Challenge report - Group 3

Netflix Challenge: Movie rating prediction

Tu Delft 2020-2021
Course: Data Mining (CSE2525)
Course Teacher: Gosia Migut
22-January-2021

Student names:
Hugo Koot - 4665945
Kevin Nanhekhan - 4959094

Kaggle Teamname: Group 3
Kaggle ID’s: hugokoat and kevinrn

1. Introduction

From 2006 to 2009, the company Netflix held a competition where contestants needed to create an algorithm that would predict user ratings for certain movies based on previous ratings [1]. Here the predicted rating indicates how much an user liked a certain movie where the error of the predicted rating compared to the true rating would be measured through the root-mean-square deviation (RMSD). Therefore the meaning behind the predicted ratings is to recommend the most likely movies a particular user would watch. For the most accurate results, the lowest error needs to be achieved.

The assignment is based on this competition where a predicted rating should be calculated through an algorithm for the user/movie listed in *predictions.csv*, based on the user data (*users.csv*), the movie data (*movies.csv*) and previous ratings (*ratings.csv*). The predicted rating should afterwards be saved to *submission.csv*. The goal we are trying to achieve for this assignment is to apply the lecture material from the Data Mining course by implementing variations of different algorithms, comparing its results and combining the best performing ones that would be as accurate as possible in predicting the ratings. Expected from the readers of this report is to possess knowledge on the course material as a short, concise and global explanation will be given. The referenced tables and graphs can be found in the Appendix on the last page.

2. Methodology

In order to calculate the predicted ratings, two different Collaborative Filtering (CF) algorithms have been used. The first one is based on the K-Nearest Neighbour (KNN) algorithm and its variations and the second algorithm is based on Matrix Factorization (MF) and its variations. By combining these two algorithms a multi-scale modeling of the data can be performed from which a more accurate prediction can be estimated. The KNN handles the local patterns of the data while MF handles the 'regional' effects. Additionally for each algorithm the overall deviations of users and movies will be modeled in order to handle the global effects of the data. By comparing these we take the best performing from each and combine them.

2.1 Collaborative Filtering: K-Nearest Neighbour

First an utility matrix is created containing the similarity between users using the similarity measure of pearson correlation. We have decided to replace the missing values with 0 to indicate no similarity. After that we loop through the predictions file row by row. Here we take the similarity weights from the utility matrix of the corresponding user we want to predict for and as well as the ratings from the movie we need to predict. For this we decided to only use the k (in our case k = 25) most similar users or items as it made calculations slightly faster. Additionally we also decided to discard ratings that do not exist (missing ratings containing 0) to avoid errors and easier calculation of the weighted average. We check if it is a valid value and correct it otherwise (e.g. 0 for the ratings that could not be predicted). We continue the loop until all predictions are given and return the matrix with its now filled predicted ratings.

For our improvements we made the type of correlation into an item-item instead of user-user by calculating the similarity between the movies instead of the users.

We also added the baseline modelling (deviation or bias) at the calculation step of the weighted average. Here we not only added the baseline estimate to the weighted average value but also assigned only the baseline estimate to ratings that could not be predicted instead of 0.

While not necessarily an improvement, we also wanted to look at the difference between the similarity measures of Cosine and Pearson considering those are similar in calculation with the exception to Pearson being invariant to adding constants (due to using centred vectors) [2].

2.2 Collaborative Filtering: Matrix Factorization

For MF, we used the singular value decomposition (SVD) and reduced the space of the decomposed matrices. SVD does not handle missing ratings well so the ratings were centred through subtracting the mean of a certain axis, setting the missing ratings to 0 where afterwards we add the mean back. For the calculation we used the standard SVD function from the numpy library to get the decomposed matrices U (user-to-feature), Sigma (eigenvalues indicating the strength of each feature) and V (movie-to-feature). In order to reduce the dimension space we would need to set the lowest eigenvalues to 0 which is the same as taking the highest values only. Considering the eigenvalues in sigma are sorted in descending order because of its numpy implementation [3],

we can just take the first k features and also take those from the corresponding rows from U and columns from V. In our case that would be k = 25 as that gave the best results (see table 3 and Graph 1). Taking the dot-product of this gives a new rating matrix.

As mentioned earlier SVD does not handle missing ratings well so in order to get better decomposed matrices a new approach is needed that approximates the values for it through a stepwise calculation until it reaches convergence. We have decided not to do Gradient descent as variation and instead do Stochastic Gradient descent (SGD) as the latter computes gradients faster. This is because the hardware to run the algorithms on would take a long time while there is a limited time to do so. Especially in case something goes wrong (which did happen a few times) enough time was needed to rerun the algorithm and get its results. In the original netflix competition 200 steps [4] reached convergence but due to time constraints we lowered this number to 100. For the learning rate we decided on 0.001 and for the regularization term 0.01 as those were close to the ones used in the original netflix competition [4]. For SGD we also included regularization to avoid overfitting and baseline modelling to include the global effects.

3. Results

To find out if something was an improvement or not we uploaded the resulting submission file into kaggle [5] to evaluate its RMSE score. In graph 2 we can see that item-item consistently performs better than user-user. For the K-nearest neighbour approach. We can also see that adding a baseline estimate was advantageous for item-item, but disadvantageous for user-user. Pearson correlation does however beat out cosine similarity consistently. And through our method we could find the best score in the item-item variation with pearson correlation and a baseline estimate. This algorithm also scored best overall for us as seen in table 1. In table 2 we can see that for matrix factorization, Stochastic gradient descent performed a lot better than svd. For the optimization of svd we found that 25 features were optimal as seen in table 3 and graph 1. We continued to use this k = 25 for all algorithms to stay consistent. Our combined algorithm (Item-Item with Pearson, SGD with a baseline estimate in both) also scored slightly worse (0.85641) than our best collaborative filtering (0.85541).

4. Discussion, Conclusion and Future Work

The main goal that we have been trying to solve was implementing various algorithms and combining the best ones into one algorithm that estimates a missing rating as accurately as possible to what its true rating would be. Item-item worked better than user-user which was expected, because items are simpler than users, thus we can find better and more similar examples. We also found that pearson correlation outperformed cosine similarity, this was also expected as cosine similarity considers missing values as negative, while in the real world this is not the case. And in our given data set there were a lot of missing values. We also found that baseline estimates only had a positive effect on item-item, we think this is because the distribution of ratings a movie gets is usually in normal form with the peak at 1 given rating [6]. While the distribution of ratings that a user gives can often have multiple peaks. This will result in the user usually being further of their baseline estimate than a movie is. While not in the same range as what was shown in course material (most likely caused by not fully optimized implementation), the results of 'normal' SVD compared to SGD still holds true to the fact that SGD (includes modelling the baseline) outperforms SVD.

When combining the best scoring algorithms for global, regional and local modeling (Item-Item with Pearson, SGD and in both a baseline estimate included) we got a slightly worse score (0.85641) than our best variant of collaborative filtering (0.85541). We expected this to be better, as this algorithm should be a more complete picture of the data. We suspect the chosen constant values to not be fully optimised. For example, the SGD probably does not reach convergence with just 100 steps compared to the 200 needed in the original netflix competition. Further testing would be needed to confirm this as well as comparison of more values.
For future work we could explore using Jaccard similarity, weighted sum instead of weighted average, gradient descent, temporal biases as we did not fully utilise the data from users.csv and movie.csv and content based recommenders. We would expect that using weighted sum calculation for example would result in an immediate improvement on our best score. Measurements besides RMSE such as mean absolute error (MAE), frobenius norm and alternating least squares (ALS) should also be explored to get more accurate results.

References

[1] Netflix (2009) Netflix Prize. Archived to the *Waybackmachine* from the original on 2009-09-24. Accessed on 22 January 2021, from https://web.archive.org/web/20100106185508/http://www.netflixprize.com//rules

[2] Egghe, Leo & Leydesdorff, Loet. (2008). The relation between Pearson's correlation coefficient r and Salton's cosine measure. Journal of the American Society for Information Science and technology, 60(5). Accessed on 22 January 2021, from https://www.leydesdorff.net/cosinevspearson/

[3] SVD function from the numpy library. Accessed on 22 January 2021, from https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html

[4] Caio B. Nóbrega, Leandro B. Marinho (2014) Predicting the Learning Rate of Gradient Descent for Accelerating Matrix Factorization. *Journal of Information and Data Management*, 5(1).

[5] CSE2525 Data Mining 2020/2021. Accessed on 22 January 2021, from https://www.kaggle.com/c/cse2525-data-mining-20202021/overview

[6] Mustafa Katipoglu (2020) User-based vs Item-based Collaborative Filtering. *Medium*. Accessed on 22 January 2021, from https://medium.com/recommendation-systems/user-based-vs-item-based-collaborative-filtering-d40bb49c7060

Appendix

Nearest Neighbour			
Type of correlation	Similarity measure:	RMSE score	
		standard	with baseline modelling
User-User	Pearson	0.96309	1.08089
	Cosine	1.00371	1.11267
Item-Item	Pearson	0.90242	0.85541
	Cosine	1.19552	0.88527

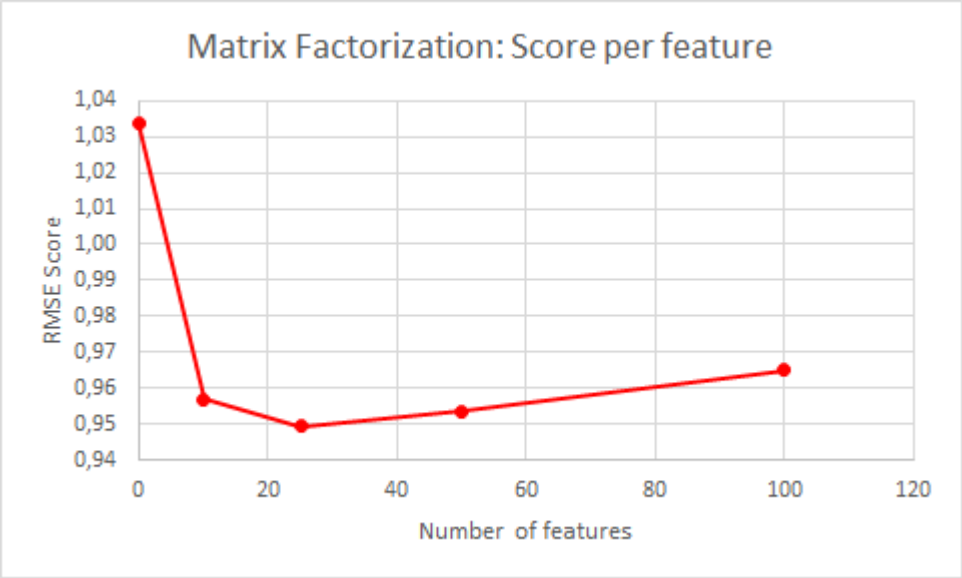
Table 1: *RMSE scores for variations of the Collaborative Filtering K-Nearest Neighbour*

Matrix Factorization	
	RMSE Score
SVD	0.94942
Stochastic gradient descent	0.89833

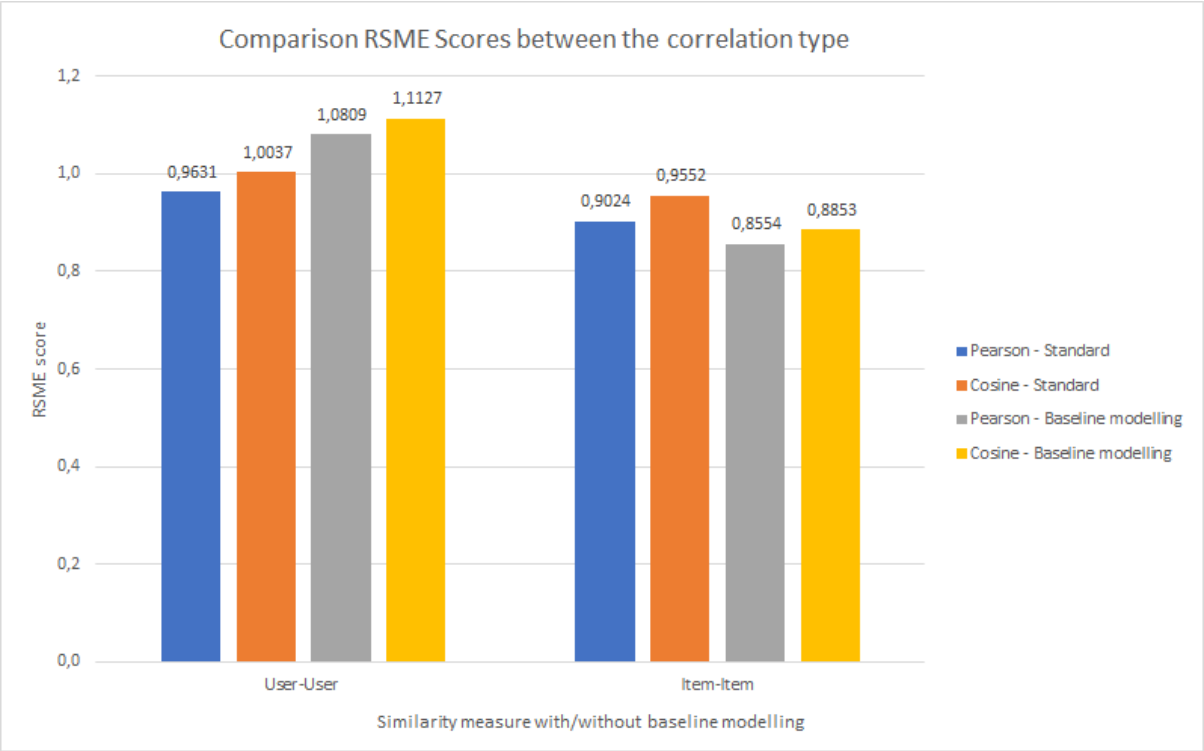
Table 2: *RMSE scores for variations of the Collaborative Filtering Matrix Factorization*

features	0	10	25	50	100
score	1,03378	0,95687	0,9493	0,95361	0,9649

Table 3: *RMSE scores for each feature used in Matrix Factorization*



Graph 1: *Score per feature used in Matrix Factorization*



Graph 2: *Comparison RSME Scores between the correlation type*