IR ASSIGNMENT 3

Recommender systems

Team Members

- *Rupsa Dhar (2018A7PS0376H)*
- *Keshav Kabra (2018AAPS0527H)*
- *Meganaa Reddy (2017B3A70973H)*

Dataset used: MovieLens 100K dataset

Language used :Python

Libraries used:

- Numpy
- Pandas
- Scipy
- Sklearn

Recommender System Technique	***Root Mean Squared Error(RMSE)***	***Precision on top K***	***Spearman Rank Correlation***	***Time taken for Prediction (sec)***
Collaborative	3.18760802565	73.107104984%	0.99996158637	0.0552254517
Collaborative along with Baseline approach*	0.46706764236	67.189819724%	0.99999917526	3.16772174835
SVD	0.61505230990	43.4075645%	0.99999856900	8.4650199413
SVD with 90% retained energy **	1.54928895892	99.89395546%	0.99999093609	8.1368098258
CUR	1.3639488322	91.16295510%	0.99999299820	10.4288876056
CUR with 90% retained energy **	1.64054323313	91.16295510%	0.99999299820	10.021700620

COLLABORATIVE FILTERING

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users.

It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. It looks at the items they like and combines them to create a ranked list of suggestions.

In most cases, the cells in the matrix are empty in the user vs movie dataset, as users only rate a few items. It's highly unlikely for every user to rate or react to every item available. A matrix with mostly empty cells is called sparse, and the opposite to that (a mostly filled matrix) is called dense.

- u.item: the list of movies
- u.data: the list of ratings given by users

The file u.data that contains the ratings is a tab separated list of user ID, item ID, rating, and timestamp. The first few lines of the file look like this:

user_id	item_id	rating	timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

As shown above, the file tells what rating a user gave to a particular movie. This file contains 100,000 such ratings, which will be used to predict the ratings of the movies not seen by the users.

Using this data along with the other ones, we will predict the ratings for movies based on user ratings from training dataset. Finally we will use a few metrics as shown in the above table to determine the effectiveness of our algorithm thus implemented.

Formulae used for rating calculations:

In the weighted average approach, you multiply each rating by a similarity factor(which tells how similar the users are). By multiplying with the similarity factor, you add weights to the ratings. The heavier the weight, the more the rating would matter.

The similarity factor, which would act as weights, should be the inverse of the distance discussed above because less distance implies higher similarity. With the similarity factor S for each user similar to the target user U, we can calculate the weighted average using this formula:

$$R_U = (\sum_{u=1}^{n} R_u * S_u) / (\sum_{u=1}^{n} S_u)$$

In the above formula, every rating is multiplied by the similarity factor of the user who gave the rating. The final predicted rating by user U will be equal to the sum of the weighted ratings divided by the sum of the weights.

Major Data-Structures used:

- df : pandas dataFrame
  - index (along the row): user
  - label (columns) : movies
- utility_matrix : 2d numpy array
  - row - user_id
  - Column - movie_id
- Users_map: { key, value }
  - key : user_no
  - value : iterator in the range(len(user_map))
- Movie_map: { key, value }
  - key : movie_no
  - value : iterator in the range(len(movie_map))

Singular Value Decomposition

Dataset used: ratings.dat from the movielens dataset

No of rows = 1682
No of columns = 983

- Each user corresponds to a single row indexed with user_id
- Each movie rated by user corresponds to a single column indexed with movie_id
- The users_movies matrix contains the ratings of movies given by corresponding users
- This matrix is sparse, i.e, there are many unrated movies for each user
- The unrated movies are initialised with 0's, and this is one of the criticism where 0 represents that user doesn't like the movie, but here it represents a missing , unrated value.
- To avoid this , we add global avg plus the row_means_bias and col_means_bias to each missing value for that particular row,column

Decomposition:

SVD for an MxN matrix is calculated by decomposing the matrix into 3 component matrices below

- U : Mxr column orthogonal matrix, r= rank of original matrix
- Sigma : rxr diagonal matrix which contains the singular values of the original matrix
- V : rxN column orthogonal matrix

Approximate_matrix = U*Sigma*V(Transpose)

Data Structures Used

- Loading data : Pandas Dataframe
- Utility_matrix : numpy array
- Row_means,col_means : array
- Num_movies,num_users : list
- U,S,Vt : numpy array

CUR Method and CUR with 80% - 90% energy

Utility matrix is divided into three matrices C, U, R
CUR details
- Rows - 50
- Columns - 50

Major Data Structure:
- Matrix C: An Mxr column orthonormal matrix where r is the rank of the original matrix.
- Matrix U: An rxr diagonal matrix containing the singular values of the original matrix.
- Matrix R: An rxN column orthonormal matrix where r is the rank of the original matrix.

The original matrix can then be approximated by calculating the product of these matrices as follows:

Approximation of original matrix = C * U * R T