Name: Abul Shariff & Venkata Pratyush Kodavanti

HW4: Movie Recommender System

Latest Submission RMSE: 0.90

Miner names: vkodavan and datamining002

The problem statement involves creating a movie recommender system with provided data. The data files contain train and test sets about the movie and users who have rated some of the movies and also include side information for creating a profile for a user. The objective of this movie recommender system is to predict the 5-star rating a movie will get for a given user.

Preprocessing Phase

1) In the preprocessing phase we wanted to create a user-item matrix. We converted our training set to a sparse matrix. We used users-id (x) as the rows and movie-id (y) as the columns and the rating of the movie by the user as the value (matrix[x][y]) , similarly we created same movie genre with movie-id as the row and genres as the columns and if the movie is of that genre we had set it 1 or else zero(same as one hot encoding).

2) To read the .dat files we used basic file reading functions in python.

User Movie Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
75	0.0	0.0	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	0.0	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
78	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	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	4.5	0.0
127	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	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	0.0	0.0
170	3.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.5	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
175	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	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
71487	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0
71497	0.0	3.5	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	3.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71509	4.0	0.0	0.0	0.0	1.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.5	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0
71525	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	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	0.0	0.0
71529	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	0.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2112 rows × 9936 columns

Movie Genre Matri

Movie_ID	Sci-Fi	Short	Western	Comedy	Mystery	Thriller	Children	Animation	Fantasy	Romance	Drama	Documentary	Action	Crime	Horror	War
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
65088	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
65091	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
65126	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
65130	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
65133	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

10197 rows × 20 columns

Approaches

1) User Based Collaborative filter

We used user based collaborative filtering for calculating the similarity of the given user to all the users in the training data. Our methodology revolved around the fact that we should only pick users who are similar to the test-user, hence we would get all the users who had rated the test-movies and then find the most similar. To calculate the similarity measure we tried Euclidean distance, Cosine similarity and Cosine Distance but each of them were time consuming. We found that the pearson correlation was the fastest of all. To calculate the Euclidean distance, cosine distance and cosine similarity we used scikit learns library, sklearn.metrics.pariwise,

a) Euclidean distance: which calculates the Euclidean distance between two given vectors, it takes 2 parameters (X, Y)

b) Cosine Similarity: Which calculates the angle of similarity between two vectors. It takes 2 parameters X and Y where X and Y are n dimensional numpy arrays.

c) Cosine Distance is the 1- cosine similarity. 2 parameters X and Y n dimensional numpy array

d) For the Pearson correlation we used scipy.stats.pearsonr. The Pearson correlation between 2 values is always between +1 and -1. A value of +1 and -1 implying the exact linear relationship between 2 values, 0 means no relation between the values.

After calculating the similarity we select the nearest values in the max similarity of 11 users and average their rating. For unseen movies and users we hardcoded a rating of 2.5, which gave us an RMSE of 1.12.

For new movies and users

a) For cold start users, we took the movie genre and converted it into a vector using TFIDF Vectorizer from sklearn.feature_extraction.text. This converts the corpus to vectors. To be precise we used 2 methods of the TFIDF class fit() which learns vocabulary and inverse document frequency of the training set(list of all genres) and fit_transform() on the test set (all the genres of that particular movie). After vectorizing the genres we calculated the cosine distance for each genre of the movie to the list of all the genres. Select the nearest 9 values. We compute the average of each genre and then again compute the average of the average 9 values.
b) For cold start movies, we took the movie genre and checked all the genres of the movies that the user has rated. Converted them into vectors and then computed their cosine distance and then selected the 9 nearest genres average the rating of all the movies in those genres and return it.

2) we also used tag and tag weights. We selected all the tags and then took all the movies genres of the top 3 tag weights of the movie but this did not improve the RMSE.

3) Finally, we have taken genres of the movies and then took the average rating that the user has given to the genre, if none of the movies in the genre were rated then we assumed a rating of 1.0 and a rating of 2.5 for unseen movie by doing so we got an rmse of 0.90

Results

With the first approach we have got an rmse score between 1.12 to 0.98

But one of the drawbacks of this method is that each user's ratings are different i.e there is no hard measure for 2 users a rating of 4 is the same. To resolve this we subtracted the average resting of each user from all the movies the user has rated. By doing so the rmse went up to 1.38.

The second approach didn't improve our RMSE score.

The third approach gave us a RMSE of 0.90

For validation we have taken one movie for each user in the training set thereby giving us a total of 2113 records for validation data. To compute the rmse we used sklearns means squared error which takes 2 parameters prediction and actual values. There by computing its square root to get the rmse.