Collaborative Filtering on Netflix Challenge

Prabhjot Kaur
CSC 7810

Wayne State University
ea4728@wayne.edu
http://thecarlab.org

Agenda

- Introduction
  - Problem Statement
- Dataset
- Methodology
- Experiments and Result
- Future Work

Introduction

- **Problem Statement**
  - Predict the ratings for a given user for a specific movie using collaborative filtering techniques
  - Competition held by Netflix for the best collaborative filtering algorithm
    - On September 21, 2009, the grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos.
    - RMSE: 0.8567
    - Cinematch scores an RMSE of 0.9514

Dataset

- **Netflix Challenge Dataset**
  - Training set
  - Probe set (test set)
  - Qualifying set (used by Netflix to evaluate all competition entries, not used in this project)

- **Total movies: 17770**
  - Movie IDs range from 1- 17770
  - Subset used: 1- 4499
- **Total customers/users: 480189**
  - Customer IDs range from 1 – 2649429, with gaps
- **Total ratings available: 24053764** ~1.11% of this subset of the dataset is rated
- **Ratings: 1 – 5**
- **Date of rating: YYYY-MM-DD**

* = 2160370311 ratings possible

MovieID1:
<CustomerID11, Rating11, Date11>
<CustomerID12, Rating12, Date12>
…
…
MovieID2:
<CustomerID21, Rating21, Date21>
<CustomerID22, Rating21, Date22>

Training set

MovieID1:
<CustomerID11>
<CustomerID12>
…
…
MovieID2:
<CustomerID21>
<CustomerID22>

Probe/Qualifying set

The matrix method is inappropriate because of the data size and sparsity.

Dataset

- **Supplementary data**
  - Not provided by Netflix
  - Not part of the original competition
  - Information is collected from IMDb
    - One rating per movie

id	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-Almerica 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

Pre-processing

- Used subset of the original training set
  - Movies 1-4499
- Selected Top 5000 users

	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

```
Top_K_users = 5000
group = df.groupby('user_id')['rating'].count()
Top_K_users = group.sort_values(ascending=False)[:Top_K_users]
```

```
user_id
305344      4467
387418      4422
2439493     4195
1664010     4019
2118461     3769
            ...
1146632      334
720436       334
2466146      334
810392       334
2015780      334
Name: rating, Length: 5000, dtype: int64
```

Pre-processing

- Final training set
  - Total users: 5000
  - Total movies: 4499
  - Total ratings: 2296470

```
#Convert dataframe to numpy matrices
X = lite_rating_df[['user_id', 'movie_id']].values
y = lite_rating_df['rating'].values
```

X =

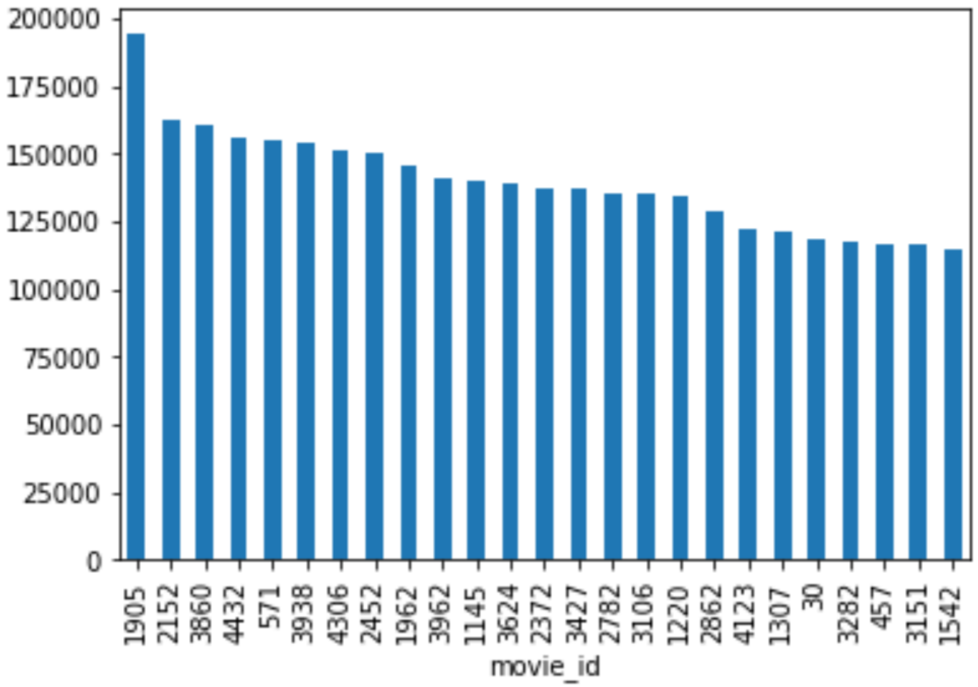
```
[[1488844       1]
 [1488844       8]
 [1488844      17]
 ...
 [  93124    4474]
 [  93124    4488]
 [  93124    4496]]
(2296470, 2)
```

Y =

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

Pre-processing

- Top 25 rated movies out of 4499 ratings
  - This is for visualization only



Pre-processing

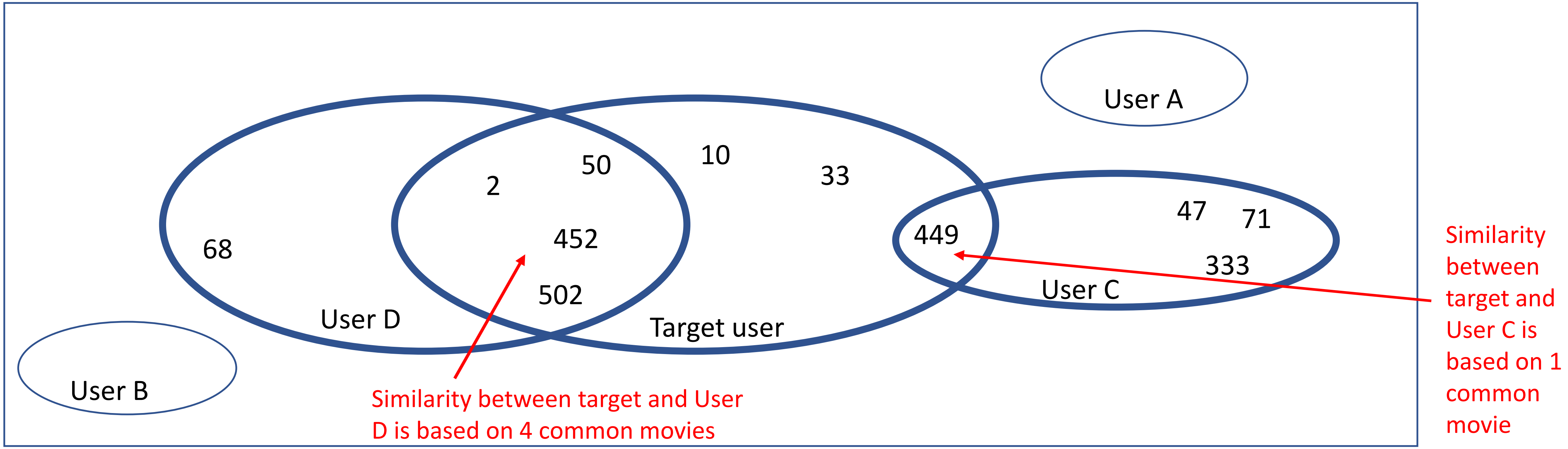
- For the probe (test) set
  - Similar preprocessing was necessary
  - As some of the users were deleted from the training set, applying kNN was not possible
- Final test set
  - Total movies: 4499
  - Total users: 5000
  - Test set: [movieID, userID]
- Goal: Predict the rating for a user for a given movieID

Test set =

```
[[      1 1394012]
 [      1 1774623]
 [    895 1774623]
 ...
 [    927 1356239]
 [    946 1271040]
 [    953 1717060]]
```

Methodology (Method 1)

- kNN: User-User similarity
  - The user for whom the prediction is to be made is compared with other similar users in the training set
  - Similarity measure: Pearson Coefficient
  - Nearest neighbors: 50 or less if 50 are not available
- Method



Methodology (Method 1)

```
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
```

Methodology (Method 2)

- kNN (Movie-Movie similarity)
  - The movie for which the prediction is to be made is compared with **other similar movies that the target user has watched**
  - Similarity measure: Pearson Coefficient
  - Nearest neighbors: 50 or less if 50 are not available

Methodology (Method 2)

```
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
```

Observations

- Some users rated all of the movies with the same rating
  - This causes division by zero when calculating the Pearson Coefficient
  - When this occurred, then Pearson Coeff was assumed to be 0

Average = 2

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

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

r = correlation coefficient

x_i = values of the x-variable in a sample

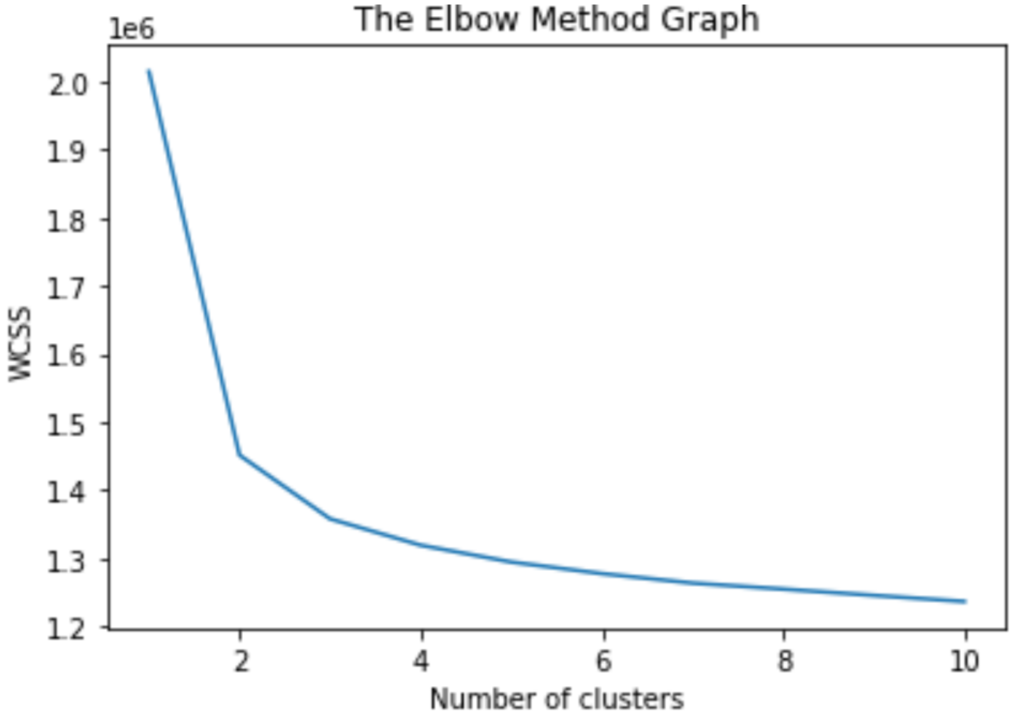
$\bar{x}$ = mean of the values of the x-variable

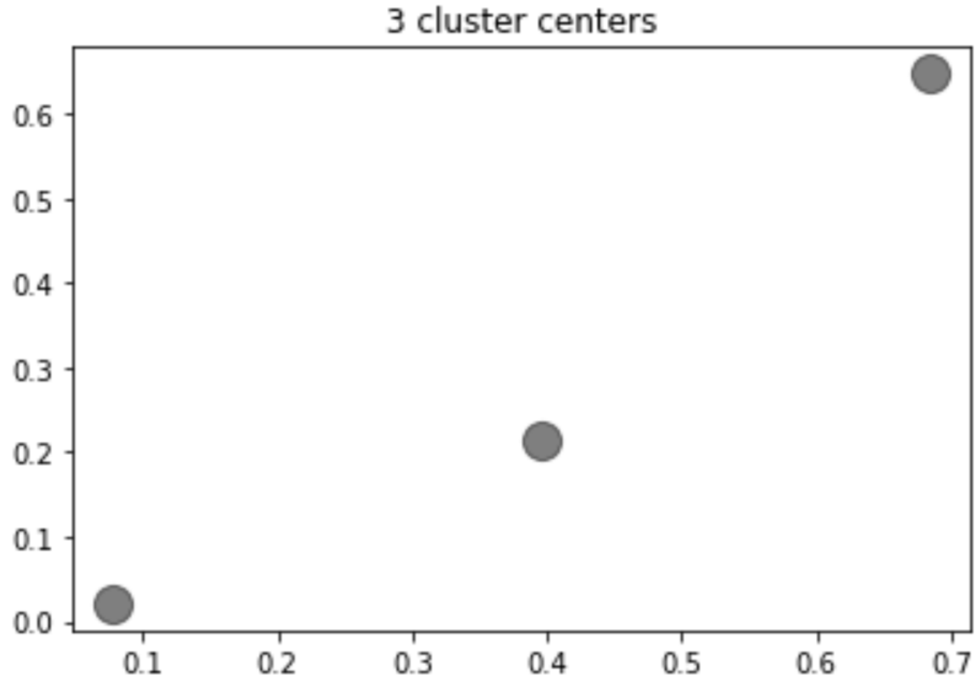
y_i = values of the y-variable in a sample

$\bar{y}$ = mean of the values of the y-variable

Methodology (Method 3)

- k-means clustering
  - Create a 4499x5000 matrix





Methodology (Method 4)

- Use the history of the target user
  - Use the supplementary information

```
The data count is as follows:
id            17769
title         17296
Rating           81
Directors      5611
Writers        9896
Genres         1808
```

```
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
```

Evaluation Metric

- Root mean squared error (RMSE)
  - The original challenge entries were measured in terms of the RMSE value
  - The goal was to beat the Cinematch RMSE by at least 10%

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \hat{x}_i\right)^2}{N}}$$

RMSD = root-mean-square deviation

i = variable i

N = number of non-missing data points

x_i = actual observations time series

$\hat{x}_i$ = estimated time series

Experiments & Results:

	Method 1	Method 2
RMSE	**2.213309**	**1.08041**
MAS	**1.846437**	**0.80641**

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

Future Work

- Finish implementing Method 4
- Implement NNMF
- Use the weighted sum of different methods rather than relying on just one method
- Length of the movie watched if rating is not available
- Use IMDb rating for a new user

References

4. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 5551-5560).

5. Smith, R. (2007). Tesseract ocr engine. Lecture. Google Code. Google Inc.

Thank You!