S3933758: PDS Assignment 3

Alex Musumeci

October 2022

1 Introduction

The basic idea of Recommender systems (RS) is to aid users in choosing the most appropriate item from the abundance of options available [1, 2, 3]. Although there are various approaches, Collaborative Filtering (CF) is one of the more common RS types[3]. CF generally relies on making connections between users and items based on past transactions, CF can be divided into two main subcategories, *latent factor models* and the *neighborhood approach*[1, 2]. Neighbourhood approach models look at similarities between users, or, items. If looking at similarities between items, a user is transferred to the item space with preferences based off the users ratings of similar items[1, 2]. Latent factor models are different, comparing users and items in the same space[1, 2]. The Netflix Prize in 2006 resulted in a huge surge of interest in CF as various researchers competed to come up with a RS that would beat the companies own RS. The reward was 1 million USD with researchers using both *latent factor models* and *neighborhood approach* CF. Singular Value Decomposition (SVD) is a latent factor model made famous by Simon Funk in the Netflix Prize[5]. Like the Netflix Prize, this paper strives to get the best Root Mean Squared Error (RMSE) score using a subset of the Netflix Prize Data, walking through the methodology and experimental design used to get these scores for four CF algorithms. This process will highlight the differences of the CF algorithms used, and in doing so, illustrate what makes a good RS.

2 Methodology

There were a total of 4 RS algorithms used, this section describes each one and outlines the rationale for their selection. Algorithms were trained and tested on pre split (80/20) data.

When considering and comparing accuracy's of various RS algorithms it is important to have a benchmark or reference point, an algorithm that does not do much work. This is exactly what the BaselineOnly algorithm does, it is neither a *latent factor model* or a *neighbourhood approach* to CF.

SVD is a type of latent factor model whereby each user has a user-factor vector, and each item has an associated item-factor vector[1, 2]. These factors

can be seen as things which help to explain ratings, in the user space examples could be whether a user prefers drama to comedy or what ratings they generally give[5]. Whilst in the item space these factors could be what genre it is or what actors are starring[5]. A prediction of a users rating is then derived from summing the various factors relating to that movie and that user[5]. SVD can be either biased using baseline estimates (b_i, b_u) or unbiased.There are a multitude of parameter settings within SVD which can be tuned to optimise performance. The rationale for choosing this algorithm was it's success in the Netflix Prize and that it is only one of two algorithms in the matrix vectorisation subsection in the Surprise library.

KNNWithMeans uses a neighbourhood approach to generate predictions. Taking users mean ratings helps adjust for extremely high or low ratings[7]. Similar to (bsl_options) in baseline estimates, (sim_options) is a dictionary of parameters which control how the similarity measures are configured. One key parameter within (sim_options) is user_based, defining whether the algorithm will be user or item based[1, 2]. When comparing the various surprise algorithms it makes sense to choose at least one of the K-NN based options. Further, upon initial hyper-parameter experimentation with KNNBasic and KNNZscore, KNNWithMeans performed the best in terms of RMSE.

CoClustering represents a different category of CF algorithm to the aforementioned three. It works by obtaining item and user neighbourhoods via co-clustering, predictions are then generated through the average ratings of co-clusters[3]. Advocates of this method suggest it can return similar accuracy's to those found in algorithms such as SVD for a fraction of the computational cost, hence its use in this comparison was deemed justifiable.

3 Experiments

Table 1 shows the four metrics and their scores for the four different algorithms used. The aim of experimental design was to achieve the best RMSE score, close to 0.8. BaselineOnly was the worst performing algorithm in terms of RMSE. The BaselineOnly algorithm takes only the baseline estimates (bsl_options) as parameter inputs to generate predictions[8]. These bsl_options account for large user and item effects[8]. For example, some users may rate items higher than the average user whilst some items may be rated higher than others[1, 2]. There are two ways to estimate baselines; Stochastic Gradient Descent (SGD) and Alternating Least Squares (ALS)[8]. SGD uses the one regularising parameter ('reg') to combine the user (b_u) and item (b_i) deviations from the average into the one calculation, penalising large b_u and b_i deviations[1, 2]. ALS instead decouples b_i and b_u through using two separate regularisation parameters for items (reg_i) and users (reg_u) respectively[1, 2]. For this experiment both ALS and SGD methods were used via the 'method' parameter, for each a RandomizedSearchCV() was used to try the relevant parameters for both methods. In each case only tiny gains were found in the RMSE when compared to using default parameters and as Table 2 shows no best values were found and default

parameters were used to generate the RMSE score seen in Table 1.

Algorithm	RMSE	MAE	MSE	FCP
BaselineOnly	0.9	0.708	0.721	0.724
KNNMeans_User	0.859	0.67	0.738	0.718
KNNMeans_Item	0.849	0.661	0.721	0.724
SVD	0.83	0.647	0.689	0.738
Coclustering	0.876	0.685	0.767	0.698

Table 1: Collaborative Filtering algorithm metric scores

The next highest RMSE score was CoClustering. There are only three parameters to choose when configuring this algorithm, n_cltr_u and n_cltr_i determine the number of clusters for users and items respectively[8]. Whilst n_epochs determines the number of iterations of the optimization loop[8]. For this algorithm hyper-parameter tuning was done via Randomized Grid Search, this option is usually more computationally expensive relative to the RandomizedSearchCV(). However, when constructed in a for loop and iterated over one parameter at a time it can increase efficiency. The for loop constructed for this experiment should have multiple iterations to ensure parameters escape a local minima. Parameter choices and the best values are displayed in Table 2.

Method	Hyperparameter	Test Values	Best Value
BaselineOnly	method reg_u reg_i reg	SGD,ALS 8, 9, 10, 11, 12 3, 4, 5, 6, 7 0.02,0.05,0.1	N/A
CoClustering	n_cltr_u, number of user clusters n_cltr_i, number of item clusters n_epochs	2,4,8,16,32,64,100 2,4,8,16,32,64,100 5,10,15,20,25,30	16 8 20
KNNMeans: User	K name shrinkage user_based=True	20,40,50,55,60,65,70,80 msd, pearson, pearson_baseline,cosine 50, 70, 90,100,110,150	55 pearson_baseline 110
KNNMeans: Item	Same as above except user_based=False	Same as KNNMeans: User	*Same as* *KNNMeans: User*
SVD	n_factors n_epochs lr_all reg_all init_mean init_std_dev	50,65,75,85,65,110,120,130 20,30,50,60,70 0.004,0.005,0.006,0.008,0.01 0.02,0.04,0.045,0.05,0.06,0.075,0.08,0.1 0.125,0.15,0.2,0.25,0.5 0.02,0.03,0.04,0.05	130 50 0.005 0.045 0.15 0.02

Table 2: Hyper-parameter options and best choices for the four CF algorithms

As seen in Table 1 the next best performing algorithm was actually split into two sub categories as a result of the differences seen in the RMSE scores when changing from user based (0.859) KNNMeans to item based (0.849). The inclusion of item based occurred late into the experimentation phase. This is

because a RandomizedSearchCV() was initially set up for both user and item based KNNMeans and early in the searching it seemed that user based was the best option, so this was set for the remaining searches. The same approach was taken for the KNNZscore algorithm, it preferred user_based=True in early searches. However, these eventual 'best values' used to generate predictions for these two algorithms showed very similar distributions in terms the predictions and errors, additionally their RMSE scores were similar. Because some of the best RS use integrated CF models[2], two algorithms doing the same thing is not a good thing as they would not enrich each other[2]. Therefore, in anticipation of any future integration re-experimentation was conducted, leading to the implementation of an item based KNNMeans model. As can be seen in Table 2 all remaining parameters were kept the same, and importantly, the resulting prediction and error distributions were different from the user based option as seen in Figure A.2 and Figure 1, respectively. Item-based consistently performs better on the Netflix data, a key reason for its success in movie ratings are these items are quite static when compared to other media items i.e., newspapers, it is also more scale-able and efficient with far less items than users [9, 9].

SVD was the winner in terms of RMSE, as seen in Table 1. Hyper-parameters were tuned using a RandomizedSearchCV(), with option's and eventual best values outlined in Table 2. init_mean and init_std_dev were only introduced into the search after extensive tuning had been undertaken on the other parameter's listed in Table 2. Understanding the true ratings of the test data used to generate the ratings predictions aids in interpreting the ratings predictions. As can be seen in Figure A.1, a movie rating of four was most common, followed by three, five, two and finally one was the least common rating score, users prefer rating movies they enjoy. Figure A.2 shows these true movie ratings as a box-plot and compares them to the other predicted algorithm ratings, the right skew of the test ratings is seen here with the median beyond the mean. Figure A.2 shows how the other predictions were influenced by the small amount of true '1 ratings', with high numbers of outliers beyond the lower whisker.

When comparing each algorithm to the other metrics, like RMSE, lower values are better for Mean Squared Error (MSE) and Mean Absolute Error (MAE), whilst higher values are best for Fraction of Concordant Pairs (FCP). Table 1 shows SVD performed best for all four of the metrics chosen. Figure 1 shows the distribution of errors for all algorithms used and supports what is seen in Table 1 in regards to the error based metrics. SVD has a relatively third quartile and the shortest upper whisker, there are also relatively fewer outliers present indicating it indeed performed the best out of the algorithms chosen.Table 2 shows that although better RMSE broadly correlate to better performance in other metrics this is not always the case. BaselineOnly bucks this trend, it has the highest RMSE and MAE but performs better in MSE and FCP, the latter two scores equalling those of KNNMeans_Item. MSE can be explained as Baseline has fewer relative outliers as shown in Figure 1, and MSE punishes outliers. Whilst FCP is an ordinal metric and completely different to the others. Although CoClustering was tuned to greater detail relative to some of the other algorithms its performance was not reflective, this is shown in poor

scores across Table 2 with no exceptions. Figure 1 confirms this as we see it has the second largest upper whisker with the largest error outliers. Figure 1 shows how different errors and performance can look on the same algorithm simply buy changing from user to item based, item based performance is consistently better on the graph with a smaller third quartile, upper whisker and lower outliers.

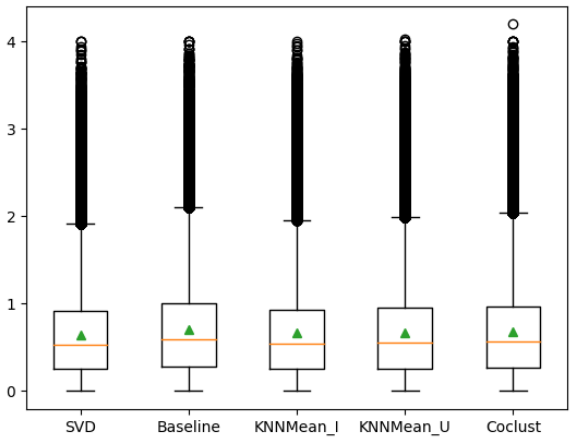

Figure 1: Distribution of prediction errors

4 Conclusion

In these experiments using RMSE, MSE, MAE and FCP as metrics, the SVD algorithm was a clear winner, performing best on all scores. This can be attributed to its ability to simultaneously consider factors in both the user and more importantly here, the items space [5, 10, 1]/. Although they are completely different algorithms, KNNMeans and KNNZscore performed similarly in terms of RMSE and overall performance when looking at errors, this may be due to flawed initial experimental design and/or the fact that these algorithms behave similarly on the data regardless. Revised experimentation outlined that changing KNNMeans from user to item based had a much bigger impact on performance. Finally, although RMSE is the benchmark metric used to evaluate CF model accuracy it may not always be the right option, this is evidenced when comparing the BaselineOnly algorithm which had the worse RMSE but the was equal second best in terms of MSE and FCP. Consequently, further research would look at other metrics such at top-K evaluations to enhance understanding[2]. Ultimately, these models when tuned correctly behave differently, and when properly tuned, an integrated CF algorithm could be used to generate better predictions[2].

References

[1] Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Transactions on Knowledge Discovery from Data (TKDD), 4(1), 1-24.

[2] Koren, Y. (2008, August). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 426-434).

[3] Al-Ghamdi, M., Elazhary, H., Mojahed, A. (2021). Evaluation of Collaborative Filtering for Recommender Systems. International Journal of Advanced Computer Science and Applications, 12(3).

[4] Koren, Y., Sill, J. (2013, June). Collaborative filtering on ordinal user feedback. In Twenty-third international joint conference on artificial intelligence.

[5] Funk, S. (2006, December 11). Netflix update: Try this at home. Retrieved October 25, 2022, from https://sifter.org/ simon/journal/20061211.html

[6] Chugh, A. (2020, December 8). MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better? Medium. Retrieved October 25, 2022, from: https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e

[7] Hatcher, E. (2021, December 5). How to Build a Recommendation System. Medium. Retrieved October 25, 2022, from: https://medium.com/@eli.hatcher/how-to-build-a-recommendation-system-e72fe9efb086

[8] Using prediction algorithms — Surprise 1 documentation. (n.d.). Surprise.readthedocs.io. Retrieved October 25, 2022, from https://surprise.readthedocs.io/en/stable/prediction$_algorithms.htmlBell, R.M., Koren, Y.(2007, Augus$
$basedcollaborativefiltering.InKDDcupandworkshopatthe13thACMSIGKDDinternationalconferen$
14).$sn.$

[9] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).

[10] Gower, S. (2014). Netflix prize and SVD. University of Puget Sound.

A Additional figures

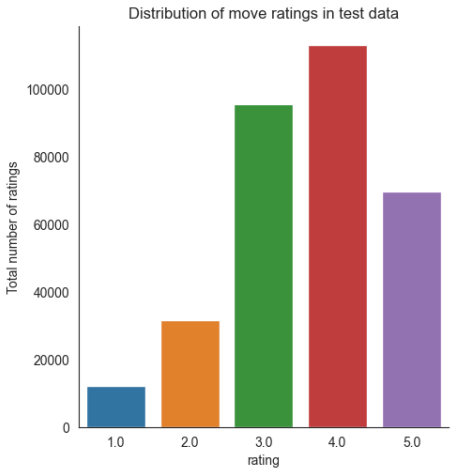

Figure A.1: Distribution of movie ratings in test data

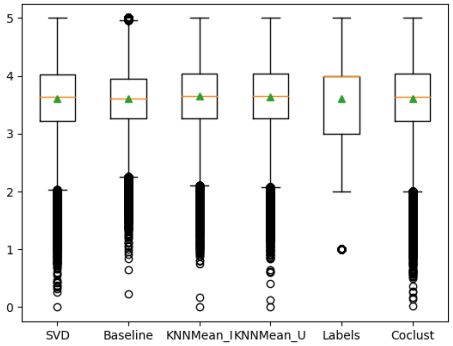

Figure A.2: Distribution of movie ratings predictions among algorithms compared true labels