DAT410 - Assignment 2

Eric Johansson & Max Sonnelid

February 1, 2021

Group number: 70	
Module number: 2	
Eric Johansson	**Max Sonnelid**
970429-2854	960528-5379
MPDSC	MPDSC
johaeric@student.chalmers.se	max.sonnelid@gmail.com
15 hours spent	15 hours spent

We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part of other solutions.

Introduction

This assignment aims to develop knowledge about recommendation systems as well as why and how they are useful. This project has been executed by Eric Johansson and Max Sonnelid. Summaries of associated lectures toghether with personal reflections will be attached in the end of this document.

1 Reflections about "The Netflix Prize" and "Netflix Prize Challenge"

1.1 Max Sonnelid

Personally, I have thought that data-based recommender system is a rather new invention, but it was surprising to read in "The Netflix Prize" that Netflix has both collected and saved data about its customers from 1998 and on. The decision to release this massive data set that had been collected over several years and make a competition about optimizing a recommending system is an excellent example of open innovation. As all the innovation created in the frames of this competition had to be made public, all the innovation is also free to use for the public and seems to be an important origin of many later innovations in the recommending systems. At first glance, it might seem that Netflix is losing money as the innovations are made public, including to Netflix's competitors. However, one may compare the cost for awarding a prize of $1 million and the alternative cost for establishing and developing an advanced RD department. The latter alternative is probably a lot less cost-efficient than the first one, and Netflix only awarded the price, when the teams were successful, which reduces the financial risk for them.

Furthermore, it was interesting to read in "Netflix Prize Challenge" that a single prediction model probably is rather bad at predicting all relevant aspects in a recommending system and that ensemble models are more successful in almost all practical cases. In this case, the neighbourhood models were excellent at finding movies that are closely related to each other, while the latent factor models were better at estimating the more general relationships between all movies for a certain user. Realizing that the combination of two different ML-models performs better than the two individual ones is valuable to have in mind when developing other AI-systems.

One design feature that was especially interesting was the so called binary viewpoint, where data such as rental history and browsing history is taken advantage of. In many settings for a recommendation system, this kind of data is probably the only one that is accessible and thus the only data to make predictions from. A good example of that would be grocery stores that have a lot of binary information, e.g. what products customers have bought and when they have bought them, but

probably no rating information at all. For the grocery store, it is crucial to be able to transform this binary data in many different ways in order to make accurate recommendations.

Another interesting design feature was the use of regularization for avoiding overfitting. With a small amount of data available relatively to all possible relations between users and products, it is very easy to draw too strong conclusions from the available data and make irrelevant recommendations to the user. Using the grocery store example again, it is probably less important to use regularization there as many customers tend to buy the same products over time and are thus happy to get personalized recommendations and discounts on the same products that they have bought in the past. Compare this to a movie recommending system, where the customers constantly want new kind of recommendations and then it probably hurts the recommendations more with overfitting.

1.2 Eric Johansson

1.2.1 Takeaway from articles

I have read two articles called *The Netflix Price* and *Lessons from the Netflix Prize Challenge*. The first paper discusses a competition that Netflix issued in 2006. To describe it briefly, people got access to their data about regarding ratings that different user gave to their movies in order to improve their recommendation algorithm (this is measured by measuring the root mean squared error). The data, as well as submitted suggestions of improvements from winners were available to all contestants which was a smart move by Netflix in order to get better and better submissions. In the end of the article, a graph of multiple submission were displayed, together with the score that different models yielded. It was interesting to see how, for example, just using movies average rating and using Bayes formula compared to each other. Reading from the graph, it seemed like SVD was a successful method to get good results.

The second article is written by the team that (by the time of writing the article) had yielded the biggest improvement of the recommendation system. They first explained their solution in a conceptual manner and then went into more depth of how their solution was built. My biggest takeaway from the article was that the winning team not only used an ensemble model to get good results, but also put a lot of thought into what models they should combine. They explained it as have a model with three axes, where each axis addresses different aspects of the data. It was interesting to see how different statistical models could be use to address different problems. For example, kNN was used to find ratings of movies closely related to the movie that was about to be rated, and then Boltzmann machines were used to estimate the overall structure of the data. To put it more general, different statistical models can complement each other in a good way if the programmer knows enough about what the models actually do. It was also interesting to read about how important it is to understand the statistical models in order to tweak their parameters til perfection. Finally, reading the discussion of the last paper, very useful tools were presented to optimize a real life model if even more data was accessible (such as rental history, browsing patterns and keywords).

1.2.2 Design features that characterizes the solution

Pick two design features (e.g., regarding data, models, algorithms) that characterise the task and the solution in the papers and discuss how they may differ in another application of recommendation systems.

Singular Value Decomposition One key contributor to the AT&T-team success was that they were using the SVD in order to extract more information about the viewers and the characteristics of the movies. By using SVD they were able to identify different concepts or genres of each movie, as well as the preferences of each user. One could also use the SVD to figure out what features of a movie that is important and what are not, in relation to each other. I think that one reason why SVD were so successful is that movies are generally easy to group in different genres. If a company that sells products that are harder to differentiate, using an SVD might not be as successful since the model would still split the available products into different categories, even if such categories does not exists.

Regularization Regularization is another aspect that the winning team emphasizes. The idea of regularization is adding an extra term to the minimizing function in order to prevent the Θ to contain large values - mitigate the risk of over fitting the data. This is especially important in the Netflix case since only a small percentage of the combinations of movies and users actually had

ratings (around 1%). For data sets that contain a higher share of ratings, regularization might not be as important (yet not unimportant).

2 Creating a recommender system

After some brief research, we came to the conclusion that building a proper model (as the one described in the article) would take a very long time (a few years for the winning team), Hence we tried two smaller versions of a recommender system, the first one by using correlation. After comparing different movies based on their correlation, we also wanted to try something that was mentioned in the articles. In both cases, we found it very challenging to use both data set right from the start. Instead, we saw this as a great opportunity to learn a lot of python by combining our previous knowledge about machine learning and statistical methods to code our methods from scratch instead of following an advanced example found online.

After some quick research on the Internet, we saw several examples of recommender systems only focusing on user reviews and got inspired by them. Therefore, we chose to focus only on the `user_reviews.csv` data set and thus base the recommender system on a *collaborative filtering method*.

2.1 Correlation method

The recommender system, based on the correlation method, was implemented in the following steps:

1. Firstly, a new DataFrame was created called `tot` with the following columns: movie name, sum of ratings, number of rating and average rating (sum of ratings divided by number of ratings; thus only divided by actual ratings done).
2. Secondly, the method `movies_watched` was created, which takes the user index in `user_reviews` as input and returns a DataFrame with all movies rated with a value of 3, 4 or 5 by this user. The columns are movie name and average rating.
3. Thirdly, the method `correlated_movies` was created, which takes a movie name as input. Then the correlation coefficient between movie ratings from this specific movie and movie ratings for all other movies in the data set was calculated. A DataFrame with the movies that are positively correlated with the input movie is returned.
4. Fourthly, the method `total_movie_value` was created, which takes a list of movies (created from `movies_watched`) as input. For each movie in that list, the `correlated_movies` method is run. For each movie in the returned correlation DataFrame, the movie (if not already present) is added to an array together with the corresponding value Q, or (if already present) only the corresponding value Q is added to the already existing value corresponding to the movie. This value Q is defined as the product of the correlation coefficient for the current movie and the rating of the corresponding movie in the input movie list. Movies that are present in the input movie list are not added to the array. Returned is a sorted DataFrame in descending order with columns movie name and value.
5. Fifthly, the five movies with the highest sums of values calculated in step 5 are recommended to the user.

2.2 kNN-method

After some discussion, we decided on trying out the kNN-method, which is something that we have tried in previous classes. Here, we only used the data set "user_reviews.csv" and thus ignored the genres of the movies given by the other data set. To use the user reviews set, we first deleted all the unnecessary columns that didn't brought any value to our method (e.g, name and unnamed). We then transformed the dataFrame into a matrix in order to train our kNN-model with it. After some research, we found that 'cosine' seemed to be the optimal metric and 'brute' was the easiest algorithm to use. One problem of using the brute-algorithm is that it loops through all combinations of movies, and thus can be very time consuming. However, we think that it yields a quite accurate result and thus stuck with that choice.

After that, we made a method that takes a movie and n_suggestions as input and gives (by using our kNN model) two lists that are of the size n, the first one containing the suggested movies and the

other one their associated distance (which could be seen as score where 0 is the best). To combine this score with the rating, we used a method that divides that rating with the distance.

After this step, we used the same approach as described above to loop through all the movies a user has rated and to combine these recommended movies into a long list, as well as deleting the movies that the users already have seen.

2.3 Strengths and weaknesses of the approach

A clear strength of our approach was that we used two different methods to suggest movies. However, we found it hard to create a test set since very few of the user-movies combinations actually had ratings. Thus, it was hard for us to know which method that worked the best. It would therefore be interesting to use this idea on a test set and then draw conclusions out of the results.

Another strength was that there is a high probability that suggested movies actually are quite similar since both the kNN and correlation method looks closely related movies. What is bad, however, is that only using this method can make the model overfitted. Suppose that a movie does not actually relate with any other movie. The kNN-method will still recommend the k movies that are closest. To reduce the risk of overfitting, we could have use a regularizer. Unfortunately we didn't know how to implement this to our model. Moreover, we could have put more thought into our choice of hyper-parameters (such as n_neighbors to get n suggestions per movie) but once again, we did not have any set to test our models on.

Some final clear disadvantages of our model was in the stage where we combine ratings with our yielded distance/correlation. Here, we only used ratings that where higher than 2 (i,e. 3,4, or 5) in order to only consider movies that the user seems to like. Here, we also wanted to give a negative value or in some way punish movies that got a low rating instead of just ignoring them. This is something that we would have fixed in a better model. Moreover, since our model could calculate correlation between different movies in the user_rating set, the same method could possibly be applied on the other data set as well.

We are very aware of that our model contains more flaws than described above, but these are some examples of obvious things that we would have corrected if we had more time and knowledge. This was a very fun and challenging assignment.

2.4 Recommended movies by the correlation method

Vincent	**Edgar**	**Addilyn**
Tomorrowland	The Hurt Locker	The Woman in Black
American Hero	The Blues Brothers	The Messengers
Das Boot	Me and Orson Welles	The Contender
The Hundred-Foot Journey	The Wash	The Work and the Glory II
The Matrix	The Twilight Saga: Eclipse	The Chumscrubber
Marlee	**Javier**	
Das Boot	The Work and the Glory II	
Licence to Kill	Nine Queens	
The Matrix	The Express	
The Blues Brothers	Zipper	
The Last Exorcism	Deep Rising	

2.5 Recommended movies by the kNN-method

Vincent	Edgar	Addilyn
Trainwreck	The Avengers	The Big Wedding
Me Before You	Flicka	Sorority Boys
Paint Your Wagon	The Wash	Supporting Characters
The Finest Hours	The Island of Dr. Moreau	The Outrageous Sophie Tucker
Boyz n the Hood	Night of the Living Dead	12 Rounds
Marlee	**Javier**	
Tarnation	My Favorite Martian	
Back to the Future Part II	Nothing	
Hot Pursuit	Blazing Saddles	
Point Blank	Perfume: The Story of a Murderer	
August Rush	Song One	

3 General discussion about recommender systems

Assessing the quality of a recommendation system before deploying it to users is difficult. Why? In a few paragraphs, discuss fundamental challenges in evaluation of recommendation systems and how they may lead to problems in practice.

One reason that the assessment of the recommendation system is hard to do before deploying is that personal taste is a very subjective matter. Without very much data, it can be hard to know what patterns that is discover through overfitting and what patterns that actually can be applied to all "similar" users.

Another reason why it is hard to assess is that people tend to look for different movies dependent on which "state" they are in. For example, if I am by myself on a Monday night I might look for something easy digestible to relax to whereas if I am with my friends on Saturday evening I might look for something else. This would be hard to take into account if ratings is the only available data.

Thirdly, in the data we were given as well as in the data in the article, a very low amount of user-movie combinations actually contains ratings. Hence, a system that would predict that each movie should get a rating of 0 would actually get quite high accuracy if it was tested on a similar data set. In the case of Netflix, this could probably be solved since they have so much data. However, for a newly started service, such as Disney +, it might be harder.

Fourthly, an interesting problem to tackle is that everyone is probably not giving their explicit, honest opinion about products. For example, some users can be expected to give higher ratings to movies that are commonly considered to be good and thus "should" be rated high, e.g. the Godfather, even though the user did not like the movie particularly much in reality. On the other hand, users can be expected to give lower ratings to movies that are commonly considered to be bad and thus "should" be rated low, e.g. Ex on the beach. To solve this problem, the recommender system probably has to take implicit information into account such as how many times the user watches the movies, how long the user stays at a movie et cetera.

Finally, before one has tested the recommender system, it is hard to know how it is going to work in reality. Probably some validation of the efficiency of the system has been done, but the real and most valuable validation is the real-life testing of the system on the group that is actually going to use the system. When launched, it is possible to observe how the users actually are interacting with the system such as whether people are watching the recommendations and how long they stay at the recommendations. Taking continuous and long-term feedback from the system is probably the only way to create a recommender system that is adapted to real and honest human needs.

4 Summary of lectures (Lecture 2)

4.1 Max Sonnelid

- Recommender systems are a vital part of the business model for companies as Netflix, Amazon, LinkedIn and Spotify. E.g. Netflix's recommender system was estimated to be worth $1 billion per year already in 2015.
- In an *uniform strategy*, the recommender system displays books that sell well overall or are on sale. However, it is hard to know whether each individual user is likely to buy the recommended

book.

- In a *personalized policy*, each user is assigned unique and own recommendations. For such a policy to work, passive data, e.g. purchase logs, is probably the best information source. Based on the passive data, it can be possible to predict future purchases.
- It is important to state the *objective* of a recommender system, which should be the subject of evaluation. Objectives can be both short-term (e.g. whether one purchase is made or not) and long-term (e.g. whether profit increases over several months). Long-term objectives are harder to optimize.
- Often proxy variables are used for predicting a higher level variable. E.g. ratings can be a proxy for purchase. Ratings is a widely used data variable in recommender systems.
- Mathematically, a personalized recommender system is the search for a personal mapping between a product and a rating. This personal mapping can be approximated by machine learning as a linear function.
- Optimizing the personal recommender system can be translated to minimize the squared error of these linear functions.
- A major problem in recommender systems is the big lack of data and therefore regularization comes as a solution to the overfitting problem.
- *Content filtering* is based on user-product and product-feature data, while *collaborative filtering* only uses user-product data.
- A way of measuring the effectiveness of a recommender system is *A/B experiments*.

4.2 Eric Johansson

Recommender systems is one of the most common AI systems. It is frequently used to serve adds and ranks posts (e.g., Netflix, Spotify or Facebook adds). Very important for creating revenue (Netflix recommendation is worth $1billion each year).

In the lecture, this was examplified by selling books. There are different strategies to use when recommending new books for visitors.

- Uniform strategy - Display most popular books
- Personalized policy - Categorize people and find personalized recommendations

The latter one is an example of a recommendation system.

To asses the recommendation system, there has to be an objective, i.e, a way of measuring the success. In the given example, the original objective was profit. To facilitate the quantification of the objective, it was initially changed to nr. of users, followed by satisfaction or engagement and finally recommendations (or even prediction of recommendations). These changes of objectives is called proxies.

To learn the model to predict ratings, an approach called collaborative filtering is often used. To do this, you only need users and books and by splitting up the matrix of these categories into two smaller matrix (this is called matrix decomposition), one can predict what ratings new movies would get by different users.

5 Reflection on the previous module

5.1 Max Sonnelid

Especially the first part of module 1 was a fun and interesting assignment. In most assignments in previous IT courses, the focus has been on either the implementation or a very detailed modelling a the problem. This time, it was fun that the assignment was about thinking very freely about the modelling and potential solution approaches to several problems. In a university setting, problems are often already well-defined and used as a means for implementing specific solution approaches described in lecture. However, in a real-life setting, a big issue is defining the problem and find an efficient and suitable solution approach before the actual implementation. It would probably benefit my future problem solving to train on this more and possibly also learn about frameworks both for

problem defining and finding solution approaches. I hope that the future tasks will have some focus on this as think it is a valuable skill to develop.

The latter part of the assignment, to learn about constraint satisfaction problems and constraint programming, was not as developing as the first part. This part mostly consisted of summarizing information without reflecting too much. However, in order to make this part more interesting, we could have put more focus to talk about applications in AI of these two themes and potential advantages and disadvantages. Next time we get a similar assignment, we will try to do this.

5.2 Eric Johansson

The first assignment did mostly consists of a lot of discussion. Moreover, it was relatively high level - a lot of answers were more conceptual rather than in depth descriptions of how to implement the solutions. I think that this was good in the sense that you got a good overview of different ways to approach regular AI-related problems. An important insight was that machine learning or advanced statistical methods is not always the best solution. Sometimes, a simulation of linear model could be just a efficient and sometimes even better.

After discussing our answers with other, I discovered that all had thought different. I do not think that anyone was wrong, but instead saw it as an example that almost all problems can be approach in multiple ways.

This was an interesting assignment, and I am now looking forward to learn more about more in depth solutions to these problems.