A Profile Based Movie Recommendation System

Mr. Devam Shah
devam.s1@ahduni.edu.in

Ms. Nancy Radadia
nancy.r@ahduni.edu.in

Mr. Varshil Shah
varshil.s@ahduni.edu.in

Mr. Parth Sarkhelia
parth.s3@ahduni.edu.in

***Abstract*—Our project centers around building a movie recommendation system based on Collaborative filtering. In particular we look at implementing three algorithms of item based collaborative filtering which are K-Nearest Neighbours(KNN), Alternating Least Squares(ALS) and Singular Value decomposition (SVD) and finally compare their performance in terms of standard measures.**

***Index Terms*—Recommendation System, Machine Learning, Item-based Collaborative Filter, K Nearest Neighbours, Alternating Least Square, Singular Value decomposition**

I. Introduction

Recommendation systems are one of the most prominent applications of Machine Learning and part of everyday life. They are well-studied and proven to provide tremendous value to internet businesses and their consumers. Recommender systems can be loosely broken down into three categories: content-based systems, collaborative filtering systems, and hybrid systems, out of which we are going for item-based collaborative filtering systems. Item-based shared filtering recommends a product based on the browsing habits of other users who looked at the same item as me (users who looked at my item also looked at these other items). Further, Collaborative filtering systems can be further broken down into Memory-Based Collaborative Filtering and Model-based collaborative filtering. Model based algorithms uses the training data to create a model that can then be used to generate predictions while memory based collaborative filter just uses training data to generate a predictions.

We intend to implement one memory based algorithm and two model based algorithm. The algorithm implemented are K Nearest Neighbours (KNN), Alternating Least Square(ALS) and Singular Value decomposition (SVD) where KNN is memory based while ALS and Truncated Singular Value decomposition are model based

This project uses a movie-lens dataset for training the recommendation models. MovieLens dataset contains two files, movies.csv, and rating.csv. This dataset contains 9000 unique movies and 600 unique users. Total ratings in this dataset are around 300,000.

II. Literature survey

The literature search for the recommendation system was mostly based on descriptors "Collaborative filtering", "Contents filtering" and "Hybrid Recommendation Systems". Content-based filtering needs a lot of information about the item to recommend it, on the other hand, collaborative filtering doesn't need anything else except the user's historical preference on a set of items.

KNN algorithm is a simple implementation of collaborative filtering [1]. It is based on feature similarity, as to how closely a sample data point feature correlates with our training dataset, thus determining the classification of the sample data point.
Although KNN model has some limitations like it recommends a movie that is popular among similar users without allowing for personalization, also it has scalability issues.

The second approach is the Alternating least squares algorithm which is a matrix factorization technique that seeks to decompose the rating matrix into two lower dimensional matrices. This method uses the iterative approach. During each iteration, one of the factor matrix is held constant and another one is tuned using the least square method. For the next iteration, the newly solved matrix is held constant and another matrix is tuned. This runs in a loop until the loss function is minimized.

The rating matrix consists of the item as the column and users as the rows. So the values in the rating matrix can be interpreted as the ratings given by that specific row of users to the specific column of the movie. But as it's impossible that all the users have watched all the movies, so mainly this rating matrix is sparse. Hence, Singular Value decomposition [4] is another matrix factorization technique that reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension where K $\leq$ N, hence it decomposes the actual rating matrix into 3 lower dimensional matrices $U \sum V^T$.

III. Implementation

A. KNN Algorithm

When it comes to recommending items/movies, we are interested in recommending only top K items/movies to user [2]. KNN relies on labeled input data to learn a function that generates appropriate output for unlabelled data. Our first step is to clean the given dataset and reduce the number of user and movies while keeping dimension of rating constant and reshape the dataset into a format which can be given as parameters. Reducing the number of users and movie is necessary as it reduces sparsity and more importantly, KNN

doesn't work properly in case of a large number of dimension.

When a movie name is given as input, the KNN model selects movies from a dataset which have a fuzz ratio of more than 50 for the given input. It then calculates Euclidean distance for each selected movie for the input movie and selects K nearest movies with the smallest Euclidean distance as per calculation. Here, a simpler approach to select k is to set $K = \sqrt{n}$ where n is the number of data points in the training dataset(unique movies). From above K-nearest movies, we select top N movies which can be recommended to user.

B. ALS Algorithm

In collaborative filtering, matrix factorization is one of the solutions for sparse data problems. Alternating Least Square (ALS) [3] is also a matrix factorization algorithm that minimizes two loss functions alternatively obtained from matrix factorization. [5]

In matrix factorization, the main goal is to divide the Rating Matrix into two lower dimension matrix which are the User matrix and Item Matrix. This algorithm assumes that there are k attributes or features which define each user and similarly there are k attributes or features that define each movie. Multiplying each feature of the user by the corresponding feature of the movie and adding it together gives a reasonable estimate of the user's rating. Mathematically, we can represent the matrix factorization as follows-

$$r_{ui} = x_u^T \cdot y_i = \sum_k x_{uk} y_{ki} \tag{1}$$

where, u is number of users, i is number of item, r is rating matrix, x is user matrix, y is item matrix, k is latent factors. The loss function can be given as:

$$L = \sum_{u,i \in S} (r_{ui} - x_u^T \cdot y_i)^2 + \lambda_x \sum_u \|x_u\|^2 + \lambda_y \sum_u \|y_i\|^2 \tag{2}$$

Now, we aim to minimise above loss function by using ALS. During each iteration, ALS holds one set of latent vectors as constant and tune the other set of latent vectors in order to minimise loss. For the next iteration , the newly solved matrix is held constant and another matrix is tuned.
Let's assume at first we hold item vectors(y_i) constant.So we take derivative of loss function with respect to user vector(x_u). Differentiating equation (2) with respect to User vector(x_u) , by assuming item vector(y_i) as constant gives-

$$\frac{\partial L}{\partial x_u} = -2 \sum_i (r_{ui} - x_u^T \cdot y_i) y_i^T + 2\lambda_x x_u^T = 0$$

$$-(r_u - x_u^T \cdot Y^T)Y + \lambda_x x_u^T = 0$$

$$x_u^T (Y^T Y + \lambda I) = r_u Y$$

$$x_u^T = r_u Y (\ Y^T Y + \lambda_x I)\ ^{-1} \tag{3}$$

Similarly, we can obtain the derivation for the item vectors by assuming solved-user vectors as constant.

$$y_i^T = r_i X (\ X^T X + \lambda_y I)\ ^{-1} \tag{4}$$

C. SVD Algorithm

SVD is one of the matrix factorization algorithm which generates the matrices with the specified number of columns. Hence, SVD decreases the number of output and better works on the sparse matrices for features output. [6]
Here, we first decompose the high dimensional rating matrix into 3 lower dimensional matrices in such a way that their product gives an approximate rating matrix i.e.

$$A = U \sum V^T$$

In the above given equation, A is rating matrix and U is the user "features" matrix that shows the relationship between the user and the features. So matrix U gives intuitions that how much the particular user likes particular features i.e. if the user likes comedy and hates horror movies, the values in U will be high under the comedy column for that user and low under the horror column. The matrix V shows the relationship between the movie and latent factors. So matrix V gives intuitions that how much the particular movie has particular features i.e. if the movie is a comedy, the values in V will be high under the comedy column for that movie and low under the horror column. $\sum$ is the diagonal matrix consisting of singular values (essentially weights) that describes the strength of the latent factors.

If matrix A(rating matrix) is mxn rating matrix, U(user features matrix) will be mxm, V^T (movie features matrix) will be nxn and $\sum$ will be mxn diagonal matrix. The product of U, $\sum$ and V^T gives us the approximate value of rating matrix A.

As the rating matrix has higher sparse density, we use the truncated SVD to reduce the sparsity. Thus we reduce the dimensions of the rating matrix to K where K $\leq$ N in truncated SVD. The optimal way to calculate the value of K is using Root Mean Square Error (RMSE). We calculated the RMSE for different values of K and selected the optimal K with minimum RMSE value. Hence we implement truncated SVD to generate prediction rating matrix.

IV. Result

A. KNN Algorithm

Given below is the output of recommended movies to the user when a movie is given as an input string. The recommended movies seem similar to the given input movie.

```
recommendation('Avengers',item_user_matrix_sparse,
               model,movieToIndex,10)
```

```
Recommendation in progress...

Viewer who watches this movie  Avengers also watches following movies.
Iron Man 2 (2010)
Guardians of the Galaxy (2014)
Captain America: The Winter Soldier (2014)
X-Men: First Class (2011)
Iron Man 3 (2013)
X-Men: Days of Future Past (2014)
Avengers: Age of Ultron (2015)
Captain America: The First Avenger (2011)
Thor (2011)
Ant-Man (2015)
```

Fig. 1. KNN

Here, movie, item-user-matrix-sparse, model, movie-To-Index and N is passed as arguments to the recommendation function where movie is input movie string, item-user-matrix-sparse is list which consists non-zeros values of movie-id and user-id, model is K-NearestNeighbors, movie-To-index is mapper that maps movie index and its title as we have two different files for movie title and movie index and finally N here is 10 which is top N movies recommended for user.

B. ALS Algorithm

In the ALS algorithm, the recommended movies are generated based on the previous preferences and ratings given by the user.
The given figure depicts the list of the movie which are preferred by user-id 50.

50th User's Actual Preference:

```
ratings.join(movies, on='MovieID').filter('UserID = 50')
.sort('Rating', ascending=False).limit(10).show()
```

```
+-------+------+------+--------------------+--------------------+
|MovieID|UserID|Rating|               Title|              Genres|
+-------+------+------+--------------------+--------------------+
|   1251|    50|   4.5|   8 1/2 (8½) (1963)|       Drama|Fantasy|
|    924|    50|   4.5|2001: A Space Ody...|Adventure|Drama|S...|
|   1204|    50|   4.5|Lawrence of Arabi...| Adventure|Drama|War|
|   1208|    50|   4.5|Apocalypse Now (1...|    Action|Drama|War|
|   1136|    50|   4.0|Monty Python and ...|Adventure|Comedy|...|
|    903|    50|   4.0|      Vertigo (1958)|Drama|Mystery|Rom...|
|   1199|    50|   4.0|       Brazil (1985)|      Fantasy|Sci-Fi|
|    750|    50|   4.0|Dr. Strangelove o...|          Comedy|War|
|   1201|    50|   4.0|Good, the Bad and...|Action|Adventure|...|
|    899|    50|   4.0|Singin' in the Ra...|Comedy|Musical|Ro...|
+-------+------+------+--------------------+--------------------+
```

Fig. 2. Preferences of user

The below figure depicts the list of movie which are recommended to user-id 50 based on their preference and rating.

50th User's ALS Recommentions:

```
recommendations.join(movies, on='MovieID').filter('UserID = 50').show()
```

```
+-------+------+---------+--------------------+--------------------+
|movieId|userId|   rating|               Title|              Genres|
+-------+------+---------+--------------------+--------------------+
|   3379|    50|3.8968225| On the Beach (1959)|               Drama|
|  96004|    50|3.8558898|Dragon Ball Z: Th...|Action|Adventure|...|
|  27156|    50|3.7518673|Neon Genesis Evan...|Action|Animation|...|
|  92475|    50|3.7072878|All Watched Over ...|         Documentary|
|   2511|    50|3.7047849|Long Goodbye, The...|     Crime|Film-Noir|
|   8477|    50|3.6856701|    Jetée, La (1962)|      Romance|Sci-Fi|
|   7767|    50|3.6633036|Best of Youth, Th...|               Drama|
|   3224|    50| 3.658248|Woman in the Dune...|               Drama|
|   5490|    50|3.6580172|  The Big Bus (1976)|       Action|Comedy|
|   7096|    50|3.6565723|Rivers and Tides ...|         Documentary|
+-------+------+---------+--------------------+--------------------+
```

Fig. 3. Recommendation for user

C. SVD Algorithm

In the SVD algorithm, the recommended movies are generated based on the previous preferences and ratings given by the user.
The given figure depicts the list of the movie which are preferred by user-id 50.

```
# The top 10 movies rated by the user
previous_rated.head(10)
```

	userId	movieId	rating	title
21	50	924	4.5	2001: A Space Odyssey (1968)
33	50	1204	4.5	Lawrence of Arabia (1962)
35	50	1208	4.5	Apocalypse Now (1979)
40	50	1251	4.5	8 1/2 (8½) (1963)
121	50	7327	4.0	Persona (1966)
28	50	1136	4.0	Monty Python and the Holy Grail (1975)
31	50	1199	4.0	Brazil (1985)
32	50	1201	4.0	Good, the Bad and the Ugly, The (Buono, il bru...
39	50	1232	4.0	Stalker (1979)
41	50	1252	4.0	Chinatown (1974)

Fig. 4. Preferences of user

The below figure depicts the list of movie which are recommended to user-id 50 based on their preference and rating.

```
# The recommended movies to the user
recommended_movies
```

	movieId	title
8773	141718	Deathgasm (2015)
5547	27751	'Salem's Lot (2004)
8623	134095	My Love (2006)
8624	134109	Radio Day (2008)
3572	5059	Little Dieter Needs to Fly (1997)
6342	53578	Valet, The (La doublure) (2006)
8640	134796	Bitter Lake (2015)
8642	134847	Ghost Graduation (2012)
6330	53355	Sun Alley (Sonnenallee) (1999)
6326	53280	Breed, The (2006)

Fig. 5. Recommendation for user

D. Performance Comparison

UserID	MovieID	Rating	prediction	Rating_binary	predictions_binary	Truth
385	471	4.0	3.2199137	1	1	1
462	471	2.5	2.49348	0	0	1
387	471	3.0	2.9785595	1	0	0
171	471	3.0	4.435947	1	1	1
32	471	3.0	3.7408144	1	1	1
469	471	5.0	3.454264	1	1	1
357	471	3.5	3.9528108	1	1	1
132	1088	4.0	2.7562292	1	0	0
563	1088	4.0	3.3933873	1	1	1
594	1088	4.5	4.1311927	1	1	1
307	1088	3.0	2.3777387	1	0	0
51	1088	4.0	3.604006	1	1	1
221	1088	3.0	3.0407107	1	1	1
414	1088	3.0	3.0470037	1	1	1
200	1088	4.0	3.690107	1	1	1
104	1088	3.0	3.6271515	1	1	1
19	1238	3.0	3.203387	1	1	1
156	1238	4.0	4.0440884	1	1	1
425	1342	3.5	2.22978	1	0	0
600	1342	2.5	2.0980847	0	0	1

only showing top 20 rows

```
[ ]  accuracy=b[0]/(b[0]+b[1])
     print(accuracy*100)

     77.35421388831651
```

Fig. 6. Performance Analysis of ALS

	userId	movieId	rating	predictions	predictions_binary	original_ratings_binary
97652	606	1663	2.0	3.986243	1	0
25050	177	1213	4.0	3.778595	1	1
57225	380	2746	4.0	4.397775	1	1
65306	419	288	1.5	3.797822	1	0
38488	264	3034	2.5	3.773193	1	0

```
1  print(accuracy)

87.09464149945457
```

Fig. 7. Performance Analysis of SVD

As you can see in above figure, performance of ALS is 77% while performance od SVD is 87% Hence, SVD works better than ALS.

V. Conclusion

- KNN algorithm is simple to implement which do not derive any function from the training data.The algorithm is versatile as can be used for classification, regression, and search.

- Our model recommends a movie that is almost released in similar years. Since we have removed unpopular movies from our dataset, this movie will never be recommended to the user. Also, KNN algorithm is not precise for recommendation as it doesn't deduce proper inference with large dimension labelled data.

- ALS deduce proper recommendations of movies even from the large datasets and it recommends the movie based on personalization and taste of user's genres.

- ALS can overcome some of the problems like Bias recommendation based on popularity, Cold Start problem and Scalability of users.Matrix factorization increases the ability of recommendation model to recommend lesser-known films.

- SVD can automatically reduce the dimensions of the large dataset and thus designing a predictive model with greater accuracy which works better with unseen raw data.

- From the performance analysis of SVD and KNN, we can say that in matrix factorization SVD performs better than ALS which implies that SVD performs the best of all three algorithms.

References

[1] Harrison, Onel. "Machine Learning Basics with the K-Nearest Neighbors Algorithm." Medium, Towards Data Science, 14 July 2019, towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[2] Gupta, Meenu, et al. "Movie Recommender System Using Collaborative Filtering." 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC). IEEE, 2020.

[3] Li, Jung-Bin, et al. "Implementation of an Alternating Least Square Model Based Collaborative Filtering Movie Recommendation System on Hadoop and Spark Platforms." International Conference on Broadband and Wireless Computing, Communication and Applications. Springer, Cham, 2018.

[4] Rajarajeswari, S., et al. "Movie Recommendation System." Emerging Research in Computing, Information, Communication and Applications. Springer, Singapore, 2019. 329-340.

[5] M Hendra Herviawan.hendra-herviawan.github.io/build-movie-recommendation-with-apache-spark.html.

[6] Kumar, D. (2021, January 12). Singular value DECOMPOSITION (SVD) in recommender system. Retrieved April 11, 2021, from https://analyticsindiamag.com/singular-value-decomposition-svd-application-recommender-system/