Movie Recommender System

Report written in Partial Fulfilment
Of credit Requirement for course
CS 725

Course project in

Foundation of Machine Learning

Submitted by

1. 18i190006 Utkarsh Singh
2. 193190024 Kunal Apurva
3. 193190008 Rahul Khankar
4. 193190014 Yogesh Nagar

Table of content

1. Objective

To build a personalized movie recommender system which is based on rating given by user previously. Our movie recommendation system helps users to discover movies based on their liking

2. Related literature

2.1 Recommendation System

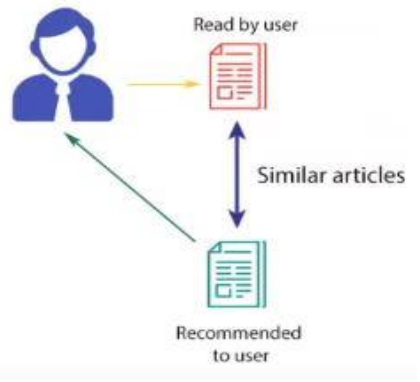
A recommender system is a system which gives recommendation of items such as movies, music, books, news, images, web pages, tools to a user. This information is processed so that it is more close to interest of users. The aim of a recommender system is often to help users learn about new products ones among variety of choices.
It aims at recommending relevant information and removing unwanted or redundant information.

2.2 Utility Matrices: Recommendation systems deal with users and items. A utility matrix gives the Information about how much a user like a particular item. In general, most of ratings are unknown, and our main objective of recommendation system is to recommend items to users by predicting the values of the unknown entries based on the information given as known entries.

2.3 Recommendation systems use a number of different technologies. We can classify these systems into two broad groups.

2.3.1 Content Based

The content based approach consists in analysing the content of the items being recommended. Each user is treated individually. There is no assumption of group or community. The system works mainly by analysing items and the proximity of the selected items to others selected by the user. Then these items are selected to be recommended as they might interest the user.
Example:
Movie based on features such as Same Actors, Director, genre.



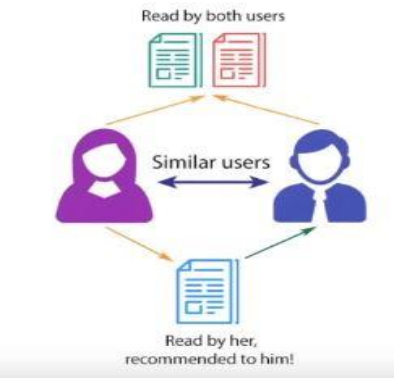
Pros

- No need of data of other users.
- Able to recommend to user with unique taste.
- Able to recommend new and unpopular items.

Cons

- Finding appropriate features is very hard.
- Never recommend item which is not matching with use profile.

2.3.2 Collaborative Filtering



The underlying assumption of the collaborative filtering approach is that if a person *A* has the same opinion as a person *B* on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person.

Pros

- No feature selection needed.

Cons

- Need enough users in the system to find a match.
- Hard to find users that have rated the same items.
- Cannot recommend an unrated item.
- Popularity bias.

Collaborative filtering can be done in two ways

- **User to User**
  In this method we find similarity between the users and recommend item to a user which are top rated by users who are similar to given user
- **Item to Item**
  In this method we find similarity between the items and recommend items which is similar to item which are top rated by given user.

In theory user vs. user and item vs. item are dual approaches. But in practice item vs. item outperform user vs. user in most of cases. There is a valid reason behind it items are simpler than users i.e. items belongs small set of category as compared to user which have very varied variety of taste. So that item similarity is more meaningful as compare to user similarity.
So that we also used item vs. item similarity to solve our movie recommendation problem.

3. Dataset

Our dataset contains 2 different CSV files.

- Movie.csv – It consist of three features movie id, title, genre with 9719 entries
- Rating.csv-It consist of four features user id, movie id , rating, timestamp with 100873 entries

4. Data Pre-processing :

- We merge this file on basis of movie id and drop timestamp, genre because to solve our problem we are using collaborative filtering we does not require any features except rating
- After that we created utility matrices which has user as row and movies as column

- Since we have 610 different user and 9719 different movie so initial utility matrices has 610 row and 9710 columns.
- We have sparse amount of data which is computationally expensive so we have dropped the movies which are rated by less than 10 users.
- After pre-processing dataset we have 610 users and 2269 Movies.

5. Methodology

Description of set of Approaches

5.1 Approach 1:- Collaborative filtering (Using Movie vs. Movie Similarity)

- Let there is a Movie X which is top rated by user A
- Then find N Movies which are similar to Movie X.
- Recommend this N most similar Movies to the user A.

Algorithm for Collaborative Filtering:

Input: Utility Matrix with 2 entities. User and Movie

Pre-processing for utility matrix:

We will subtract the mean rating given by all users to particular movie from each filled entry for that movie.

$$r_{(x,i)} = \frac{r_{(x,i)} - \boldsymbol{mean\_rating(coloum\ of\ Movie\ i)}}{(\boldsymbol{maximum\ rating\ given\ to\ movie\ i - minimum\ rating\ to\ movie\ i})}$$

Similarity Calculation:

Each movie will act as a vector in which user ratings are component of that vector corresponding to that user. We use here Cosine Similarity to find the similarity between movies.

Let A is vector for movie A and B is vector for movie B

$$S_{AB} = cos\theta = \frac{\mathbf{A}.\mathbf{B}}{|\boldsymbol{A}||\boldsymbol{B}|}$$

Technically the cosine distance is actually angle θ between A and B and cosine similarity is angle 180-θ .For our convenience we use cosθ as our similarity measure and call it as cosine similarity.

In this way we calculate similarity between every movie and create similarity matrix which show similarity between Movies.

Recommendation:

Let user X have watched one movie M. Then we will recommend most N similar movie to user X.

Rating Prediction:

$$r_{(x,i)} = \frac{\sum_{y \in N} S_{xy} * r_{yi}}{\sum_{y \in N} S_{xy}}$$

$r_{(x,i)}$= predicted rating given by user i to movie x

S_{xy} = Similarity between two movie x and y

r_{yi} = rating given by user I to movie y in the original utility matrix

Now we will have Prediction of ratings of user y to every movie i.

5.2 Approach 2 : Content based algorithm

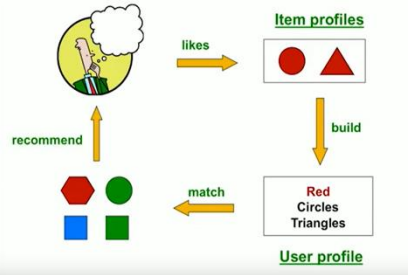
Create Item Profile

For each item create item profile (Set of features)

Example

Movie: {Actor, Director, Genre, Language…..}

Book :{ Author, Publication, Language, type…..}



Create User Profile

For each user create user profile (on same set of features as item)

How to calculate features value for different user?

Let user rated N items i^1, i^2 ,…… i^n with rating r^1, r^2,…… r^n

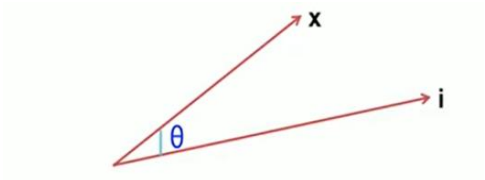
Simple Method: Calculated weighted average of rated the rated item profile (feature)

Better Method : Calculated normalised weight using average of rating for the rated item profile.

Making Prediction using Cosine similarity

User profile X and item profile I

$$cos\theta = \frac{\text{I.X}}{|I||X|}$$



Technically the cosine distance is actually angle θ between X and I and cosine similarity is angle 180-θ

For our convenience we use $\boldsymbol{cos\theta}$ as our similarity measure and call it as cosine similarity.

5.3 Approach 3:- Using gradient descent (Failed Approach)

Recommendation is currently a very popular application of machine learning. In our project, we are trying to recommend movies to customers. We use the following definitions:

n_u = number of users

n_m = number of movies

$r(i, j)$ 1 if user j has rated movie i

$y(i, j)$ = rating given by user j to movie i (defined only if $r(i, j) = 1$)

$\theta^{(j)}$ = parameter vector for user j

$x^{(i)}$ = feature vector for movie i

For user j, movie i, predicted rating: $(\theta^{(j)})^T(x^{(i)})$

$m^{(j)}$ = no. of movies rated by user j

To learn $\theta^{(j)}$:

$$\frac{min}{\theta^{(j)}} \quad \frac{1}{2}\sum_{i:r(i,j)=1}\left((\theta^{(j)}).x^{(i)} - y^{(i,j)}\right)^2 + \frac{\sigma}{2}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

To learn $\theta^{(1)}$, $\theta^{(2)}$, $\theta^{(3)}$..... $\theta^{(nu)}$:

Objective:

$$\frac{min}{\theta^{(1)},\theta^{(2)},\theta^{(3)}\ldots\theta^{(nu)}} \quad \frac{1}{2}\sum_{j=1}^{nu}\sum_{i:r(i,j)=1}\left((\theta^{(j)}).x^{(i)} - y^{(i,j)}\right)^2 + \frac{\sigma}{2}\sum_{j=1}^{nu}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha * \left(\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right).x^{(i)} - y^{(i,j)}\right)x_k^{(i)} + \sigma * \theta_k^{(j)}\right)$$

Given : $\theta^{(1)}, \theta^{(2)}, \theta^{(3)} \ldots \theta^{(nu)}$

to learn: $x^{(i)}$

$$\frac{min}{x^{(i)}} \quad \frac{1}{2}\sum_{i:r(i,j)=1}\left((\theta^{(j)}).x^{(i)} - y^{(i,j)}\right)^2 + \frac{\sigma}{2}\sum_{k=1}^{n}(x_k^{(i)})^2$$

Given $\theta^{(1)}, \theta^{(2)}, \theta^{(3)} \ldots \theta^{(nu)}$

to learn: $x^{(i)}$,....., $x^{(nm)}$

$$\frac{min}{x^{(1)},x^{(2)},x^{(3)}\ldots x^{(nm)}} \quad \frac{1}{2}\sum_{i=1}^{nm}\sum_{i:r(i,j)=1}\left((\theta^{(j)}).x^{(i)} - y^{(i,j)}\right)^2 + \frac{\sigma}{2}\sum_{i=1}^{nm}\sum_{k=1}^{n}(x_k^{(i)})^2$$

Hence algorithm becomes,

- Given $x^{(i)}$,....., $x^{(nm)}$ *learn* $\theta^{(1)}, \theta^{(2)}, \theta^{(3)} \ldots \theta^{(nu)}$
- Given $\theta^{(1)}, \theta^{(2)}, \theta^{(3)} \ldots \theta^{(nu)}$ *learn* $x^{(i)}$,....., $x^{(nm)}$

Minimizing $x^{(i)}$,....., $x^{(nm)}$ and $\boldsymbol{\theta^{(1)}, \theta^{(2)}, \theta^{(3)} \ldots \theta^{(nu)}}$ simultaneously

$$\frac{min}{x^{(1)},x^{(2)},x^{(3)}\ldots x^{(nm)},\ \theta^{(1)},\theta^{(2)},\theta^{(3)}\ldots\theta^{(nu)}}$$

$$\frac{1}{2}\sum_{i=1}^{nm}\sum_{i:r(i,j)=1}\left((\theta^{(j)}).x^{(i)}-y^{(i,j)}\right)^2+\frac{\sigma}{2}\sum_{i=1}^{nm}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2+\frac{\sigma}{2}\sum_{j=1}^{nu}\sum_{k=1}^{n}\left(\theta_k^{(j)}\right)^2$$

Collaborative filtering algorithm:

1. Initialise $x^{(1)}, x^{(2)}, x^{(3)} \ldots x^{(nm)}$, $\theta^{(1)}, \theta^{(2)}, \theta^{(3)} \ldots \theta^{(nu)}$ to some random value.
2. Minimize Cost function using gradient descent

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha * (\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right).x^{(i)}-y^{(i,j)}\right)x_k^{(i)}+\sigma*\theta_k^{(j)})$$

$$x_k^{(i)} := x_k^{(i)} - \alpha * (\sum_{j:r(i,j)=1}\left(\left(\theta^{(j)}\right).x^{(i)}-y^{(i,j)}\right)\theta_k^{(j)}+\sigma*x_k^{(i)})$$

3. For a user with parameter θ and movie with learned parameter x, predict star rating as $\theta^T * x$

Finding most similar Movie:

For each movie we have learn feature vector $x^{(i)}$

Small value of $||\ x^{(i)} - x^{(j)}||$ corresponds to most similar movie.

6. Experiments

6.1 Language and environment

- Language :- Python 3.7
- Lines of code :- 53

6.2 URL

a) https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

b) https://d3c33hcgiwev3.cloudfront.net/_49836d3df9f40f803323ac9d614fcb94_Lecture16.pdf?Expires=1574726400&Signature=QFkYRv7pTsifMbMC7S4GWLY55177Or-1rUxRrGrfkV5f7SpCoZ~yERQ8F2KjyXJtAsJyYfwoYL-jbYvF5bcsnDTno5pVCVpQR4Zb-4ANBeTZq-JZ9Ivdd5zZwLQVPCbJDOg6yDk8pMzHGAt32CqekSlq9xLrHnhHjCoFjsT3Amk_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A

6.3 Results

a) This is utility matrix after combining two datasets for first 5 users

title	'burbs, The (1989)	(500) Days of Summer (2009)	10 Cloverfield Lane (2016)	10 Things I Hate About You (1999)	10,000 BC (2008)	101 Dalmatians (1996)	101 Dalmatians (One Hundred and One Dalmatians) (1961)	12 Angry Men (1957)	12 Years a Slave (2013)	127 Hours (2010)	...	Zack and Miri Make a Porno (2008)	Zero Dark Thirty (2012)	Zero Effect (1998)	Zodiac (2007)
userId															
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	...	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

b) After pre-processing utility matrix

title	'burbs, The (1989)	(500) Days of Summer (2009)	10 Cloverfield Lane (2016)	10 Things I Hate About You (1999)	10,000 BC (2008)	101 Dalmatians (1996)	101 Dalmatians (One Hundred and One Dalmatians) (1961)	12 Angry Men (1957)	12 Years a Slave (2013)	127 Hours (2010)	...	Zack and Miri Make a Porno (2008)	Zero Dark Thirty (2012)	Zero Effect (1998)
userId														
1	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0
2	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0
3	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0
4	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	0.922459	-0.0	-0.0	...	-0.0	-0.0	-0.0
5	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0
6	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0
7	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0
8	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0
9	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0
10	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.000000	-0.0	-0.0	...	-0.0	-0.0	-0.0

c) Cosine similarity matrix

title	'burbs, The (1989)	(500) Days of Summer (2009)	10 Cloverfield Lane (2016)	10 Things I Hate About You (1999)	10,000 BC (2008)	101 Dalmatians (1996)	101 Dalmatians (One Hundred and One Dalmatians) (1961)	12 Angry Men (1957)	12 Years a Slave (2013)	127 Hours (2010)	...	Zack and Miri Make a Porno (2008)	Zero Dark Thirty (2012)	Zero Effect (1998)
title														
'burbs, The (1989)	1.000000	0.099362	0.000000	0.175586	0.035771	0.121634	0.254107	0.078088	0.033189	0.033597	...	0.044805	0.054712	0.151827
(500) Days of Summer (2009)	0.099362	1.000000	0.170668	0.323552	0.224416	0.205295	0.197406	0.224471	0.167848	0.230498	...	0.398158	0.206225	0.103567
10 Cloverfield Lane (2016)	0.000000	0.170668	1.000000	0.036648	0.133145	0.045701	0.020693	0.074532	0.000000	0.291757	...	0.263504	0.118474	0.000000
10 Things I Hate About You (1999)	0.175586	0.323552	0.036648	1.000000	0.278207	0.277931	0.268958	0.093686	0.128758	0.087631	...	0.280563	0.142068	0.165959

d) Movie Recommendation for user

Input User rated for movies as shown

Movie	Rating
(500) Days of Summer (2009)	5
Alice in wonderland (2010)	3
Aliens (1986)	1
A space odyssey (1986)	2

Output

Recommendation for user

```
(500) Days of Summer (2009)              2.405416
Alice in Wonderland (2010)               1.324862
Silver Linings Playbook (2012)           1.179835
Adventureland (2009)                     1.055558
About Time (2013)                        1.039252
Yes Man (2008)                           1.038109
Marley & Me (2008)                       1.035291
50/50 (2011)                             1.026058
Crazy, Stupid, Love. (2011)              1.010425
Help, The (2011)                         1.003605
Up in the Air (2009)                     0.985961
Friends with Benefits (2011)             0.978460
Holiday, The (2006)                      0.977706
Secret Life of Walter Mitty, The (2013)  0.947109
Notebook, The (2004)                     0.940766
Easy A (2010)                            0.934985
Ugly Truth, The (2009)                   0.920143
Perks of Being a Wallflower, The (2012)  0.917591
Step Brothers (2008)                     0.883775
Bridesmaids (2011)                       0.882311
```

7. Efforts

7.1 Fraction of time spent in different parts of project:

- We spent almost 3 days in using gradient descent method to recommend movie.
  But due to sparsity of data and limited time, we were not able to reach the useful result so we have used another approach called **Collaborative filtering**
- We spent around 2 days in understanding and implementing collaborative filtering. We came up with solution of using cosine similarity matrix to find similar movies using **Item to Item collaborative filtering.**

7.2 Most challenging Part:

- Most challenging part was to implement Gradient descent algorithm to learn parameter and feature of movie.
- Because sparsity of data it was computationally expensive to find similarity between movies so we eliminated all movies which have rated by less than 10 users.

8. Challenges

a) Handling New user (Cold Start Problem)
b) Data Scarcity : Not much information about user
c) Scalability: Determining similarity between large numbers of users.
d) Dynamic Updates: Constant updates about new users at every time.
e) human preference varies from one situation to another as well as over time

9. References

a) Towards data science blog
https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

b) Lecture of Prof. Rajaraman, Stanford University
https://www.youtube.com/watch?v=h9gpufJFF-0&t=125s

c) Lecture of Prof. Sudeshna Sarkar, IIT Kharagpur
https://www.youtube.com/watch?v=RVJV8VGa1ZY&t=1148s

d) Lectures of Andrew NG on Coursera
https://www.coursera.org/learn/machine-learning/home/week/9