Collaborative Filtering on Netflix Challenge

Prabhjot Kaur
CSC 7810

Introduction

The goal of this project was to create a movie recommendation system. This helps content providers such as Netflix to recommend movies to their customers that match their interests. This similar system could be used for other related problems such as for book recommendation or any other product recommendation. To accomplish this task, the Netflix Prize dataset was used. This dataset was first published by Netflix in 2006 when they announced the 'Netflix Prize' challenge. The description of the dataset is provided below.

Dataset Overview

Original Dataset: The dataset is divided into three sets.

1. Training set
2. Qualifying set
3. Probe set

The difference between the qualifying set and the probe set is that the ratings for the qualifying set are withheld by Netflix and they are not released to public. The probe set, however, is created to help the participants test the accuracy of their methods before submitting the results for the qualifying set to Netflix. Below is the breakdown of the data in numbers.

- Total movies across the entire dataset: 17770
  - Movie IDs range from 1 – 17770 with no gaps
- Total Customers/Users included in the entire dataset: 480189
  - Customer IDs range from 1 – 2649429, with gaps
- The rantings are on a scale from 1 – 5
- The date of rating is also provided in the dataset. It is in the format YYYY-MM-DD

Dataset used for this project: For the class project, I used training set and probe set. The qualifying set is not used because there is no way to compare the predicted ratings to the actual ratings by the customers as those are not available to public and the challenge is already closed. Additionally, I also used the overall movie ratings from IMDb for the movies in the Netflix original data. This data is not provided in the original Netflix dataset, but it is a community effort. The ratings in the IMDb range from 1-10. This additional data also contains the director, writers, genres, and production companies for some of the movies. See Table 1 for more details.

d	year	title	Runtime	Rating	Directors	Writers	Production companies	Genres
1	2003	Dinosaur Planet	50	7.7	Pierre de Lespinois	Mike Carrol-Mike Carroll-Georgann Kane		Documentary-Animation-Family
2	2004	Isle of Man TT 2004 Review						
3	1997	Character	122	7.8	Mike van Diem	Ferdinand Bordewijk-Laurens Geels-Mike van Diem	First Floor Features-America Film	Crime-Drama-Mystery
4	1994	Paula Abdul's Get Up & Dance	54	8.8	Steve Purcell			Family
5	2004	The Rise and Fall of ECW	360	8.6	Kevin Dunn	Paul Heyman	WWE Home Video	Documentary-Sport

Table 1: Supplemental data

To further understand the dataset, below is a small description of the format of the data in the training set and the probe set.

1. Training set format: It is a .txt file that contains lines indicating a movie id followed by a colon. Each subsequent line in the file corresponds to a rating from a customer and its date in the following format:

   MovieID1:
   <CustomerID11, Rating11, Date11>
   <CustomerID12, Rating12, Date12>
   ...
   ...
   MovieID2:
   <CustomerID21, Rating21, Date21>
   <CustomerID22, Rating21, Date22>
2. Probe set format: It is a txt file that contains lines indicating a movie id, followed by a colon, and then the customer ids, one per line for that movie id. See below.
   MovieID1:
   <CustomerID11>
   <CustomerID12>
   …
   …
   MovieID2:
   <CustomerID21>
   <CustomerID22>

   The ratings and the rating dates for each pair are contained in the training dataset. This can be used to test the methods developed in this project.

   The goal here is to predict the rating for a user and movie as specified in the Probe set. The implantation to obtain the rating prediction is explained below.

Methods

The methods adopted in this project belong to a class of algorithms called collaborative filtering. The idea of collaborative filtering is to use collective learning from all the users in the database to make future predictions/recommendations for products (in this case, for movies) for the users. This is opposed to content-based methods, which rely on a particular user's history alone. The methods

in collaborative filtering have an advantage over content-based methods because more data is available to use to make predictions. However, despite the availability of more data compared to content-based methods, the data is still very sparse to use in the raw form. Therefore, pre-processing is required. The next section explains the pre-processing steps that were applied in this project.

Preprocessing

Below is a snapshot of the data without any pre-processing applied. There is a total of 24053764 data points.

	movie_id	user_id	rating
0	1	1488844	3
1	1	822109	5
2	1	885013	4
3	1	30878	4
4	1	823519	3
...	...	...	...
24053759	4499	2591364	2
24053760	4499	1791000	2
24053761	4499	512536	5
24053762	4499	988963	3
24053763	4499	1704416	3

24053764 rows × 3 columns

Figure 1

As a pre-processing step, I selected top 5000 users from the entire dataset who rated more often. This is a valid preprocessing step because the users who did not rate much do not add any more value to the predictions. Note that this pre-processing is only applied to the training set. Meaning that we can still make predictions for the users who do not provide much feedback, but these users do not add much value to the collaborative filtering approach. After this step, the training set has 5000 unique users and a total of 2296470 data points (ratings).

The final dataset for training after this pre-processing step is converted to the following format, where X contains user IDs and movie IDs in column 1 and 2, respectively. Y contains the ratings for those users and movies.

```
X =  [[1488844       1]
      [1488844       8]
      [1488844      17]
      ...
      [  93124    4474]
      [  93124    4488]
      [  93124    4496]]
```

```
Y =       [3 4 2 ... 4 3 5]
```

The next section of the report reviews the methods implemented in this project.

Method 1: K-Nearest Neighbor (User-User similarity)

The first method used is called the K-Nearest Neighbor method that is based on user-user similarity. Since the data is large and sparse, it is impractical to use matrix method to apply the KNN model. Therefore, the approach that was developed to deal with the big dataset is as follows. Given a user (referred to as target_user) and a movie (referred to as a target_movie) for which the rating is to be predicted, the following steps are applied.

1. Find all the movies that this target_user has watched/rated.
2. Find all the other users in the dataset that have watched one or more of the same movies as the target user.
    a. This step is to select a subset of users to compare the similarity of the target_user with. So, rather than comparing the target user with all other users in the dataset (or even with the top 5000 users down selected in the pre-processing step), we only compare him/her with those users who share similar movies watched in the past.
3. After we have a subset of users with whom the target_user is to be compared, the following steps are further applied,
    a. When comparing the target_user with each of the other user selected according to the criteria defined in steps 1 and 2, the comparison is based only on the movies that they both have watched. So, for example, if the other user A and the target_user have 1 movie in common, then their similarity is based on the rating for this 1 movie. However, if other user A and target_user have 500 movies in common, then the similarity is based on their ratings for these 500 movies.
    b. The similarity is calculated for each of the other user and the target_user using Pearson Coefficient.
4. At this step, we have the Pearson Coefficient. That is, we know how similar the target_user is with each of the other users selected in steps 1 and 2 above. However, there is problem that needs to be addressed before using the Pearson Coefficient calculated above.
    a. As is pointed in step 3a, the similarity is based on the common movies watched between the target_user and the other user A. Because the common movies between the target_user and other users are not always the same, the Pearson Coefficient calculated above does not give a fair comparison with all the other users. See table 2 as an example that highlights this scenario showing that User 1 and the target_user have 6 movies in common User 2 and the target_user only have 2 movies in common.

Target user	**1**	**3**	**5**	**2**	**1**	**3**	**4**	**5**
User 1	**5**	**5**	**2**	**4**	**4**	**5**		
User 2							**5**	**4**

Table 2

Further, the Peason Coefficient between (target_user and User 1) = 0.01 and the Pearson Coeffiicient between the target_user and IUser 2 is 0.9, one cannot conclude that the target_user is more similar to User 2 than User 1. To address this, we consider how many movies/data points were used when calculating the Pearson Coefficient and then adjust the Pearson Coefficient accordingly. The adjusted Coefficient is obtained is as follows. See column 5 in Table 3 below.

Subset of users with whom to compare the target_user	Common movies between target_user and other users	Normalized common movies vector (previous column) [0 -1]	Pearson Coefficient	**Corrected Pearson Coefficient**
User 1	6	N1	P1	**N1*P1**
User 2	2	N2	P2	**N2*P2**

Table 3

The idea here is to use the Pearson Coefficient that considers how many data points were used to produce the Pearson Coefficient. The result of this method was that the Pearson Coefficient values after multiplying with the normalized movie count were smaller. This was as expected but it is not a problem.

Example 1: When Pearson Coefficient was originally positive

```
[0.11375276886539265, 0.05753105788328998, 0.005380506454629558, 0.221870878
[157, 236, 177, 221, 345, 133, 272, 157, 166, 224, 165, 161, 159, 137, 116,
[0.0001726357967471897, 0.0002595034906518266, 0.0001946276179888 6994, 0.000
[1.96377999e-05 1.49295103e-05 1.04719515e-06 ... 5.52817588e-05
 5.32036802e-05 2.93629749e-05]
```

Pearson coefficient before (raw): 0.11375
Pearson Coefficient after (adjusted): 1.963* 10^-5

This works well for the positive values of the Pearson Coefficient. Because all of the positive values were impacted the same way. However, the problem of applying the above method when the Pearson Coefficient was negative had the opposite effect. See an example below.

Example 2: When Pearson Coefficient is originally negative. Then multiplying with a small as done above lead to a bigger number. Meaning that if the target_user and User 1 had a similarity score of -0.0685. Then after adjusting, the score became -1.289*10^-5. In other words, the target_user and IUser 1 became more similar after the adjusted Pearson Coefficient than they were before. See an example below.

```
-0.06857723042270793
171
0.0001880300716163 6586
-1.2894581547633795e-05
```

This is a problem indeed. However, with the

Figures 2 and 3 explain the steps discussed above pictorially.

```
for i in range(0,size(TestSet):
    target_movie = TestSet[i,:][0] #Get the target user
    target_user = TestSet[i,:][1]  #Get the target movie

    #Get all the other movies rated by the target user using the training set
    OtherMoviesRated_target_user = (X[X[:,0]==target_user])[:,1]

    #[1] Get all the users who have watched/rated the same movies as the target user
    OtherUsers = []
    for common_movies in OtherMoviesRated_target_user:
        OtherUsers_temp = (X[X[:,1]==common_movie])[:,0]
        OtherUsers = OtherUsers + OtherUsers_temp

    #[2] Create a list of movies that have been rated by both (target_user and the other user)
    # Then, use this set to find the similarity between the target user and this user using the Pearson Coefficient
    for user in OtherUsers:
        #Find a common set of movies rated
        TotalCommonMovies
        #Find the Pearson Coefficient
        PearsonCoefficient

    #[3] Post process the Similarity score.
    # Normalize the TotalCommonMovies vector
    # Multiply the Normalized TotalCommonMovies vector with the PearsonCoefficient
    Corected_PearsonCoefficient

    #[4] Select Top NN neighbors
    NN = 50 #when available, otherwise select however many are available

    #[5] Predict the ratings of the target_user using the weighted average of NN nearest neighbors
```

Figure 2: Method 1 explained

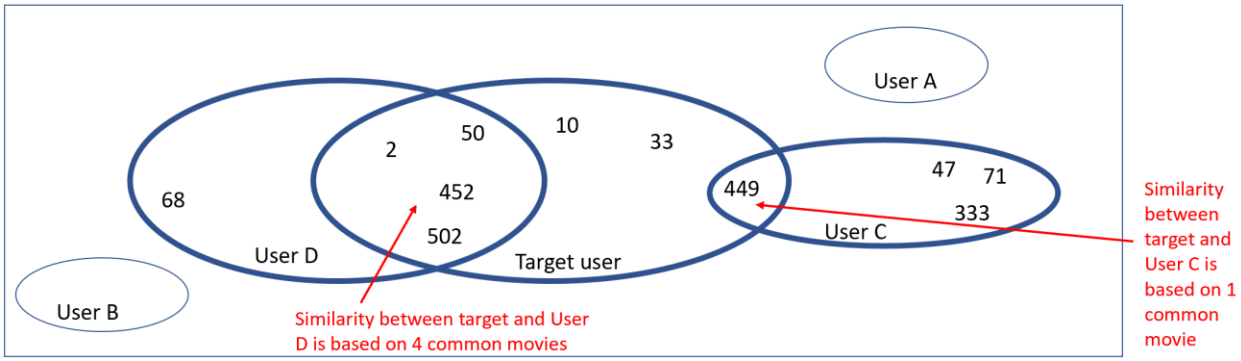


Figure 3: Method 1 explained

Method 2: K-Nearest Neighbor (Movie-Movie similarity)

This method uses the K – nearest neighbor algorithm. The movies are compared against each other and predictions are made for the users in the test set. The following steps were adopted. For all the users in the test set,

1. Find all the movies that this target_user has watched/rated.
2. Find all the other users in the dataset that have watched one or more of the same movies as the target user.
    a. Compare the target_user with each of the users in this subset using the Pearson Coefficient.
    b. Note, the movies not rated by the target_user or the other users being compared were marked as being rated 0 before the Pearson Coefficient was calculated.

3. Select the top 50 neighbors if available. If total neighbors are less than 50, then use all the available neighbors.
4. Predict the rating for the target_user using the weighted sum of his/her neighbors.

The outline of the method is also shown below.

```
for i in range(0,size(TestSet):
    target_movie = TestSet[i,:][0] #Get the target user
    target_user = TestSet[i,:][1]  #Get the target movie

    # Get all the other movies rated by the target user using the training set
    OtherMoviesRated_target_user = (X[X[:,0]==target_user])[:,1]

    # Find all the users who have watched the target_movie and their ratings
    # Create a 3x5000 matrix where the 2nd row contains ratings of the target_movie
    # And the 3rd row will contain the ratings of the other_movie that the target_user has watched

    # [1] Find the similarity of the target_movie with all other movies watched by the target_user
    for common_movies in OtherMoviesRated_target_user:
        # Fill in the third row of the matrix created above with the ratings for the common_movie
```

User IDs	1	2	3	4	5	6	7	8
Ratings for the target_movie	x	0	x	0	x	x	0	0
Ratings for the common_movies	0	x	x	0	x	0	0	x

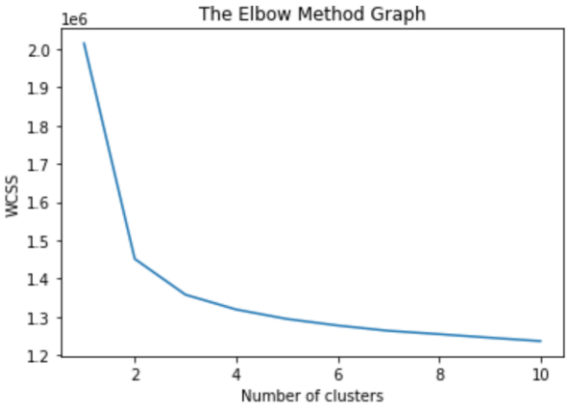
```
        # Find the Pearson coefficient

    #[2] Select Top NN neighbors
    NN = 50 #when available, otherwise select however many are available

    #[3] Predict the ratings of the target_user using the weighted average of NN nearest neighbors
```

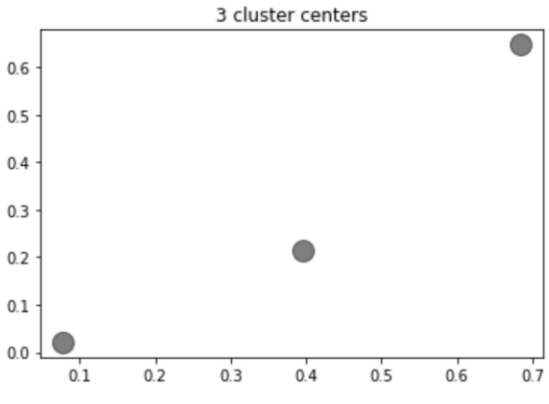
Method 3: K-means Clustering

The third method that was studied was k-means clustering. The idea here is to divide similar movies into clusters.

First, elbow method is used to find the number of clusters that would fit the data. The graph below suggests 3 clusters. However, given the data size, this seemed insufficient.

The graph below also shows the cluster centers for the three clusters. K-means clustering was eventually ruled out from this project.





Method 4: Non-Negative Matrix Factorization

Non-Negative Matrix is another method that is used to solve this problem. Here the user-movie matrix is split into two factors as follows.

$$V = W*H$$

Where V is the user-movie matrix of size (m X n) and W and H are factors of V of size (m X p) and (p X n), respectively.

To implement this method, only a small subset of data was used because of the lack of enough memory on my machine to store the large matrix. The data size was 100 x 100 (This is top 100 movies and top 100 users). The number of factors used was 20.

Method 5: Prediction using user's history (Supplemental data)

Finally, I also tried to make use of data that was originally not part of the Netflix dataset. This supplemental data contains the name of the director, genre and producer for the movies in the Netflix dataset. This data is collected form the IMDb. The dataset is highly sparse and the genre was not very useful. The data was sued as follows.

1. For a target_user for whom a prediction is to be made for a target_movie, find all the other movies that this user has watched.
2. For all the movies that this user has watched, find the similarity of the target_movie with all the other movies he/she has watched.
    a. The similarity here simply means that we check to see if there is a favorite director or producer that this target_user has. If the target_movie is also from the favorite director or producer, then recommend the target_movie to the target_user.

```
for i in range(0, size(TestSet)):
    target_movie = TestSet[i,:][0] #Get the target user
    target_user = TestSet[i,:][1]  #Get the target movie

    # Get all the other movies rated by the target user using the training set
    OtherMoviesRated_target_user = (X[X[:,0]==target_user])[:,1]

    # [1] First, find the top Z movies rated by this user and their related information
    Z = 10
    Directors =
    IMDb rating =

    # [2] Find if the target_user has a favorite director or a specific genre

    # [3] Check if the target_movie is from the same director or a genre that target user likes more

    # [4] Rate the target_movie as the same as the other movie he/she liked with the same director/genre
```

Analysis Results and Comparison

Before reviewing the experimental results, few issues are discussed below that were encountered when working on this problem.

Issues encountered:

1. For methods 1 and 2, when K-Nearest Neighbor was used, Pearson Coefficient was used to measure the similarity. This was problematic in the cases where some users rated all the

movies with the same rating. This is because, when calculating the Pearson Coefficient using the formula below, it resulted in a division by a Zero.

Average = 2

x	2	2	2	2	2
y	5	4	2	3	1

Example: User x rated all movies the same

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient
x_i = values of the x-variable in a sample
$\bar{x}$ = mean of the values of the x-variable
y_i = values of the y-variable in a sample
$\bar{y}$ = mean of the values of the y-variable

Pearson Coefficient

In order to compensate for this problem, whenever such as case encountered, the Pearson Coefficient was set to 0.

Results

I used Root Mean Squared Error (RMSE) and Mean Absolute Error (MAS) to evaluate the results. RMSE metric was chosen because this was the metric that was used in the original challenge by Netflix. Below are the results from Methods 1 and 2 and 4.

Note the results from Method 4 (using the Non-Negative Matrix Factorization) are certainly better than the other two. However, the amount of data used for Method 4 was very small compared with other results. So, it is not totally a fair comparison.

	Method 1	Method 2 (with part of the dataset)	Method 2 (with the entire dataset)	Method 4 (with part of the dataset)
RMSE	**2.213309**	**1.08041**	**1.060477**	**0.368629**
MAS	**1.846437**	**0.80641**	**0.77795**	**0.120844**

Results

Below is an example of few predictions using Methods 1 and 2.

Actual	5	4	4	5	5	4	3	1
Method 1	1.26	0.84	2.16	2.57	2.69	1.46	3.48	1.02
Method 2	3.44	3.68	3.78	4.22	5.2	4.02	2.26	3.24

Conclusion

The goal of this project was to explore collaborative filtering techniques to create a recommender system for movies. KNN, k-means clustering and NMF methods were explored, and the results are shared in the section above. It was apparent that NMF performs better than KNN. The main problem encountered in this project was dealing with data sparsity and the size of the dataset. The size of the dataset meant that the data simply could not be converted into a matrix format where KNN and NMF methods would have been very easy to apply. Therefore, the techniques discussed in the Methods section were created to solve the problem. The code developed for this project is shared in Appendix of this report.