

Summer Project: Movie Recommendation System

UNDER THE GUIDANCE OF MR. AMIT SHRIVASTAVA

Shradha Shankar | 24th July, 2017

1. Dataset

1.1 DESCRIPTION

The dataset used is a version of the MovieLens dataset. It describes 5-star rating and free-text tagging activity from [MovieLens](http://movielens.org), a movie recommendation service. It contains 100004 ratings and 1296 tag applications across 9125 movies. These data were created by 671 users between January 09, 1995 and October 16, 2016.

The data are contained in the files `links.csv`, `movies.csv`, `ratings.csv` and `tags.csv`.

tags.csv: All tags are contained in the file `tags.csv`. Each line of this file after the header row represents one tag applied to one movie by one user, and has the following format: `userId, movieId, tag, timestamp`

movies.csv: Movie information is contained in the file `movies.csv`. Each line of this file after the header row represents one movie, and has the following format: `movieId, title, genres`

ratings.csv: All ratings are contained in the file `ratings.csv`. Each line of this file after the header row represents one rating of one movie by one user, and has the following format: `userId, movieId, rating, timestamp`. Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

links.csv: Identifiers that can be used to link to other sources of movie data are contained in the file `links.csv`. Each line of this file after the header row represents one movie, and has the following format: `movieId, imdbId, tmdbId` where `imdbId` stands for the id of the movie on IMDB and `tmdbId` stands for the id of the movie on themoviedb.org.

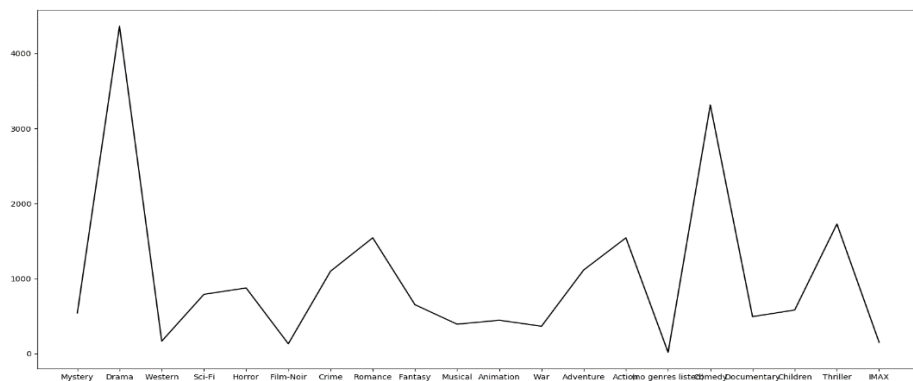
1.2 EXPLORATORY ANALYSIS

The movies in the dataset belong to one or more of the following genres:

- * Action
- * Adventure
- * Animation
- * Children's
- * Comedy
- * Crime
- * Documentary
- * Drama
- * Fantasy
- * Film-Noir

- * Horror
- * Musical
- * Mystery
- * Romance
- * Sci-Fi
- * Thriller
- * War
- * Western
- * (no genres listed)

The following plot shows the number of movies belonging to a particular genre.



4364 movies can be categorized as Drama while 17 movies have no genres listed.

59 movies have not been rated by any user and 1 movie has neither been rated nor has any genre listed.

These 59 movies have been removed from consideration in the implementation of recommender algorithms.

2. Recommender Algorithms

The project aims to build a movie recommendation system on the above described dataset. Experiment with two techniques is described further. Python has been used for programming in all experiments.

2.1 ALSO BOUGHT ALGORITHM

The also bought algorithm is basically an item based collaborative filtering algorithm and was first put to use by Amazon. An item based collaborative filtering model uses similarities between items to make recommendations. It

performs better than a user based model as there are generally fewer items than users and hence computation of similarity between each item pair is faster. The original algorithm as described by Amazon is as follows: -

```
For each item in product catalogue, I1
  For each customer C who purchased I1
    For each item I2 purchased by customer C
      Record that a customer purchased I1 and I2

For each item I2
  Compute the similarity between I1 and I2
```

In a movie recommendation system, the term 'also bought' may not be appropriate and hence 'also viewed' is used.

2.1.1 Implementation

We assume that a movie being rated by a user is equivalent to the movie being viewed by the user. Hence, according to this assumption, movies that have not been rated by the user have not been watched by him.

First, a subset of the movies.csv file is taken by removing the movies that have not been rated by any user.

The first portion of the algorithm essentially calculates the co-occurrence of two items. A Boolean vector is created for each user. The vector is formed by the movie ids as vector components. The vector component is 1 if the user has rated the movie and 0 otherwise. Thus, a user-item matrix is created. Computing the product of this matrix with the transpose of itself gives us the final co-occurrence count item-item matrix.

The co-occurrence count item-item matrix is normalized by using similarity measures to generate similarity matrix. `similarity_calc.py` uses cosine similarity for computation while `pearson_sim_calc.py` uses Pearson coefficient. The cosine similarity is computed using the `cosine_similarity()` function from `sklearn.metrics.pairwise` library. The Pearson similarity is computed using `corrcoef()` function from `numpy`. After creating the similarity matrix, the scripts extract the 10 (number can be changed) most similar movies for every movie stores them in a csv file which is used for recommendations. This algorithm is computes similarities offline.

For a particular (userId, movieId) pair, the system retrieves the most similar movies for the given movie and outputs only those movies that have not been rated by the user in consideration.

2.1.2 Input

Since the dataset only the user id as the information about the user, a user id and movie name is given to the system as input. The system uses the movie id for the given movie name to generate recommendations.



This is similar to the situation where a user logs into his profile on the Amazon website and looks at the description of a particular item. The item in this case is analogous to the movies in our system.

2.1.3 Output


The recommendations given by also bought algorithm for two users on Amazon.in looking at Nutella Hazelnut Spread with Cocoa 290g – Despicable Me 3 Limited Edition Pack is given below:

User 1:


Customers who bought this item also bought




Hershey's Chocolate Syrup, 623g
★★★★☆ 176
₹ 190.00 ✓prime



Nutella Hazelnut Spread with Cocoa, 350g
★★★★☆ 269





Betty Crocker Breakfast Pancake Mix Original, 1KG
★★★★☆ 55
₹ 370.00 ✓prime




Betty Crocker Pancake Mix, Original 500g
★★★★☆ 166
₹ 149.00 ✓prime

User 2:


Customers who bought this item also bought




Hershey's Chocolate Syrup, 623g
★★★★☆ 176
₹ 190.00 ✓prime



Nutella Hazelnut Spread with Cocoa, 350g
★★★★☆ 269



Betty Crocker Breakfast Pancake Mix Original, 1KG
★★★★☆ 55
₹ 370.00 ✓prime



Betty Crocker Pancake Mix, Original 500g
★★★★☆ 166
₹ 149.00 ✓prime

From the results, we can conclude that also bought outputs items from a subset of similar items for a given item and hence there is not much variation in the recommendations for different users.

Given next are the recommendations for two users for the movie ‘Sudden Death (Action 1995)’:

User id 254:

```
output_pear_254_9.txt - Notepad
File Edit Format View Help

movieId      title      genres
559 640 Diabolique (1996) Drama|Thriller
movieId      title      genres
2 3 Grumpier Old Men (1995) Comedy|
Romance      movieId      title
genres
602 711 Flipper (1996) Adventure|Children
movieId      title      genres
69 74 Bed of Roses (1996) Drama|Romance
movieId      title      genres
70 76 Screamers (1995) Action|Sci-Fi|
Thriller      movieId      title
genres
73 79 Juror, The (1996) Drama|Thriller
movieId      title      genres
650 786 Eraser (1996) Action|Drama|Thriller
movieId      title
genres
340 376 River Wild, The (1994) Action|
Thriller
```

User id 110:

```
output_pear_110_9.txt - Notepad
File Edit Format View Help

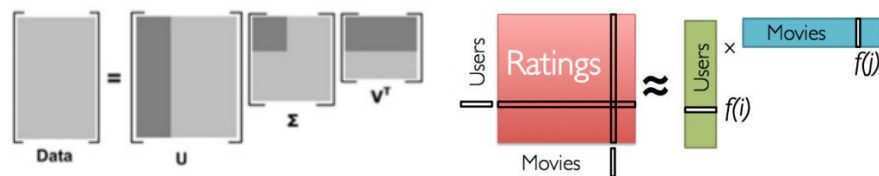
movieId      title      genres
559 640 Diabolique (1996) Drama|Thriller
movieId      title \
536 609 Homeward Bound II: Lost in San
Francisco (1996)
genres
536 Adventure|Children      movieId
title      genres
602 711 Flipper (1996) Adventure|Children
movieId      title      genres
69 74 Bed of Roses (1996) Drama|Romance
movieId      title      genres
70 76 Screamers (1995) Action|Sci-Fi|
Thriller      movieId      title
genres
340 376 River Wild, The (1994) Action|
Thriller
```

Consistent with the output from Amazon's also bought recommendations, we notice that most movies that feature in the recommendations for the users are the same. The movies causing the difference do not appear in the other user's recommendations as he may have already rated them. For example, Homeward Bound II: Lost in San Francisco (1996) has not been recommended to user id 254 as he has already viewed it.

2.2 SINGULAR VALUE DECOMPOSITION (SVD) FOR RECOMMENDATION

SVD in the context of recommendation systems is used as a collaborative filtering (CF) algorithm. Collaborative filtering is a method to predict a rating for a user item pair based on the history of ratings given by the user and given to the item. Most CF algorithms are based on user-item rating matrix where each row represents a user, each column an item. The entries of this matrix are ratings given by users to items.

SVD is a matrix factorization technique that is usually used to reduce the number of features of a data set by reducing space dimensions from N to K where $K < N$. For the purpose of the recommendation systems however, we are only interested in the matrix factorization part keeping same dimensionality. The matrix factorization is done on the user-item ratings matrix. From a high level, matrix factorization can be thought of as finding 2 matrices whose product is the original matrix.



Let 'Data' be an $N \times M$ matrix. Using SVD, it is decomposed into U , Σ and V . U is an $N \times K$ matrix, Σ is a $K \times K$ matrix and V is an $M \times K$ matrix. Σ is a diagonal matrix with values decreasing along the diagonal. The product of U and V^T approximates the original matrix.

Using facts from SVD, the user-movie matrix having dimensions $U \times M$ can also be decomposed into two matrices- users ($U \times K$) and movies ($M \times K$). The value of K defines the number of latent features. Each entry in the user matrix can then be assumed as the affinity of a user for a given latent feature. Likewise, each entry in the movies matrix can be thought of as the amount a latent feature affects a movie. For example, for a user-movie pair

(Bob, Titanic) and a feature 'drama', the user matrix entry (Bob, drama) indicates how much Bob likes drama and the movie matrix entry (Titanic, drama) is a measure of the drama factor in Titanic. The product of users and transpose of movies matrices gives a matrix that is the approximation of the original user-item matrix. For each user, the approximation assigns ratings to the movies that the user has not rated and hence predicts the ratings for such movies.

2.2.1 svds function in scipy library in Python

The svds function in Python returns U , σ and V^T as output for the SVD of a given matrix. The matrix to be decomposed and the number of latent features are passed to the function. The function can be imported from `scipy.sparse.linalg`.

The implementation using this function using 100 latent features did give some good quality recommendations (about 10 correct from top 3 result pages) but the ratings for a given user movie pair were far from accurate. The recommendations were matched with the recommendations of movielens.org by creating sample profiles with rating history similar to two test users.

2.2.2 Netflix Prize – Simon Funk's algorithm

The Netflix Prize was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films, i.e. without the users or the films being identified except by numbers assigned for the contest. Submitted predictions were scored against the true grades in terms of root mean squared error (RMSE), and the goal is to reduce this error as much as possible.

Simon Funk is the inventor of a simple & ingenious SVD algorithm during the Netflix contest. Simon Funk's description includes proposition of learning rate and regularization constants, and a method of clipping predictions. His algorithm uses gradient descent to find the matrices after decomposition. To minimize the error between the SVD-like model and the original data, the derivative of the function is taken with respect to the parameters to be inferred. For a model to be able to generalize well and not over-fit the training set, a penalty term is introduced into the minimization equation. This penalty term is called regularization parameter.

2.2.3 Implementation

The dataset is split into training and test datasets using the holdout method for cross validation. 80% of the data belongs to the training set.

Relevant files from the train set are used to form the user-movie matrix.

A baseline predictor is the prediction matrix that is used as a starting point for comparisons. A baseline predictor is generated using the following equation:

If y_{ij} is the original entry in the user-movie matrix, then,

$$\text{pred } y_{ij} = \text{user_bias}(i) + \text{movie_bias}(j) + \text{global_mean}$$

$\text{user_bias}(i)$ is the bias of the user i and is calculated by taking the average of all the ratings given by him.

$\text{movie_bias}(i)$ is the bias of the movie m and is calculated by taking the average of all the ratings given to a movie.

global_mean is the average of all the ratings in the user-movie matrix.

The gradient descent model is then trained according to the following equations:

$$r_{ij} = y_{ij} - \hat{y}_{ij}$$

$$u_{ik} += \text{lr} * (r_{ij} v_{jk} - \lambda u_{ik})$$

$$v_{jk} += \text{lr} * (r_{ij} u_{ik} - \lambda v_{jk})$$

where y_{ij} is the rating given by user i for movie j .

$$c_i += \text{lr} * (r_{ij} - \lambda_2(c_i + d_j - \text{global_mean}))$$

$$d_j += \text{lr} * (r_{ij} - \lambda_2(c_i + d_j - \text{global_mean}))$$

where c_i is the bias weight for user i and d_j is the bias term for movie j .

$$\hat{y}_{ij} = c_i + d_j + u_i^T v_j$$

RMSE is used for error calculation. Error is minimized on the test set since other values are unknown.

Suitable values for lr , λ and λ_2 need to be determined.

2.3 FUTURE WORK

A mechanism to predict ratings for the recommendations of also bought needs to be devised.

After appropriate constants in the SVD approach have been determined, the output of the also bought algorithm can be blended with outputs from the gradient descent approach.

A few more models can be experimented with and the results can be blended. Quality of recommendations can be improved by considering more features like location. They can be made more dynamic by considering the change in a user's rating for a movie with time.

More approaches like clustering can be applied and the results can be compared. Different metrics for error calculation can also be considered.