Movie Recommendation System Using Deep Neural Network Based on historical Rating Records

Team Members: Jiawei Yu, Yuqi Wang, Dantong Chen

Abstract

In this project, we built a movie recommendation system using Deep Learning method. We adopted a deep neural network to predict the rating given by a specific user on a movie and then added one additional input layer to this network. The new inputs are user embedding, movie embedding, and movie genre embedding. Then, for a given user, we found a set of movies which haven't rated by this user and predicted ratings on these movies by this user. Finally, we will recommend top 5 movies with 5 highest predicted ratings to the user. In addition, we compared the performance of our model with the original model on the MovieLens dataset and found out our model slightly decreased both mean absolute error and mean squared error by adding one input layer, movie genre.

1. Introduction

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user and make suggests based on these preferences[1]. It has been widely used in the industry of e-commerce. Compared to a bad, outdated recommendation system, a good recommendation system will improve the efficiency of advertising so that the number of sales and revenue will effectively increase. From the consumers perspective, people can hardly find perfect items they want every time and it's extremely time-consuming to search by themselves since there are too many different items nowadays. Therefore, we want to build a recommendation system which could recommend appropriate items to specific users. More specifically, we will focus on a movie recommendation system in this project.

2. Data

The data we used describing 5-star rating and free-text tagging activity from MovieLens. The dataset includes 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users from March 29, 1996 to September 24, 2018 and was updated on September 26, 2018. In the datafile, users were randomly selected. All selected users had rated at least 20 movies. No demographic information is included. In the datafile, users were randomly selected. All selected users had rated at least 20 movies. No demographic information is included[4].

We used the following 3 datasets:

- ratings.csv: userId, movieId, rating, timestamp
- tags.csv: userId, movieId, tag, timestamp
- movies.csv: movieId, title, genres

3. Related Work

In recent years, as one of the most popular applications of machine learning technologies in e-business, movie recommendation system has been greatly improved and spreading out widely. There are 2 different types of algorithms for movie recommender commonly used by people in industry.

3.1 Content-based Filtering

As the easiest and simplest algorithm to understand, the content-based movie recommender[3] only depends on the similarity of the items. For example, if user A likes movie M, then user A will also like movies which are similar to movie M. Additionally, people can cluster similar users. Based on users' information, movie recommender will recommend movie M to user A when movie M is liked by similar users of user A. It works generally well when it's easy to determine the genre and context of each movie or the representative characteristics of each user. However, we don't usually have much personal information about users.

3.2 Collaborative Filtering

Instead of focusing on content or property of items, collaborative filtering methods calculate similarity from the past interactions. For instance, if user A likes movies M, N, K and user B likes movies M, N, J, then they have quite similar interests. Thus, it is possible that user A likes movie J and user B likes movie K. For most of collaborative filtering methods, it is necessary to build a user-item matrix where each row represents a user, each column represents an item (movie), and each cell represents an interaction (rating) between a user and an item. The simplest algorithm computes cosine or correlation similarity of rows (users) and recommends movies that k-nearest neighbors enjoyed[3].

A more complicated and popular method is matrix factorization[2][3], which reduces the dimensionality of the original sparse user-movie matrix by approximately factorizing it into two or more small matrices embedded with a certain number of latent components. So the model can learn a low-dimensional representation of users and movies. While predicting the rating, people can simply take dot product of the user embedding and the movie embedding.

Then people apply deep learning method[2][3] with similar idea to predict ratings. For deep learning methods, those low-rank matrices are not necessary to be orthogonal since the neural network will learn the values of embedding matrices itself and use them as training features. Thus, people can choose two different numbers of latent components for user embedding and movie embedding. In addition, the model doesn't necessarily combine user embedding and movie embedding with dot product so that the neural network can learn more complex nonlinear relationships between two embeddings.

4. Methods

In order to give movie recommendations to a specific user, we first need a model to predict the rating on a movie given by this user.

In this project, we first adopted a deep neural network model[2] from other's previous work to predict ratings and then revised the model by adding one additional input layer. The original model, as the last method mentioned in 3.2 Collaborative Filtering, embedded the user and movie matrix and treat those as inputs to the network. For our new model, the network embedded the user, movie, and movie genre matrix and treat all those as inputs.

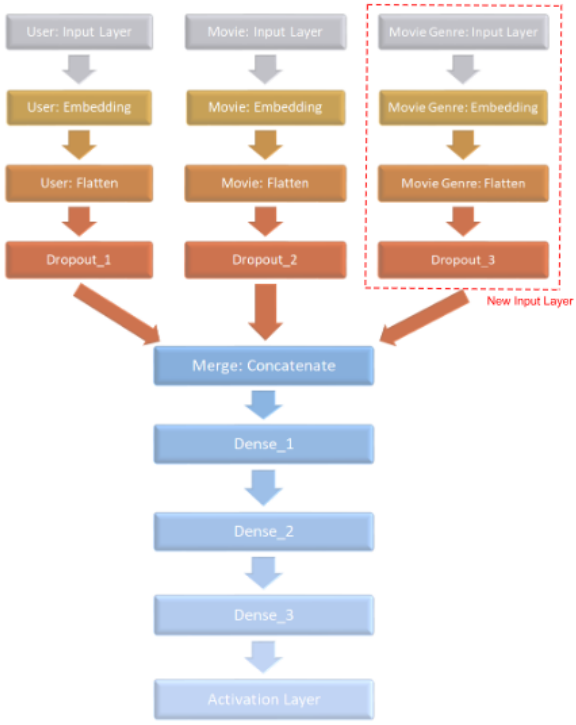

Fig 1. Our neural network

we noted that the original rating prediction was a classification problem since we can classify a predicted rating into total of 10 classes from 0.5 to 5. However, we decided to change the rating prediction into a regression problem since, in order to build a movie recommender, we only want to predict the level of preference on a movie by this user instead of exact rating score. For example, user A will rate 5 to both movie M and movie N. But it is highly likely that user A thinks movie M and N are both good movies but he or she still has a preference for either movie M or movie N.

After we had the model to predict the rating (or the level of preference), we first need to identify a user X to whom we want to give recommendations. Then, we found a list of movies which haven't rated by user X in our movie database and predicted ratings on this list of unseen movies by user X. After sorting all predicted ratings, we would recommend top 5 movies for user X.

5. Results

The mean absolute error and mean squared error for the model which only uses the embedding user id and movie id are 0.66702 and 0.76486 respectively. And our new model which uses the embedding user id, movie id and movie genre has the mean absolute error and mean squared error are 0.65754 and 0.75325 respectively. The performance of the model on the test dataset shows in the following figure:

	userId	movieId	rating	timestamp	title	genres	Predicted
60347	18	1355	3.0	965706276	Zero Effect (1998)	731	3.20989
87019	198	73	3.0	940544216	Things to Do in Denver When You're Dead (1995)	769	2.53266
23354	302	2096	5.0	1053302810	Airplane! (1980)	634	4.01753
18496	560	1053	3.0	1491095552	Star Trek: The Motion Picture (1979)	508	2.70716
89535	367	1191	2.0	975828061	Cop Land (1997)	237	2.84386

The figure only shows the top 5 rows of the test dataset, and the predicted rating errors are all blow 1.0, so we could conclude that the prediction performance is pretty good.

Then, we built the movie recommendation function which takes a user id as the input and our pre-trained model would predict ratings of movies which the user hasn't seen. And then the function will return top 5 movies with high ratings.

For example, the user 55 has rated those movies with 5 stars:

	userId	movieId	rating	timestamp	title	genres
388	55	43	5.0	835799219	Seven (a.k.a. Se7en) (1995)	937
2458	55	314	5.0	835799274	Forrest Gump (1994)	690
26001	55	302	5.0	835799081	Ace Ventura: Pet Detective (1994)	634
27757	55	18	5.0	835799219	Ace Ventura: When Nature Calls (1995)	634
4336	55	510	5.0	835799139	Silence of the Lambs, The (1991)	786

And our recommendation system would recommend those movies for the user 55:

	userId	movieId	rating	timestamp	title	genres
388	55	43	5.0	835799219	Seven (a.k.a. Se7en) (1995)	937
2458	55	314	5.0	835799274	Forrest Gump (1994)	690
26001	55	302	5.0	835799081	Ace Ventura: Pet Detective (1994)	634
27757	55	18	5.0	835799219	Ace Ventura: When Nature Calls (1995)	634
4336	55	510	5.0	835799139	Silence of the Lambs, The (1991)	786

Compared those two figures, we could conclude that the model will recommend some movies with the same genres.

6. Discussion of Results

Compared to the original model, we added the movie genre as a new input layer and the mean absolute error and mean squared error both decrease after we add the movie genre layer for the model. And the result supports that the system would recommend some movies with the same genres.

7. Conclusions

After we added the movie genre layer, we found out that adding new input embedding layer could improve the prediction performance. Therefore, in the future, we may add more information such as user age and gender to the model to see whether it could have better prediction performance.

References

[1] Prateek Sappadla, Yash Sadhwani, Pranit Arora, "Movie Recommender System", Spring 2017, NYU Courant, https://pdfs.semanticscholar.org/767e/ed55d61e3aba4e1d0e175d61f65ec0dd6c08.pdf

[2] Nipun Batra, “Recommender Systems in Keras”, https://nipunbatra.github.io/blog/2017/recommend-keras.html

[3] James Le, “The 4 Recommendation Engines That Can Predict Your Movie Tastes”, April 22, 2018, https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223

[4] Datasets: https://grouplens.org/datasets/movielens/