Technische Universiteit Delft
Faculty of Electrical Engineering, Mathematics and Computer Science

Netflix Challenge: Movie Rating Prediction

CSE-2525 Data Mining
Thomas Abeel, Gosia Migut

Prepared by
Yanqing Wu
Student ID 5142571
Kaggle ID yanqingwutudelft
Exchanged Computer Engineering
2 January 2020

Table of Contents

List of Figures iii
List of Tables iv
1 Introduction 1
  1.1 Netflix Datasets 1
2 Methodology 1
  2.1 Item-Item Collaborative Filtering 2
    2.1.1 Core Equation 2
    2.1.2 Nearest Neighbors 2
3 Results 3
  3.1 Local *Cap.* 3
  3.2 Selectively Rounding 3
  3.3 Dynamic *N* 4
  3.4 Local Deviation 4
4 Discussion 5
  4.1 Discussion on The Results 5
  4.2 Observation on Local Deviation 5
  4.3 Bigger Training Set 6
5 Conclusions and Future Works 7
  5.1 Conclusions 7
  5.2 Future Works 7
References 8

List of Figures

Figure 4-1 Year-of-release vs. Ratings 6

List of Tables

Table 1-1 The Basic Information of Provided Data Sets 1
Table 3-1 *N* and *Cap.* vs. RMSE 3
Table 3-2 Selectively Rounding vs. RMSE 3
Table 3-3 Dynamic *N* vs. RMSE 4
Table 3-4 Local Deviation vs. RMSE 4

1 Introduction

The report, entitled ‘‘Netflix Challenge: Movie Rating Prediction’’, is prepared as my Challenge report for the course CSE2525-Data Mining at the Technische Universiteit Delft. The purpose of this report is to develop a recommendation system for predicting movie ratings. The goal of the recommendation system is to achieve Root Mean Square Error (RMSE) as small as possible on an unseen dataset.

1.1 Netflix Datasets

Table 1-1. The Basic Information of Provided Data Sets

Dataset	Fields	Mean	Std	Min & Max
users	gender	0.72	0.45	0.00 1.00
	age	30.64	12.90	1.00 56.00
	profession	8.15	6.33	0.00 20.00
movies	year	1985.81	16.91	1919.00 2000.00
	title (string)	-	-	-
ratings	rating	3.58	1.12	1.00 5.00

In users - ‘gender’, ‘0’ and ‘1’ indicates female users and male users, respectively;
In movies - ‘year’, only non-zero entries are considered.

Three datasets are provided for training, as described in Table 1-1. There is a total of 910,190 ratings, which were given by 6,040 users and 3,706 movies. The rest ‘predictions.csv’ file is used for final testing, which contains only ‘userID‘ and ‘movieID’ for each entry. There is a total of 90,019 entries in ‘predictions.csv’ to be predicted and the result is put on Kaggle to determine RMSE score.

2 Methodology

Data interpretation was first conducted at the beginning of the Challenge, as demonstrated in Table 1-1. After examining the data and reviewing the course materials, the first version of the recommendation algorithm was decided as Collaborative Filtering (CF). More specifically, Item-Item CF was selected as the first attempt to the Challenge. Item-Item CF is more reliable than User-User CF in practice [2], because items are much less dynamic than users.

2.1 Item-Item Collaborative Filtering

Collaborative filtering is an approach of ''making automatic predictions (filtering) about the interest of a user by collecting preferences information from many users (collaborating)'' [1]. There are usually two approaches for collaborative filtering (CF), namely Item-Item CF and User-User CF. User-User CF is generally more difficult to scale than Item-Item CF due to the dynamic nature of users, whereas items usually remains constant. Hence, Item-item CF was selected.

2.1.1 Core Equation

$$\begin{aligned} r_{xi} &= b_{xi} + \frac{\sum_{j \in N(i;x)} S_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} S_{ij}} \\ b_{xi} &= \mu + b_x + b_i \end{aligned} \tag{1}$$

Equation (1) shows the core idea of the algorithm [3]. Term r_{xi} represents the rating of user x on movie i. Term s_{ij} is the similarity of movie i and movie j, which is measured by centered cosine similarity. Term b_{xi} is the baseline estimator for r_{xi}, where μ is the average ratings of all movies, b_x is the rating deviation of user x and b_i is the rating deviation of movie i.

2.1.2 Nearest Neighbors

For each prediction, the top similar movies to movie i need to be determined, so that a more reasonable prediction can be made. The top similar movies to movie i is known as the *nearest neighbors* of item i. A parameter N is used to denote the amount of nearest neighbors in the following context. Additionally, Jaccard similarity is abandoned in determining similarity since it only considers common movies rather than ratings. Moreover, Cosine similarity has a more informed interpretation than Jaccard similarity; however, it fails to generalize among users since different users have different standards. Pearson correlation (a.k.a centered cosine similarity) preserves the advantage of cosine similarity and removes the biased views of users via subtracting row mean for each movie.

An upper triangular matrix of similarity score is built to fulfill all aforementioned points.

3 Results

This section shows the Root Mean Square Error (RMSE) of the Item-Item Collaborative Filtering system on Kaggle test set. In addition to *N*, some extra options and parameters are introduced to improve RMSE: *Cap.*, ε, L_x and Local Deviation.

These are explained in the following sections, along with their test results.

3.1 Local *Cap.*

Table 3-1. *N* and *Cap.* vs. RMSE

N	*Cap.*	RMSE
5	off	0.88071
7	off	0.86398
10	off	0.85458
10	on	0.85397
12	on	0.85144
15	on	0.84996

Table 3-1 shows the result of the recommendation system (described in 2.1) on Kaggle test set. As mentioned in 2.1.2, *N* is the amount of nearest neighbors.

After noticing some negative ratings and nearly 6.0 ratings, option *Cap.* is introduced. The *Cap.* option sets ratings that below 1.0 to 1.0 and ratings that above 5.0 to 5.0.

The maximum static *N* is set to 15, because the minimum amount of user *x* rated movies is 15.

3.2 Selectively Rounding

Table 3-2. Selectively Rounding vs. RMSE (*Cap.* = on)

N	ε	RMSE
10	0.1	0.85458
10	0.05	0.85409
10	0.02	0.85398
10	0.01	0.85398
10	0	0.85397

Table 3-2 shows an attempt to use rounding to improve RMSE. Hyper-parameter ε represents the range of rounding. For instance, $\varepsilon = 0.05$ rounds any ratings in the range of $[3.95, 4.05]$ to a 4.0.

3.3 Dynamic *N*

Table 3-3. Dynamic *N* vs. RMSE (*Cap.* = on)

N	RMSE
L_x	0.87736
min(round(0.1L_x), 15)	0.85761
min(round(0.2L_x), 20)	0.85280
min(round(0.3L_x), 30)	0.85307
max(round(0.1L_x), 5)	0.85754
max(round(0.2L_x), 5)	0.86148

Table 3-3 shows an attempt to use dynamic *N* to improve RMSE. Parameter L_x is the amount of user *x* rated movies. Given the fact that L_x ranges from 15 to 2,097, a dynamic way of choosing *N* is introduced to avoid overfitting or underfitting.

3.4 Local Deviation

Table 3-4. Local Deviation vs. RMSE (*Cap.* = on)

N	Dvi.	RMSE
min(round(0.2L_x), 20)	-	0.85280
min(round(0.2L_x), 20)	exception users	0.85275
min(round(0.2L_x), 20)	all users	0.90199

Table 3-4 shows an attempt to use local deviation to improve RMSE. Local deviation is introduced to address cold-start problem (i.e., when there is insufficient information to make an decision). Local deviation takes user’s gender, age, profession and movie’s year-of-release into consideration.

4 Discussion

4.1 Discussion on The Results

A general trend of N vs. RMSE can be observed in Table 3-1, RMSE drops when N increases. This result is desired, since the more relative information is taken during training, the more accurate prediction will be. Moreover, RMSE drops dramatically when N is small; RMSE still drops when N is above a certain threshold (10 here), but not as obvious as a small N. This result makes sense, because there is a diminishing return between the amount of information and the final prediction. Furthermore, option *Cap.* improved the result due to the elimination of out-of-range ratings.

As demonstrated in Table 3-2, Seletively Rounding ε does not help minimizing RMSE. Since the final rating is an integer, the author had a belief that rounding those almost certain ratings to their nearest integers would enhance the final result. Nevertheless, the result did not improve. The author concludes that there might not be as many worth-rounding ratings as expected in the datasets.

As shown in Table 3-3, Dynamic N did not improve RMSE. Although dynamic N did not achieve a lower RMSE, the author believes that using a dynamic N is more reliable than a static N in terms of avoid overfitting or underfitting.

The last attempt of minimizing RMSE was using local deviation to address cold-start problem. Before using local deviation, new users or new movies or a combination of both were given the global average rating. After using local deviation, the RMSE improved from 0.85280 to 0.85275 while kept other parameters constant (see Table 3-4).

4.2 Observation on Local Deviation

Some interesting facts were discovered while calculating local deviations.

1. Old movies have higher average ratings. Female or elder users give higher average ratings.

Some possible interpretations to these phenomenons are that (1) people are more strict with new movies than old movies (see Figure 4-1); (2) people are usually nostalgia; (3) female users are more emotional than male users; (4) elder users are more generous on giving high ratings than young users.

2. Some users are too lazy to change default profiles.

Around 200 users (out of 3,706 users) have $age = 1$ and $profession = 0$. These users are considered

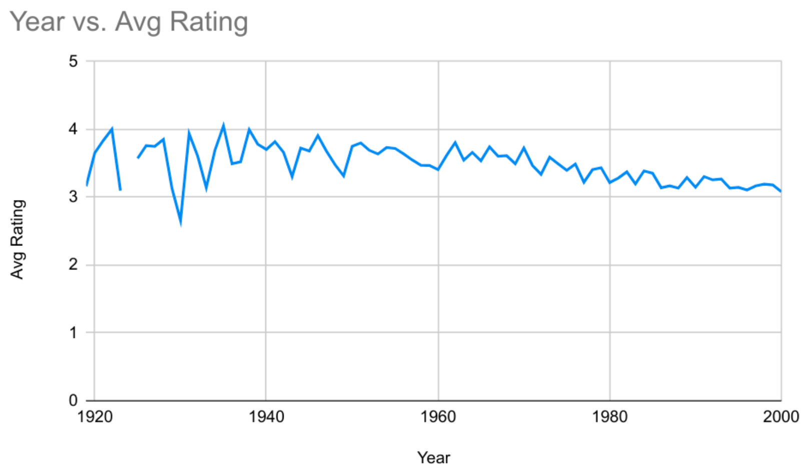

Figure 4-1. Year-of-release vs. Ratings

as abnormal data entries, which are ignored in the calculation of local deviations.

4.3 Bigger Training Set

Since there is only a limited amount of training data, an attempt of increasing training dataset was conducted. More specifically, a new rating dataset was produced by adding the least RMSE submission to existing rating dataset. As a result, the size of the rating dataset was increased 9.9% to a total of 1,000,209 ('ratings.csv' 910,190 + 'submission.csv' 90,019) entries. However, a re-training and re-submission on the new rating dataset did not further minimize RMSE. The result was near 0.858, which was 0.004 higher than then optimal submission.

Although the result failed to meet expectation, the author still believes that an iterative training (i.e. update training set with the least RMSE submission) may enhance the result eventually. Due to a lack of computational resources, the author cannot afford such experiment.

5 Conclusions and Future Works

5.1 Conclusions

In this report, the objective was met. An Item-Item Collaborative Filtering recommendation system was successfully developed, achieving a minimum RMSE of 0.84991 on the Kaggle public score board. The important conclusions that can be drawn from the report are included below.

1. Larger *N* improves RMSE, but there is a diminishing return.

2. It is critical to address cold-start problem.

3. Rounding is a bad idea.

5.2 Future Works

Some potential steps for future improvement of the work started by this report have been identified.

1. Develop local RMSE test.

Developing own local RMSE test will save a huge amount of time resource. The ratio of splitting limited training set into a new training set and own test set needs to be carefully chosen.

2. Take movie names into consideration.

In the future, one may use techniques (such as TFIDF) to find correlation of movie names and ratings. This may help enhancing the RMSE result.

3. Incorporate latent factor models.

Combining latent factors can further reduce RMSE, help achieving more accurate result.

References

[1] *Collaborative filtering*. 2019. URL: `https://en.wikipedia.org/wiki/Collaborative_filtering`.

[2] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets*. New York: Cambridge University Press, 2009.

[3] *Mining of Massive Datasets*. URL: `http://mmds.org/`.