Assignment 1: Recommender System

Tiffany Woodley

August 2019

1 Introduction

The purpose of this report is to run through the steps taken in order to investigate the validity of a company's business model. The business model encompasses selling movie recommendations based on different amounts of users, as the company's hypothesis is that the more users used in the making of a new recommendation, the higher the accuracy of the recommendation. The company however does not of yet have their own database of ratings, and so the MovieLens data-set is used as an approximate.

2 Data Set

The data used was taken from the MovieLens library which contains 20 Million ratings. Since the company was only interested in recommendations based on 5000 users or less, this amount of users was sampled from the initial 138 000 users in the data-set. There was high variation in the number of movies a user had rated, this can be seen in Figure 1, as 50% of the total ratings came from approximately only 20 000(+-15%) of all the users in the data-set. Although using users who have rated more movies would lead to better ratings, in a real-world situation user's number of recommendations would occur in a similar manor, and for this reason it was decided to sample the users randomly to maintain the distribution.

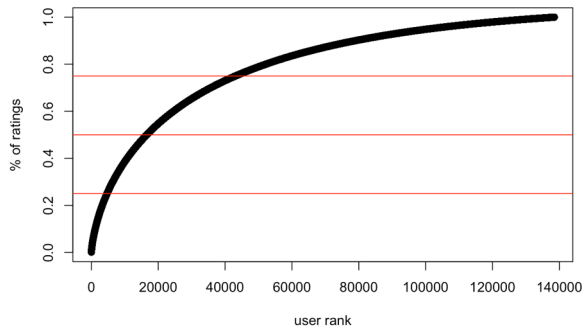

Figure 1: Percentage of ratings vs user rank

The data-set contains 27 000 movies. For computational reasoning a sub-sample of the movies was used. The highest ranked movies were taken since the more people that had watched them, the more likely it would be that other users would enjoy them as well. Figure 2 shows that the top 1000 ranked (based on number of ratings) movies contained around 50% of all the ratings, and so these were the movies used for the simulations in the rest of this report.

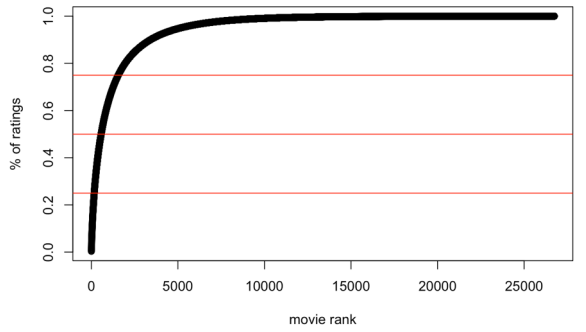

Figure 2: Percentage of ratings vs user rank

There were some double entries in the data-set where a user had rated a movie twice, these were removed from the data-set.

3 Train Test Split

The train test split ratio used was 90:10. The construction of the training data was done with a bit more consideration as it would be pointless testing on users, or movies, which had not appeared in the training process. The following steps outline the approach taken:

1. Create the training matrix, with one rating from each user.
2. Check the data to see if there are any movies which only have one rating.
3. If so, make sure these single rated movies are added into the training matrix.
4. Calculate how many samples are still needed to complete the training matrix.
5. Randomly sample from the remaining ratings, and add them to the training matrix.
6. The remaining ratings are then added to the test matrix.

4 User and Item Based Collaborative Filtering

4.1 Overview

Collaborative filtering uses other user's recommendations to come up with new recommendations, rather than using attributes about the item's being recommended themselves.

Of the different approaches, two memory based approaches are user and item based collaborative filtering, which use the ratings to find similarities between the users and movies respectively.

The user based approach finds similarities between the users, and then uses these similarities to predict what a user would have recommended an unseen movie based on what similar users have rated these movies as.

The item based approach finds similarities between movies, the user's unseen movies are then rated based on their similarities to the user's seen movies.

4.2 Method

This subsection goes through the process taken to generate a recommendation for a user using both the user and item based approaches.

The movies and users were put into a matrix as shown below, the rows represent the users, and the columns represent the movies. The matrix was populated with the respective ratings, if there was no rating for a specific user/movie pair, then a NA was inserted into the matrix. To calculate the similarities between users, the cosine similarity (method described in next sub section) between each row was found. To work out the similarities between movies, the cosine similarity between each column was calculated.

$$\begin{bmatrix} 5 & NA & NA & \dots & 3 \\ NA & 2 & 5 & \dots & NA \\ \dots & \dots & \dots & \dots & \dots \\ NA & NA & 3 & \dots & NA \end{bmatrix}$$

Once the user similarities were found, in order to get a predicted rating for a user of a particular unseen movie, the following steps were taken:

1. The users that have seen the movie are found.
2. These users' similarities are then scaled by dividing each similarity by the sum of themselves.
3. These scaled values are then multiplied by the ratings the respective users had given the movie and summed.
4. This value was then returned as the predicted rating that a user would give that movie.
5. If there were no similarities the average of the user's ratings was returned.

Once the movie similarities had been found, in order to get a predicted rating for a user of a particular unseen movie, the following steps were taken:

1. The ratings the user has made are found.
2. The similarities between the unseen movie and the movies that user has rated are found.
3. The similarities are then scaled by dividing each similarity by the sum of themselves.
4. The scaled similarities are then multiplied with the ratings and summed.
5. This value was then returned as the predicted rating that a user would give that movie.
6. If there were no similarities the average of the user's ratings was returned.

To make a recommendation to the user the previous steps can be repeated for all unseen movies, and then ranked by predicted rating to find the predicted best picks.

The reason these two approaches added scaling into them was so that the scores they produced were between 0 and 5, and could be compared against the test set to find the RMSE.

4.3 Cosine similarity

The cosine similarity was used as the similarity metric to calculate the similarity between different users and movies. The cosine similarity can be calculated according to the equation below.

$$f(x,y) = \frac{\sum xy}{\sqrt{\sum x^2}\sqrt{\sum y^2}} \tag{1}$$

The equation was implemented in R with the function shown below - to account for the NAs in the data-set, the vectors within the summations in the denominator have to be referenced with the indexes which have values in the other vector. If this is not accounted for the equation will be dividing by a larger value than it's supposed to leading to some inaccuracy.

```
cosine_sim <- function(a, b){

    top <- sum(a*b, na.rm = TRUE)

    bottom_one <- sqrt(sum((a[!is.na(b)])^2, na.rm =TRUE)

    bottom_two <- sqrt(sum((b[!is.na(a)])^2, na.rm = TRUE)))

    return(top/(bottom_one*bottom_two))
}
```

To apply this operation across all the users by use of matrix multiplication, the following code snippet was used:

```
zeroed <- ifelse(is.na(train),0,train)
similarities <- tcrossprod(zeroed/sqrt(rowSums(zeroed^2)))
```

This made the simulation more computationally efficient; unfortunately this method had the inaccuracy of dividing by a larger denominator as described earlier. I could not find a way around this using matrix multiplication, and ended up using this method with some decrease in accuracy for the sake of computational efficiency.

4.4 K Nearest Neighbours

An addition that can be added to these methods would be the K nearest neighbours approach, this approach only considers the top k similar users, or items, when calculating the final recommendation score. This was not looked into for this assignment as the higher the k, the higher the accuracy. The extra accuracy in this case was worth the additional computational power needed.

5 Matrix Factorization Collaborative Filtering

5.1 Overview

Another way to do collaborative filtering is through a matrix factorization approach. This approach uses the training matrix to create two new smaller matrices, that when multiplied together replicate as close as possible the initial values in the matrix.

$$\begin{bmatrix} 0.2 & 0.3 & 0.1 & \dots & 0.3 \\ 0.1 & 0.2 & 0.5 & \dots & 0.7 \end{bmatrix}$$

$$\begin{bmatrix} 0.5 & 0.3 \\ \dots & \dots \\ \dots & \dots \\ 0.1 & 0.2 \end{bmatrix} \begin{bmatrix} 5 & NA & NA & \dots & 3 \\ NA & 2 & 5 & \dots & NA \\ \dots & \dots & \dots & \dots & \dots \\ NA & NA & 3 & \dots & NA \end{bmatrix}$$

Once these two matrices have been derived, they can be multiplied together in order to fill in the gaps where there are no ratings, and these can be used as the predictions of what users would have rated the movie.

5.2 Method

The matrix was factorized into two smaller matrices using the nnmf R package, which uses non-negative matrix factorization to find the optimal solution.

L2 regularization was added to the objective function being minimized(RMSE) to stop the matrices over-fitting to the training matrix. Bias values were also added to help reduce the effect of user's own biases towards the ratings.

Once the matrices were computed they were multiplied out and the RMSE of the of predicted ratings was found by comparing the predicted ratings to the ratings in the test set.

5.3 Choosing latent variable

The latent variable(k) is used to determine the size of the smaller matrices. The value of k in matrix factorization helps with the accuracy of the predictions, but there is a trade-off. The higher the value of k, the larger the computational time needed and greater the chance of over-fitting. But too small a k can lead to the model under-fitting and the accuracy of the results being sacrificed to some extent. In practice it would be best to work out a k value to be used based on the number of users in that iteration.

The value of k was determined by using 2500(middle ground) users and working out the RMSE over a range of values for k (2,4,6,8,10,12,14,16,18,20). Since the simulation was not practical to perform a grid search on, the lambda value was

held constant at 0.001. Once the k value was found it was then used throughout.

Figure 3 visually shows the investigation into finding the correct k. The simulation seemed to favor smaller values of k, this could be due to the small lambda value, as k increases the more chance of over-fitting. A k value of 6 was chosen as after this point the error started increasing more substantially. Even though 6 isn't the minimum RMSE, it gave more wiggle room for playing around with lambda value.

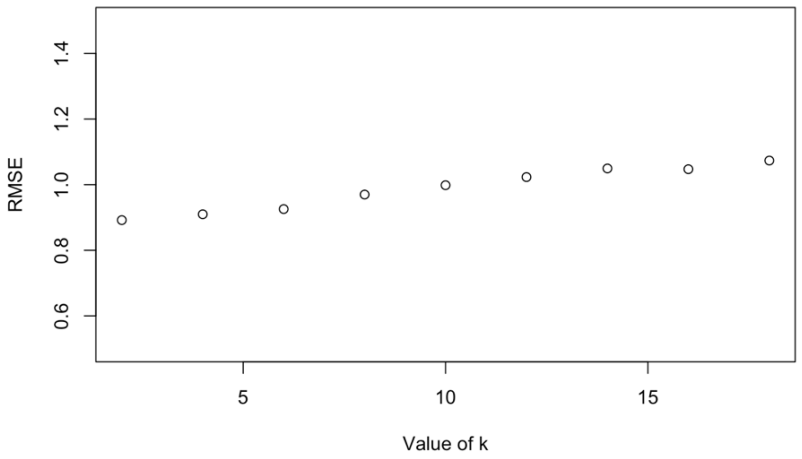

Figure 3: RMSE vs latent variable

5.4 Choosing Lambda

Only L2 regularization was explored. k was held constant at 6 whilst iterations were run to find the best lambda value. Lambda values explored were (0.00001,0.0001,0.001,0.05, 0.1,0.2, 0.3, 0.4, 0.5). The log of lambda was plotted for the purpose of easier visualization. The lambda value chosen was 0.4(log(0.4) = -0.3979).

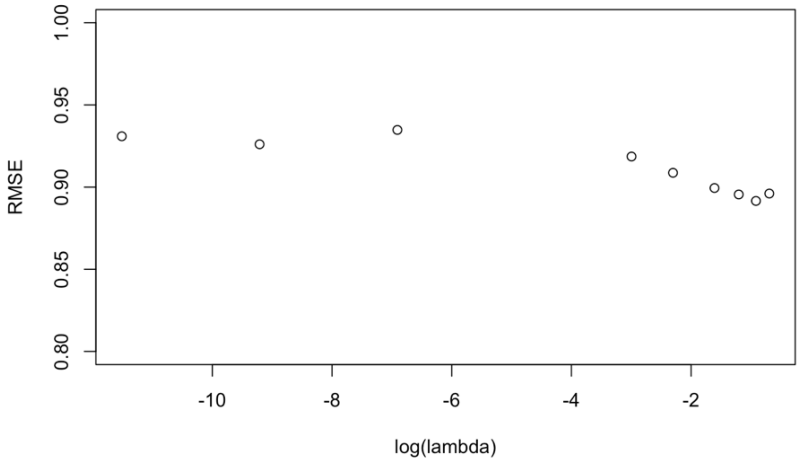

Figure 4: RMSE vs log of lambda

5.5 Capping the predictions

Since the matrix factorization was done without a constraint on the range the predictions should lie within, some of the predictions were above 5. This could throw off the accuracy of the ensemble as a whole. And so, if a prediction came back as greater than 5 it was then reduced back down to 5 and the assumption was made that the user would have given it the maximum rating.

6 Simulation

A simulation was run to plot the effect the number of users had on the accuracy of a recommendation. Instead of finding the highest recommended movie for each user, the simulation found the predicted rating for each rating in the test set. This gave the simulation a way of comparing inaccuracies.

The simulation was run on a different size samples of the users (10,50,100,150, 200, 300, 400, 500, 1000, 1500, 2000, 3000, 4000, 5000). For each of these sizes the simulation was run 4 times in order to observe the variation of the accuracy(RMSE) of the test observations.

The results for the user based, item based, matrix factorization and ensemble(average of the three) are stored at each iteration.

In the simulation, only ratings for the ratings in the test set were computed for increased efficiency. These predicted ratings were then compared to the ratings in the test set to find the RMSE

7 Results

7.1 Overall

Figure 5 highlights the comparison between the three described methods and the ensemble of the average of them all. From this figure it can be seen that overall the ensemble method outperforms the other methods. All of the methods' errors decrease as the number of users increases until 2000 users were it plateaus remains pretty constant from there on out.

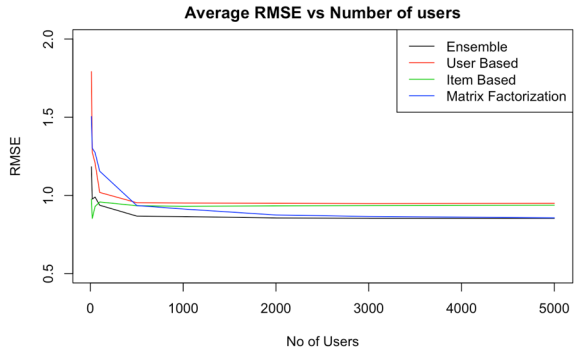

Figure 5: Average RMSE of all Methods

Figure 6 shows the variation per each number of users simulated, the x axis is not scaled to proportion for ease of visualization of the variation. For all the methods the variation of the results decreases as the number of users increases.

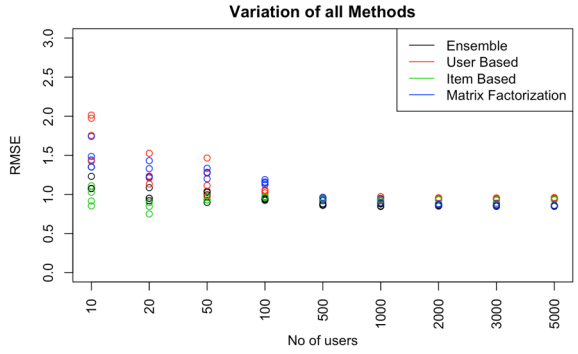

Figure 6: Variation of RMSE for all Methods

7.2 User-based

The user based method decreases exponentially until around 500 users were it flattens out and remains constant at around 0.95

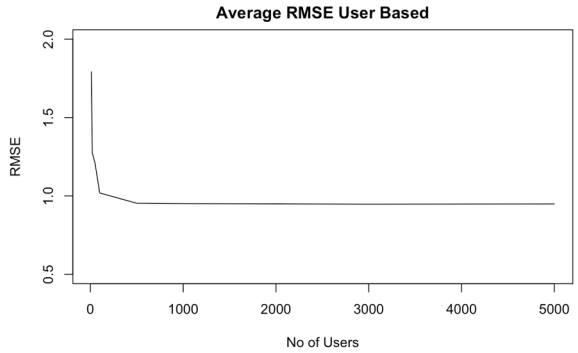

Figure 7: Average RMSE for user based method

The variation starts to converge at around 500 users, before then the variation decreases with the number of users.

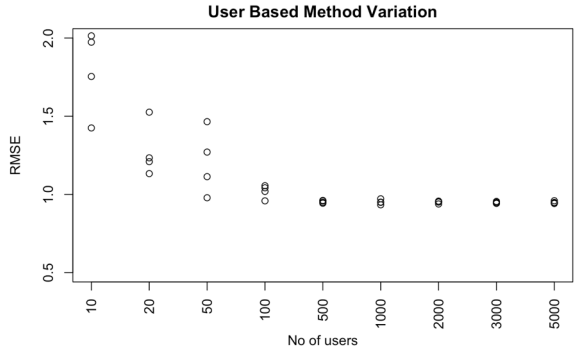

Figure 8: Variation of RMSE for user based method

7.3 Item-based

The item based method remains averagely fairly constant around 0.95 for all number of users, with slight fluctuations under 50 users

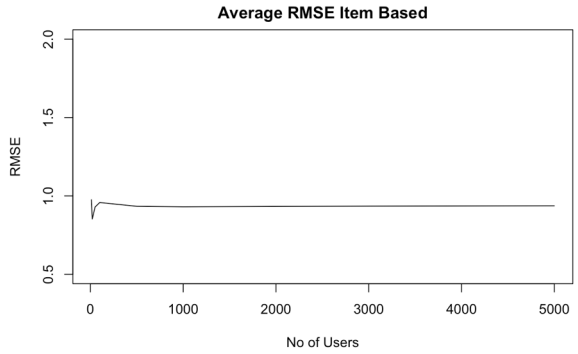

Figure 9: Average RMSE for item based

The variation starts to converge at 500, there is less variation than the user based approach.

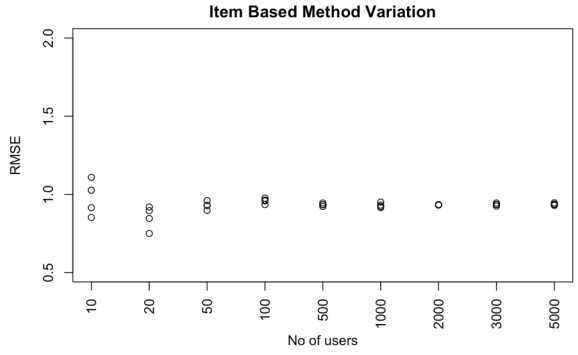

Figure 10: Variation of RMSE for item based

7.4 Matrix-Factorization

This method can be seen to improve with the more users used, it decreases exponentially and then plateaus at 0.87 around 3000 users.

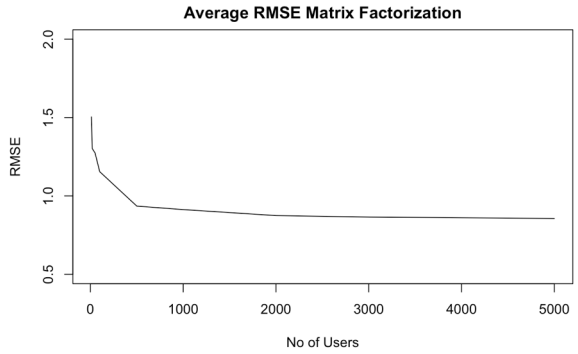

Figure 11: Average RMSE for matrix factorization

The variation decreases as the number of users increases, it starts to converge at 500.

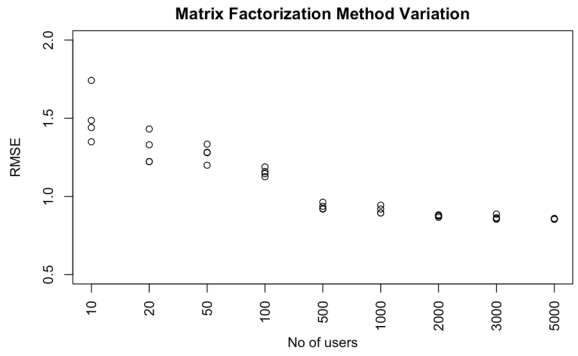

Figure 12: Variation of RMSE for matrix factorization

7.5 Ensemble

The ensemble method decreases as the number of users increases until around 2000 users where it flattens out. The curve at the beginning is brought through the user based and matrix factorization methods as the item based is constantly quite flat.

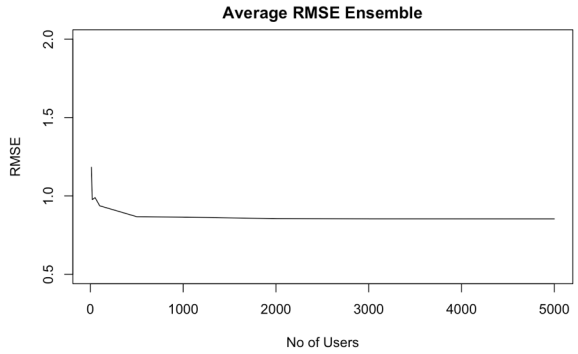

Figure 13: Average RMSE for ensemble method

The variation follows the same pattern as the three previous methods as expected.

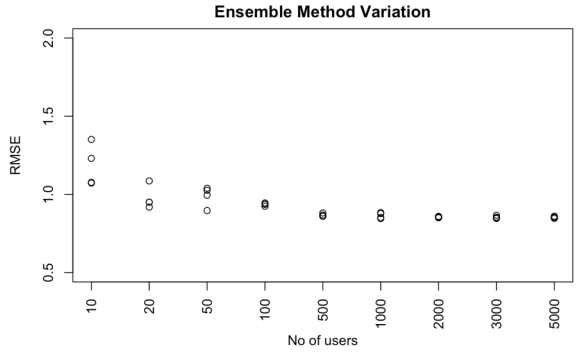

Figure 14: Variation of RMSE for ensemble method

8 Conclusion

Of the three main approaches item based performs the best for number of users under 500, whilst matrix factorization outperforms the other two for users over 500. Ensemble is the best performing overall as it encompasses the advantages of all the methods. Because the ensemble averages out over all three methods it also helps even out over/under predicting.

As for the business model of selling different recommendations based on different users the average error of the ensemble model for the users for the suggested 100,1000 and 5000 users are shown below

	Avg RMSE
100	0.9369173
1000	0.86454555
5000	0.85344955

With the ratings based on less than 500 users there is the added inaccuracy of individual recommendations due to the variation on the predictions.

The difference between 1000 and 5000 users is very small. And so I would suggest the company look into two different ratings one at 2000 users (increasing the number of users after this point hasn't much difference) and 100 users. The 100 user rating would have a larger error with an added uncertainty due to the variation.