University of Amsterdam

MSc. Econometrics

Quantitative Marketing

Assignment 2 - Recommendation Systems

Lucas Berndsen, 12337218
Sylvia Cheng, 12383708
Justyna Kozlowska, 12131199
Tom Schellings, 10774726

May 3, 2019

1 Introduction

This assignment will cover the topic 'Recommender systems' within quantitative marketing. A recommender system is an algorithm that adds value for both customers and providers. It helps a user to discover products and content by narrowing down the set of choices which consists of items that they would rate highly. This information is based on predictions of the user's rating of each item using different filtering techniques. For the providers, value is created since sales and customer satisfaction will increase if customers are able to find more efficiently what they are looking for. These recommender systems are very popular for many internet products: Spotify, Netflix and Amazon for instance, all rely on recommender systems to make the best recommendations for their customers by filtering available contents. When you visit Spotify, you will note that some songs are being recommended to you that you probably like or when you are searching for a specific book on Amazon, you will also be recommended to note some other interesting books. Thus, it can be concluded that recommender systems play a very important role in helping users to find the products they will probably be interested in. For this assignment, we will analyze the data set provided by Movielens which is a web-based recommender system and virtual community that recommends movies for its users to watch. The aim of this analysis is to predict unknown movie ratings based on Item-Based Nearest Neighborhood, User-Based Nearest Neighborhood and Matrix Factorization techniques.

Firstly, the dataset and the corresponding estimation and evaluation samples are explained. Thereafter, the theory behind the models will be discussed separately and it will be explained how each model is programmed in order to predict unknown ratings. The next section contains the results including the prediction accuracy based on Sum Squared Error and the calculation times to compare the models following the final conclusion with a discussion about the practical difficulties we encountered by implementing these recommendation techniques.

2 Data set

Before we can start we cleaned the data by removing all the non-values. This leaves us with 10677 movies and 69878 users. The data is split between a training and validation set. We use the training data to predict the rating a user would give in the test data on a movie. Consequently, we compare the actual rating with the predicted rating and take the squared differences for all ratings summed up to get the sum squared error (SSE) for each model as well as the meas squared error (MSE). The results section gives an overview of the SSE's, MSE's and calculation times for each model to get an idea of the models accuracy and practical usefulness. In order to be as effective as possible we want to be able to recommend the user quickly with a movie he probably likes most. Due to the computational difficulties for nearest neighborhood method, the test set only contains 100 values.

3 User-Based Nearest Neighborhood

We begin our analysis with the use of User-Based Nearest Neighborhood method. It is simply based on the similarity assumption between users. For instance, if person A and person B have watched the same movies, and they rated them similarly, we can easily assume, they are interested in the same movies. Therefore, when person A watches for example "Toy Story (1995)" and rates it very well, a user-based nearest neighbourhood method will recommend this movie to person B, if he or she hasn't watched it yet.
With the User-Based Nearest Neighborhood we calculate the similarity of user i with all the other users based on the ratings of movies. Then for each item j we construct a set of k users that are most similar to user i. Then we take the weighted average (weighted according to similarity measure) of the rating for this movie of these k users in this set as our predicted rating. Eventually you do this for all the movies and recommend the movie with the highest predicted rating.

The user-based nearest neighborhood method is implemented manually in python to predict 'unknown' ratings. It was not possible to use the build-in package for the nearest neighborhood methods due to memory errors caused by an extremely large similarity matrix between users. Therefore, all steps are programmed manually: for each user in the test set separately, the 'unknown' ratings will be predicted based on the training set. For example, for user i in the test set, the four nearest neighbors in the training set are found by the cosine similarity measure. Due to the large calculation time, we only calculated the similarity measure for users who have an overlap in movies. Thereafter, the rating for movie j will be predicted by averaging over the ratings for movie j for the four nearest neighbors. To compare the predicted and actual rating, we use the mean and sum squared error as a measure

which equals the squared value of the predicted rating minus the actual rating.
We used the cosine similarity measure instead of the pearson similarity measure because this was the easiest to implement in python and we found multiple examples of the use of the cosine measure on User-Based recommendations. We chose for 4 nearest neighbors by the eyeballing technique, meaning that for a small data set we checked which prediction gave tje smallest MSE, we could have checked this by cross vallidation but due to the computational time, this was not a computer friendly option.

4 Item-Based Nearest Neighborhood

For the second method we used Item-Based Nearest Neighborhood as recommendation system. This method is characterized by looking at the similarity between items using the rating by each user. For each item we calculate the similarity with the other items based on the cosine measure, but only if they have users in common. For instance, 'Toy Story (1995)' has three nearest neighbors which are 'The Ballad Of Little Jo (1993)', 'Private Lessons (1981)' and 'Willy Wonka the Chocolate Factory (1971)'. After that we obtain a set of the k most similar items. Then we take the weighted average (weighted according to similarity measure) of the rating for this movie of these k users in this set as our predicted rating and recommend the item with the highest rating. The programming part is very similar to the User-Based Nearest Neighborhood, however, the main difference is that we loop through items instead of users.

We see that item-based is quite similar to used based nearest neighborhood. However, with item-based we look at the similarity between items (movies) and with with used-based we look at the similarity between users. The similarity measure used for the item-based method is the cosine measurement and we used the 4 nearest neighbors as well regarding the same argument.

5 Matrix Factorization

Last but not least, we will use the Matrix Factorization method to predict unknown ratings. One problem that nearest neighborhood method suffers from, is the scarcity of rating matrix. For example, if one person watches Jaws, directed by Steven Spielberg and another person watches Jurassic Park, also directed by Steven Spielberg: Nearest neighborhood will not distinguish the similarity between both users, because they won't have any ratings in common.

The matrix factorization method can recognize unobserved factors, so called latent factors, which are constructed based on "shared variance". As the number of latent factors increases, the number of parameters increases relatively rapidly. Therefore, we cannot choose the number of latent factors to be too big. In our case we have chosen the number of latent factors equal to 10, which still leaded to a reasonable calculation time.

We began matrix factorization with setting the column *userid* as an index, and a column with movie titles as a column in a new rating matrix. Thereafter, we continued with the data split, where train and test set were of a size 50% each. As in matrix factorization the users and items are represented in a lower dimensional latent space, we make use of the singular value decomposition (SVD) which decomposes the rating matrix into another matrix. In order to properly use SVD, we have to normalize the rating matrix. Therefore, we demeaned the data both in train and test set. That matrix is an approximation of the rating matrix, but has a lower rank which is helpful because we have a large data set. We only keep the k most important taste and preference vectors.

6 Results and conclusion

Table 1: Comparison of the models

Model	prediction accuracy (SSE)	MSE	calculation time (seconds)
Item-Based Nearest Neighborhood	68116	0.97	4081
User-Based Nearest Neighborhood	10739	1.03	3726
Matrix factorization	44646439	767.18	45

The above table, illustrates the most important results. Please note that the item-based and user-baser methods are using a lot more computer power. Therefore, we made 100 predictions using the item-based and user-based methods (the other data points are used for training purposes). For the matrix factorization we used a 50 % data split. Therefore, it does not make sense so compare the SSE between all three models. Since the MSE takes the amount of observations into account we also provided that. Between item-based and user-based, the item-based is performing the best (based on the value of the MSE). However, the SSE of the user-based is much lower which is as expected since we predict less items compared to the user-based method. Since we normalized the data for matrix factorization, the SSE and MSE cannot be properly compared to the user-based and item-based.

The theoretical framework for the item-based and user-based methods may look very simple and intuitive. On the contrary, matrix factorization seems more complex and less intuitive. However, the Matrix factorization is a lot faster to run, since it was able to make 5 million predictions. Both item-based and user-based have the benefit of capturing features such as taste. Nevertheless, they suffer from large computation times with big data sets such as the one in this assignment, largely due to the heavy similarity calculations between items and users. We encountered difficulties when using built in functions for our predictions. They were not designed for data sets this large.