The Netflix Prize Solution

Authors
Group 4
Prashan Sengo
Manishka Mathur
James Biondi

1 Background/Introduction

Netflix has a lot of movies and millions of users that consume their content. Whenever a user logs into Netflix to watch a show or a movie, that activity is logged by Netflix to enhance their experience. Netflix as a business wants to engage their audience and enhance their experience by providing more accurate recommendations to users. As such Netflix wants to be able to recommend relevant shows/movies to users(even if we don't have enough knowledge to predict their taste). In order to do this a contest was created by Netflix where contestants will try to predict a users score to a given movie based on a list of ratings users gave to movies in the past.

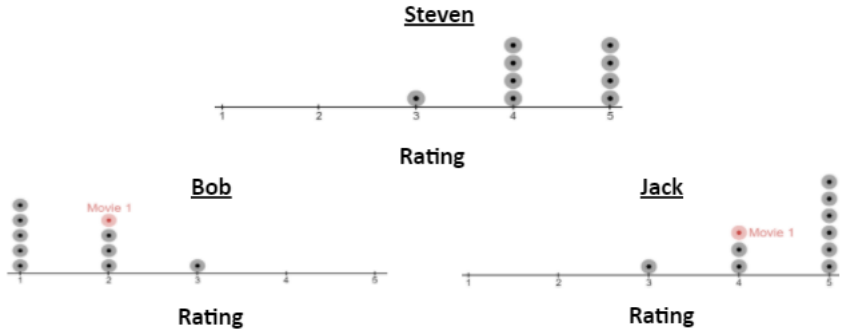
Looking at the works that other contestants have created it seems that a lot of people fundamentally chose to tackle this problem by predicting ratings off of a given user's previous data. For instance one of the leading solutions which is the BellKor solution with accuracies of about 65% found at the following link:

`https://netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf`

tackled this problem using linear regression. Where predictions to a user's rating for a given movie were made by looking at the user's prior data and increasing or decrease weights of the linear regression's explanatory variables based on the user's previous data. Other solutions also seemed to fundamentally rely on only using the given users prior data to make predictions except with different methods like decision trees or Support Vector Machines instead of linear regression.

2 Method

In this section we will be talking about the method we used to solve our problem. Unlike the solutions other contestants came up with where only the previous data of a user was used to predict a user's ratings we see in our solution we used users who are similar to the current user to predict ratings. In our method we assume that users who have a similar rating history to other users will give similar ratings to a particular movie. For example in the diagram below:



If we wanted to predict the rating the user Steven would give for movie 1 we would look at each of the other users and find the most similar user to Steven. As seen the most similar user would be Jack as both Steven and Jack are seen to give high ratings for movies based on the distribution of previous ratings. We would then predict the rating that Steven would give for movie 1 by looking at the rating our similar user gave for movie 1 which in this case we see that Jack gave movie 1 a rating of 4. We would then predict this rating for Steven so we would predict Steven to give movie 1 a rating of 4. It should be noted that this example is highly generalized and there will be some differences made that will be talked more about in the Experiment Setup section.

There are a couple reasons why we felt our method was better at predicting a user's ratings when compared to the standard method of predicting ratings using prior data. The biggest reason is that our method is an enhancement to the standard method. As seen by our method we do use prior data in developing our model however we use the user's prior data only to build a representation of the user's rating patterns. Our prediction comes mainly from looking at ratings given by other users with similar rating patterns. In other words we feel that this method would be better as we use both a user's prior data and other similar users data in tandem to make predictions which should be better than just predicting using only prior data.

3 Experimental Setup

In this section we talk about the following topics that is necessary to understand how our experiment was conducted:

- Data set
- Hypothesis
- Experiment Design

3.1 Data Set

The Data sets we will be using in our experiment comes from the Netflix Prize Data sets found on

`https://www.kaggle.com/netflix-inc/netflix-prize-data`

Because the main data set is too large to contain in 1 file the data is broken into 4 different files called `combined_data1.txt` all the way to `combined_data4.txt`

Each of the `combined_data.txt` files follows the following format shown below:

```
1:
1488844,3,2005-09-06
822109,5,2005-05-13
885013,4,2005-10-19
......
2:
2059652,4,2005-09-05
1666394,3,2005-04-19
1759415,4,2005-04-22
.....
```

The files contain a list of movie Ids which represents a different movie in Netflix's database. Underneath each movie Id there is a list of customers who rated the given movie with each line having the following format: User Id, Rating, Date. For example the row [1488844, 3, 2005-09-06] represents that the user with the user Id = 1488844 gave the movie with the movie Id = 1 a rating of 3 on the date 2005-09-06.

This dataset contains approximately 100 million rows with about 500,000 unique users and 20,000 unique movies. More information on the movies such as date movie was created and title of movie is found in the `movie_titles.csv` file however this file isn't used for any of the experiment.

3.2 Hypothesis

In this Experiment we will be looking at how well the method we laid out in the Method section works when it comes to classifying a users rating for a movie. In particular we will look at the following questions:

1. Does classifying using similar users actually result in better accuracies than random guessing?
2. How well does our classification algorithm compare to other competitor's solution such as the BellKor's solution ?

3.3 Experiment Design

To answer the questions laid out in our Hypothesis section we will be conducting a single experiment that should provide answers to all the given questions. This experiment has 8 steps that we will be going over in more detail below.

- **Step 1:** First we read in the Netflix data set and and perform a 10 fold Cross validation on the data set. This splits our data set so that for each cross validation we see that 10% of the data becomes data used for testing while the other 90% of the data becomes data used for training.
- **Step 2:** We then preprocess the training data set. We do this by going through the training data set and for each unique user we compute the median and standard deviations of the ratings and dates the user gave. Each row in the preproccesed training data is supposed to represent each unique users previous voting patterns by using the center and spread of their ratings and dates of ratings in order to represent the users previous rating patterns. It should be noted that the reason why median is used instead of mean is because median is far less sensitive to outliers in the data set.

  In the graphic below which depicts an example of this step we see that the first row of the preprocessed training data comes from taking all of user 1's ratings found in the training set and getting the median and standard deviation of the rating and dates. This results in the preprocessed training data hav-

ing the same number of rows as the number of unique users in the training set.

Training Data

User ID	Rating	Date	Movie ID
User 1	5	2009-09-06	Movie 1
User 1	4	2009-10-25	Movie 25
...	...	...	...

Pre processed Training Data

User ID	Median Rating	Std Rating	Median Date	Std Date (in days)
User 1	4.5	.35	2009-09-30	15
...	...	...	...	...

- **Step 3:** Next we will be creating and training our K nearest Neighbors model. Using the preprocessed training data that we created before we use the User Id as the classification variable and the median and standard deviation of ratings and dates of ratings as the input variables. This creates a K Nearest Neighbor model which takes in a user's rating patterns(median and standard deviation of rating and dates of ratings) and returns the user Ids that have the closest or most similar rating patterns to the given user.
- **Step 4:** Now we will reach the testing stage. We will be using our testing data set for this step and it should be noted that although each row in the testing data set has the same features as our training data set(user id, rating, date, movie Id) we are unable to use the rating feature in our model. This is because the rating found in the test data set represents the actual rating which will be used in a later step. In this step the only features that will be used in the test set is the the user id, movie id and date of rating.

  In order to start testing we need to preprocess the testing data so that we can use it in the K Nearest Neighbors model. To start we get each row of testing data and search through the preprocessed training data that was created in step 2 and try to find the distribution of the user that is found in the testing data. For example in the graphic below we see that to preprocess the first row in the testing data we take the user id in the testing data which in this case is user 1 and then we find user 1's median and standard deviation ratings and dates of ratings in the preprocessed training data. We then assign this value into the testing data. Since we expect a users rating distribution to not change between training and testing we are able to assume user 1 holds the same rating distribution in the testing set as found in the training set.

Testing Data

User ID	Movie ID
User 1	Movie 1
User 1	Movie 15
...	....

Preprocessed Training Data

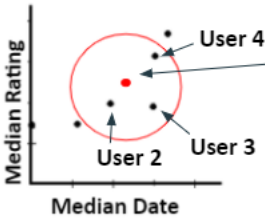
User ID	Median Rating	Std Rating	Median Date	Std Date (in days)
User 1	4.5	.35	2009-09-30	15
...	...	...	...	...

Preprocessed Testing Data

User ID	Median Rating	Std Rating	Median Date	Std Date (in days)
User 1	4.5	.35	2009-09-30	15
User 1	4.5	.35	2009-09-30	15
...	...	...	...	...

  If the user we are trying to find a rating distribution for isn't found in the preprocessed training data then that means that the given user never rated a movie before. In this case we only have access to the features found in the test data set(user id, date of rating, movie Id). In this case we will set the following for the features in the given row of the preproccessed test data: median rating = 3 , standard deviation rating = 2, median of rating date = user's date from test set, standard deviation of rating date = 0. This represents a user who has almost an equal chance of rating any number from 1 - 5 and who rates solely on the date found in the testing set. We do this because since we don't know how this user will rate we expect that the user would have an equal chance to rate all values however since we have information on the date of the rating we want to make sure to find users that rate as closely in their rating dates to the given user.
- **Step 5:** In this step we will take the preprocessed testing data that we created in the previous step and use our model to classify the point. Our preprocessed testing data should contain the following features: [median rating, standard deviation of ratings, median date of ratings, standard deviation of date of ratings] which represents the rating distribution of the user in the test data row. Using these features and inputting it into our K Nearest Neighbors model described in step 3 we should get a list of user id's that have similar rating distributions to the rating distribution that was inputted in the K Nearest Neighbor

model. This list contains all the user ids in the training data with the first value in the list being the user with the most similar rating distribution to the inputted values and the last value in the list being the user with the least similar rating distribution to the inputted values. As seen by the graphic when inputting User 1's rating distribution we see that the list of users with the most similar rating distributions are User 2, User 3, and User 4.



Test point 1 values

User ID	Median Rating	Std Rating	Median Date	Std Date (in days)	Movie Id
User 1	4.5	.35	2009-09-30	15	Movie 1

- **Step 6:** In step 6 we take the list of similar users given by the K nearest Neighbors model and reduce it more. We do this by taking the movie id of the rating our test user is trying to predict and remove all users from the list of similar users who didn't rate the given movie. As seen in the example below which is a continuation of the previous examples we see that we were trying to find user 1's rating for movie 1. In this case we are given the list of similar users to user 1 from our K Nearest Neighbor model and with the list of similar users we keep only the users who rated movie 1. As seen by the graphic only User 2 and User 4 rated movie 1 so they were kept. At this point you should have a list of similar users who rated the same movie that the user you are testing for did.

List of Similar Users

User ID	Rating	Movie ID
User 2	5	Movie 1
User 3	2	Movie 3
User 4	5	Movie 1

List of Similar Users who rated given Movie ID

User ID	Rating	Movie ID
User 2	5	Movie 1
User 4	5	Movie 1

- **Step 7:** In step 7 we look at the ratings that each user in the similar user list gave for the movies we are trying to find. From the ratings given we take the super majority of the ratings and predict the test users rating for the given movie using the super majority. In the example below we see the super majority of the ratings of the similar users who rated movie 1 was found to be 5 so we would predict user 1 to give movie 1 a rating of 5.

List of Similar Users who rated given Movie ID

User ID	Rating	Movie ID
User 2	5	Movie 1
User 4	5	Movie 1

- **Step 8:** The last step is conducted after each row of the test data set has been classified with a predicted rating. In this step we calculate the accuracy by taking the actual rating that was found in the test data and comparing it to the predicted rating we gave. We get the accuracy of our classification through the formula below:

$$\text{Accuracy} = \frac{\text{Total Amount of Correct Predictions}}{\text{Total Amount of Test Data Rows}}$$

4 Results

In this section we will start to look at the results of our experiment and answer the following 2 questions that we looked at in the hypothesis section.

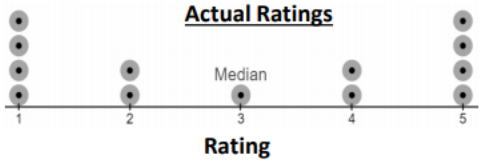
1. Does classifying using similar users actually result in better accuracies than random guessing?
2. How well does our classification algorithm compare to other competitor's solution such as the BellKor's solution ?

Looking at the results of our experiment it is seen that we got the following accuracies for the 10 cross validations that were performed.

```
0: 0.4525083820987526
1: 0.458193631687457
2: 0.46552406339157415
3: 0.4741248255898707
4: 0.4742081259501447
5: 0.45681917574293507
6: 0.4533413857014932
7: 0.44386596972031905
8: 0.4421583123347008
9: 0.4119496855345912
```

The results help answer the first two questions that we asked in the hypothesis. In the first question we asked whether classifying using similar users actually resulted in an improvement in accuracies when compared to random guessing. As seen by our results the average accuracy for the 10 Cross validation is about 45.2% which is a 25% increase in accuracy when compared to random guessing which yields accuracies of 20%. This was expected and does show a significant improvement in classification accuracy when compared to random guessing proving our expectations correct.

Looking at the second question in the hypothesis section it was seen that our answer isn't as favorable compared to our answer to the first question. When we compare the results of our algorithm compared to other competitors like the BellKor solution we see that we under performed in our accuracy by about 20% as it was seen that the BellKor's solution had an average accuracy of about 65%. There are a couple reasons why our solution was seen to have a lower accuracy. The first reason was the case where a user has multiple peaks within their data. For instance in the diagram shown below we see that the rating distribution for the given user has multiple peaks. In cases like this we notice that the center of the data isn't accurately represented as the center doesn't fall in the location where the majority of the data is found like it should be. This results in the users rating patterns not being properly represented resulting in inaccurate predictions being formed. If our method accounted for this by keeping track of all the local maximum found in the rating distribution we would then be able to avoid this case and give better predictions.



The second reason why our accuracy wasn't as good was due to the accuracy metric we used. In our experiment we used a strict accuracy metric where we only counted rating predictions that were equal to the actual ratings found in our test set. In the case that our predicted rating was close to the actual rating our accuracy wouldn't increase. This was different to other solutions such as the BellKor solution that used RMSE(Root Mean Square Error) as their accuracy metric which would increase the accuracy if the predicted rating was close to the actual rating. In this case you would naturally see a higher rating score if you were to use RMSE instead of the strict rating we used. In the end it is hard to know if our solution was less accurate due to the fact that we used similar users in prediction or if it was due to one of the two potential issues listed above either way more testing would have to be conducted where accuracy measures are kept the same and user rating representations are more better represented.

5 Conclusion

Doing this experiment I believe that we can conclude that predicting using similar users definitely results in greater accuracies compared to random guessing. That said it is inconclusive if using similar users for prediction is better or worse than the standard method of using a user's prior data. This is due to unaccounted for factors like differences in accuracy metrics, misrepresented data, and differences in modeling techniques(KNN vs Linear regression) which all can influence the accuracy of a classification. If this experiment was to be conducted again some changes that should

be made is to better represent a users rating by keeping track of the local maximums in the data. Another change would be to use RMSE as the accuracy metric which would make our solution more easier to compare to other competitor's solutions. A final change would be to use a better modeling technique than K Nearest Neighbors. Although KNN is a good modeling method that works it takes a significant amount of time to run and process data as such it would be recommended to change the modeling technique to linear regression or decision trees as these will perform much faster with larger inputs. Given all the circumstances it should still be noticed that if you want to classify a users ratings for a movie the method proposed in this paper is still a valid and useful technique.

Github Link: `https://github.ncsu.edu/engr-csc422-fall2020/P04`