baseline	1.510	0m08s
User based	1.538	1m04s
Item based	1.544	0m52s
SVD	1.495	0m39s
Meta	1.483	2m10s

Table 1: Algorithms evaluation on a local test-set

The SVD, was the algorithm which showed the best accuracy performance, on the other hand it was the one which needs more execution time. It could be noticed that its prediction quality depends heavily on the baseline and the number of iterations, usually a higher dimensional space showed better results but it demanded more computational resource. Thus, to keep a low execution time reasonable parameters for SVD were to represent user and item in a 3 dimensional space, learning rating of $1e^{-3}$ and $8e^{-2}$ for regularization.

After several examinations, it could be noticed that the user-user and item-item has a lot of space for parallelization to compute user/item neighbourhood, thus user/item neighbourhood are computed in parallel using multi thread, where each thread takes a range of user to compute the neighbourhood, by default the source code uses 4 threads. An extra observation is that it is only necessary to compute neighbourhood of user/item that are on the test set, those two observations could reduce the execution time significantly.

To improve the overall accuracy, all three methods were trained in parallel and a meta model were used. The meta model uses a simple weight average to compute the final score, i.e 0.4 for SVD and 0.3 for user and item based CF. This weighing score were learned by changing the values arbitrarily, they might not be optimal but produced a slight improvement if only one algorithm were used to make prediction.

As it can be seen at table 1, the baseline algorithm had an incredible performed reaching RMSE of 1.510, which was a smaller error than user and item based CF, and its is also important to note that the baseline runs faster than both of those algorithms. Next, we have the user based CF which had the worst RMSE performance (1.532), even though the lowest RMSE it could be oberseved that user based CF was able detect lower user score, which the baseline algorithm was not able to detect, thus it would include some extra information to the meta algorithm. The item-based, had a similar performance to user-based but its performance was lower but with a slight smaller running time. The best overall performance was achieved by the SVD, it could give the smallest RMSE in a shorter time in comparisson to user/item based CF even though it is representing users and items a small dimensional space.

4.2 Kaggle Performance

The meta algorithm, which uses a weighed average of user/item CF and SVD, had a good performance on the final scoring on kaggle, being able to reach the first position with the best submission reaching a RMSE of 1.57226, which was significantly better than the second position that had a RMSE of 1.62991. It is important to say that those param-

eters described in this paper does not represent best Kaggle submission, since a lot of parameters were tested and it was difficult to keep track of all the changes. In general, the meta algorithm time execution was kept to run faster than 3 minutes, the described meta algorithm runs on average of five execution was 2m04s.

5. CONCLUSIONS

Overall, the classical user-user and item-item based could provide a reasonable RMSE, but they were outperformed by the baseline, which is a simple algorithm. Also, the baseline had a significant impact for the SVD performance, since the SVD predictions uses the baseline prediction plus a dot product between the user and item vector, thus finding reasonable parameters for the baseline was important. The weighed average of all three classical model could provide a slight better RMSE most likely for being able to detect low/high ratings score.

6. REFERENCES

[1] Amazon. Collaborative recommendations using item-to-item similarity mappings, 09 1998.

[2] B. S. Greg Linden and J. York. Amazon.com recommendations item-to-item collaborative filtering. *IEEE*, pages 76–79, February 2003.

[3] Y. Koren. Collaborative filtering with temporal dynamics. *ACM, KDD*, July 2009.

[4] J. T. R. Michael D. Ekstrand and J. A. Konstan. Collaborative filtering recommender systems. *HCI*, 4:81–173, November 2010.

[5] M. S. P. B. P. Resnick, N. Iacovou and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. *ACM*, 1994.