The Final Report of a Recommendation System for Movielens

YANG Rongfeng, LAU Kwan Yuen, WANG Xueying, LAI Wendi, Li Yunli
MSBD 5001, Group FMVP, BDT, HKUST

Abstract

With the rapid development of machine learning and related technologies in the past few years, many problems that are difficult to solve with traditional algorithm ideas have made breakthroughs. Among them, the fields of recommendation system have very large development and obtain rich research results with the introduction of the neutral network. In this paper, we focus on building a recommendation system for movies by exploring different solutions including neutral network models. With constant parameter modification and model improvement, we analyze our built system and discuss more specific details for possible exploration.

Key Words: Neutral Network, DeepFM, Recommendation System

1 Introduction

1.1 Background

The recommendation system is one of the most widely researched and most important applications for machine learning. The recommendation system helps users to deal with information overload and provide personalized recommendations, content, and services to them. The task of recommendation system is to link users to information, on the one hand to help users find information that is valuable to them, and on the other hand to present information to user who might be interested in it, to achieve a win-win situation for information consumers and information producers.

In the recommendation algorithm, the input parameters are various attributes and features of the user. After being processed by the recommendation algorithm [1], a list of items sorted according to the user's preference is returned. Common recommendation algorithms can be roughly divided into the following types: popularity-based algorithm, collaborative filtering algorithm, content-based algorithm, model-based algorithm and hybrid algorithm. In our project, we apply a user-based collaborative filtering approach to obtain the input of next module.

In addition, another core question in recommendation system is the prediction of click-through rate. The CTR estimate is a prediction of each user's clicks, predicting whether a user will click on a collection of contents or not. With the rapid development of deep learning, some classic methods and innovative directions have emerged in the research field of predicting user click-through rate.

1.2 Project Description

Our team proposes to build a recommendation system using data set provided by Movielens. Based on the data set, we train the neutral network model to predict movies that the user prefers by analyzing the previous data and then we recommend them to the user. We also build a web interface to demonstrate our work. The article is organized as follows: we first introduce the background of project and describe the project goal under individual contributions of each group member (§ 1). Then at the (§ 2), we discuss the related work to our project. We describe our work on the phase of data preparation (§ 3) and discuss the architecture of our recommendation system based on it (§ 4). In addition, we show the detailed experimental process with model evaluation and obtained experimental results (§ 5). Based on the above work, finally we do a briefly analysis and make a conclusion.

The core problem we explore is selecting several films to recommend to the users according to the movies they have seen. We extract features from the user contributed content including tags, ratings, and textual reviews. Each movie can be described as a feature vector. We propose a solution framework with two independent modules called recalling module and ranking module. When recommending movies to a particular visitor, recalling module first filters out a majority of less relevant movies from the candidate list. After that, ranking module will sort out movies with high rank from the remaining candidates for the visitor.

2 Related Work

2.1 Collaborative Filtering Recommendation

Collaborative Filtering Recommendation is one of the earliest and most successful technologies in the recommendation system. Collaborative filtering is based on the assumption that it is a good way for a user to find content that he is really interested in by first finding other users with similar interests to the user and then recommending the content they are interested in to the user [7]. It generally adopts the nearest neighbor method, calculating the distance between users by using the user's historical preference information. Then, it uses the weighted value of the product evaluations of the target user's nearest neighbors to predict the preference of the target user for a specific product [5]. Finally, it recommends products for the target user based on the predicted preferences.

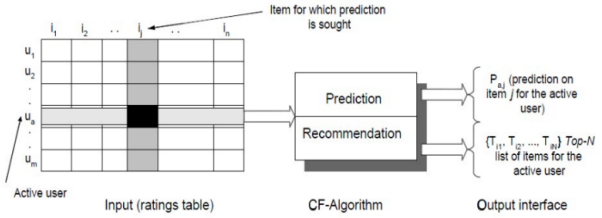

Figure 1: The collaborative filtering process [5]

The CF algorithm is divided into two types, one is user-based, and the other is model-based, also called item-based. The basic idea of user-based one is to use the nearest neighbor (neighbor-neighbor) algorithm to find a set of neighbors of a user, and then recommend the items that those neighbors like to the user. The users of this set have similar preferences to the user, and the algorithm predicts the user according to the preference of the neighbor. The basic idea of item-based[5] is to calculate the similarity between items based on historical preference data of all users in advance, and then recommend the items similar to those that the user likes to the user.

Whether it is based on users or based on items, the key is to establish an association matrix [7].

2.2 DNN in CTR

Deep neural networks (DNN) have shined in recent years in the fields of images, speech, and natural languages. Especially in image classification, speech recognition, and machine translation, DNN has surpassed humans, and its accuracy has reached the level of commercial application. However, the application of DNN in CTR's estimation of this scenario is still being explored. Data in the fields of images [9], languages, and natural languages are generally continuous, and there are certain structures between parts. For example, there is a close connection between parts of the image and its surroundings; there is a strong correlation between speech and text. However, the data estimated by the CTR, as described earlier, are very discrete. Many of the relationships between the features are the result of our ranking, and they are not interconnected. The biggest problem is how to construct a matrix like an image for a sample, which can have local connections and structures.

2.3 DeepFM

The focus of CTR is on learning combination characteristics. Combined features include second-order, third-order, and even higher-order ones. The higher the order, the more complex and difficult it is to learn. Before DeepFM [6], a variety of models including FM have been proposed and applied to solve the problem of constructing CTR and recommendation systems.

However, these models generally have two problems. The first problem is the bias between low-order and high-order combined features. We cannot extract both types of features at the same time. The second is the need for professional domain knowledge to do feature engineering. DeepFM successfully solved these two problems, and made some improvements.The core idea of DeepFM is:

(1) There is no need to pre-train the FM to get the hidden vector.

(2) FM Component + Deep Component. No artificial feature engineering is required. FM extracts low-order combined features, and Deep extracts high-order combined features.

(3) Ability to learn low-order and high-order combined features simultaneously.

(4) The FM module and the Deep module share the feature embedding section for faster training and more accurate training and learning.

DeepFM includes two parts, FM and DNN [6] [9], so the final output of the model is also composed of these two parts:

$$Y = sigmoid(Y_{FM} + Y_{DNN}) \tag{1}$$

The FM module enables the modeling of 1st-order and 2nd-order combined features. The role of DNN is to construct high-order combined features. The black lines in the network are fully connected layers. The parameters need to be learned by the neural network.

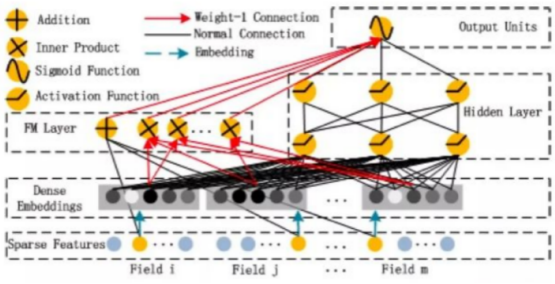

Figure 2: Wide&deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features [6]

3 Data Preparation

3.1 Dataset

We use the dataset ml-latest-small which is provided by the Movielens to do our project. The dataset describes 5-star rating and free-text tagging activity. It contains 100,836 ratings and 3,683 tag applications across 9,742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018.

Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. The data are contained in six files:

- **links.csv** Containing identifiers that can be used to link to other sources of movie data such as imdb and the moviedb.
  Data format: `movieId, imdbId, tmdbId`.
- **movies.csv** Containing the movie information.
  Data format: `movieId, title, genres`.
- **rating.csv** Containing rating data by users.
  Data format: `userId, movieId, rating, timestamp`.
- **tags.csv** Containing the tags information that users use to describe movies.
  Data format: `userId, movieId, tag, timestamp`.

3.2 Data Preprocessing

In the original dataset, the data of movies, rating and tags could be used in the training directly with regulated data format. We extract features from the user contributed content including tags, ratings, and textual reviews. In addition, each movie can be described as a feature vector. It's a kind of dense matrix like:

$$scores = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m,1} & s_{m,2} & \cdots & s_{m,n} \end{bmatrix}$$

where $n =$ feature number, $m =$ movies number

We used it to train a model to find the movies with close relevance. We also did PCA (Principal Component1Analysis) for the dataset so we can extract the key features further and clear away distractions.

4 System Architecture

We propose a solution framework with two independent modules, which are the recalling module and ranking module. When recommending movies to a particular visitor, recalling module first filters out a majority of less relevant movies from the candidate list. In addition, we also need to consider the user cold start problem.After that, ranking module will sort out movies with high rank from the remaining candidates for the visitor.

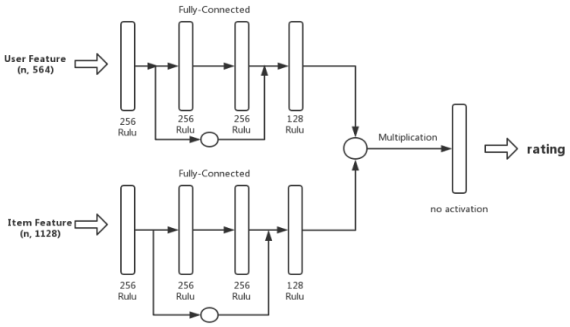

Figure 3: System Architecture

4.1 The recalling module

In recalling module, we apply a user-based collaborative filtering approach to obtain the input of ranking module. Specifically, we first find out users, who have similar preferences about movie to the visitor's, based on a rating matrix calculated from previous rating records. After that, we can obtain a list of movies, which those users having similar preferences have watched before, as the input of ranking module.

Therefore, The core algorithm of this part is a user-based collaborative filtering. Simply, user-based CF algorithm means when a user A needs personalized recommendation, he can first find a user group G that is similar to his interest, and then recommend to A items that G likes, which A has not heard of. We take the implementation process into two steps:

(1) Find a set of users with similar interests to the target user;

(2) Find movies liked by this set of users as well as not heard by the target user before and recommend those movies to the target user.

4.1.1 Discovery of similar users

The similarity between two users is usually calculated using Jaccard's formula or cosine similarity. Let $N(u)$ be the set of movies that user u likes, and $N(v)$ be the set of movies that user v likes, then the similarity between u and v is

Jaccard's formula :

$$W_{uv} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{2}$$

Cosine similarity :

$$W_{uv} = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u) \times N(v)|}} \tag{3}$$

The specific approach is that calculating the similarity between all users, we first needs to establish a "movie-user" correspondence table, and then for each movie, add 1 to the same movie between two users who both like it. Finally, the value of the cosine similarity or Jaccard's formula is used to calculate the similarity between the users, so that users who are more similar to the target user's interests can be found intuitively.

4.1.2 Movie candidates

First, we need to find the K users that are most similar to the target user u from the matrix, and use the set $S(u,K)$ to extract all the movies that the user likes in S and remove the movies that u has liked. For each candidate movie i, the degree to which user u is interested in it is calculated using the following formula:

$$p(u,i) = \sum_{v \in S(u,K) \cap N(i)} w_{uv} \times r_{vi} \tag{4}$$

Where r_{vi} represents the degree of user v's preference for i. In our project, because the user needs to give a rating, so we need to substitute the user rating.

Finally, the movies are sorted by score. Take the first few high-scoring movies to get the result of movie candidates.

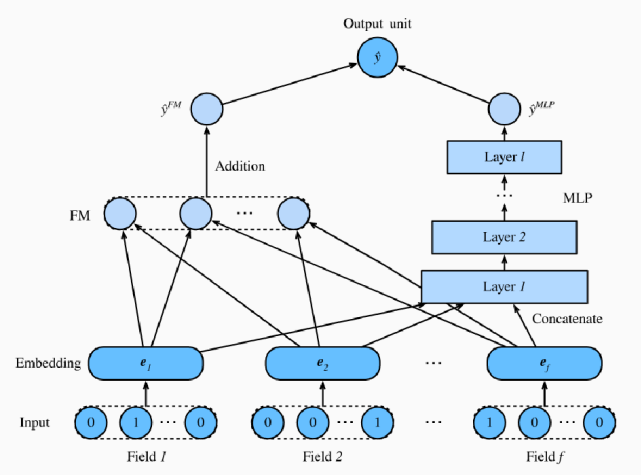

Figure 4: Ranking module

4.2 The ranking module

For our ranking module, we have implemented three different models. Firstly, we call naive ranking model. Briefly, we first calculate the average genome score over movies which the user has watched before. Then, we adopt the euclidean distance between the average score and the candidate movie's score as the criteria for sorting. We finally sort out movies having the largest weigthed sum of overall rating and the aforementioned distance.

Next, we apply a deep learning method in the second model. Specifically, we take the user's historical rating record and the genome score of a candidate movie as the input of our network. Then, those features are fed to a two-branch residual neural network to obtain a predicted rating. It is notable that user's features are difficult to extract, since we have only the historical rating record of the user. To address this problem, we first leave out one movie in the rating record for each time to augment our training data. After that, we reduce the dimension of these generated rating records with PCA to get enriched representations of user's features.

The third method we used is the DeepFM model. DeepFM was proposed in 2017. It combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. The architecture of DeepFM is illustrated below. The main reason why we use DeepFM is that it has a shared input to its "wide" and "deep" parts, with no need of feature engineering besides raw features. Comprehensive experiments are conducted to demonstrate the effectiveness and efficiency of DeepFM for recommendation system. Similar to what we do with the residual neural network model, we pass the user's historical rating record and the genome score of a candidate movie to a DeepFM model, and then get a predicted rating for a moive. With both the residual neural network model and the DeepFM

model, we recommend movies having highest predicted rating for the user.

5 Experiments and Results

5.1 Model Training Comparison

After training, all models can generate fairly reliable and effective recommendation results. For instance, if a user rated Spider-Man 1, one of the recommendation prediction will be Spider-Man series.

In terms of space complexity, since naive model suggests that there is no need for the extra parameter, its space consumption is significantly smaller. The number of parameters in the ranking model is the largest and the third model follows shortly. In terms of time complexity, because the candidates lists of the three models vary just a little, there are not much differences.

However, due to the lack of test users, we cannot verify our model rigorously. Thus, we adopt subjective testing approach within group members and the overall accuracy ranges between 80% to 90%.

5.2 Interactive Webpage

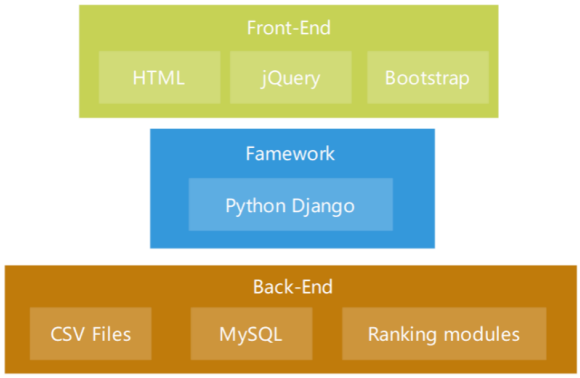

Figure 5: Project structure

We build an interative website to present our final works. Based on the framework of python Django, users can interact with the web interface without understanding the algorithms behind it. They first need to rate several movies they have ever seen and the system will recommend other movies to them.

Our project stack structure is shown as Figure 6 above. The front-end is built with HTML, jQuery and Bootstrap. The framework of python Django builds a bridge between front-end and back-end which is the core skeleton of the website. In the back-end, both CSV files and MySQL is used as data

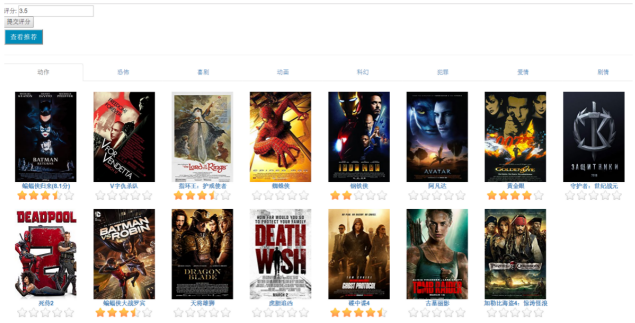

Figure 6: Interative webpage

storage method, and ranking modules are called by Django's components. The developed project structure is shown on the right, all website action functions are written in view.py and it'll call the functions in ranking modules to calculate which movies would recommend to the user.

When you login, choose a movie to score, and click on your username, you can see your scored movies just now. And after ranking several movies, you can get your personalized recommended results by clicking on the 'See Recommended Movies' button.

Figure 7: Ranked movies

Figure 8: Recommendation results

The process behind it is that when you rank movies, the

results would be stored in the Table `users_resulttable`. Then the data is sent to ranking modules and it returns the recommended movieId. After transforming movieId to imdbId by `links.csv`, the system reads the moviegenre Table and print all the movie posts on the screen.

5.3 Test Results & Analysis

Through testing the system by rating some movies and then show the recommended new movies, we find that if we rate for Spider-Man II, it will recommend Spider-Man I to me; if we rate for Mission: Impossible - Ghost Protocol, it will recommend Mission: Impossible - Rogue Nation; if we give several comedies high grades, it will recommend many other comedies to us. The system is relatively accurate overall.

In conclusion, in terms of accurate recommendations, User-based Collaborative Filtering gives us a relatively broad results whereas Item-based Collaborative Filtering predicts fewer outcome with higher accuracy. If we intersect these two sets of results, it's possible to get an empty set. From our testing result so far, we can conclude that the recommendation effectiveness of the three ranking methods(naive, ranking and DeepFM) are fairly consistent. However, in terms of training speed, naive ranking model outweighs the rest. DeepFM performs a bit slower and ranking is the slowest.

Therefore, if the user rates only a few movies, the running time of our recommendation system is approximately 2-3 seconds; Once they rate over 10 movies, the time will increase significantly, larger than 10 seconds or even worse; If rated movies is larger than 20, the waiting time will be longer than 25 seconds. The time complexity is nonlinear.

Acknowledgements

In this section, it's my honor to represent all my group members to show appreciation to our professor and TA's instruction. And to our each member in the team, we have been working day and night together and have been through so many difficulties for the past few months. Therefore, thanks for all of us, we could cheer for ourselves.

There are clearly individual contributions for each group member. **YANG Rongfeng** was the project manager, building frameworks and developing web interface system. **LAI Wendi** did Data collection & processing, model evaluation and PPT preparation. Our final system test was done by him. **LAU Kwan Yuen** designed the model construction and also helped to optimize the model. **WANG Xueying** kept an eye on our project process and also wrote report part one. **LI Yunli** collected and pre-processed data and also designed powerpoint and wrote report part two.

References

[1] Adomavicius G, Tuzhilin A, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions[J], IEEE Transactions on Knowledge and Data Engineering, 2005, 17(6):734-749.

[2] Pazzani M J, Billsus D, Content-Based Recommendation Systems[J], 2007.

[3] Pasquale Lops, Marco de Gemmis and Giovanni Semeraro, Chapter 3 in Recommender Systems Handbook, 2011.

[4] Pazzani M J, A Framework for Collaborative, Content-Based and Demographic Filtering[J], Artificial Intelligence Review, 1999, 13(5-6):393-408.

[5] Badrul Sarwar, George Karypis, Joseph Konstan, et al, Item-Based Collaborative Filtering Recommendation Algorithms[C], 10^{th} International World Wide Web Conference, 2001:285-295.

[6] Guo H, Tang R, Ye Y, et al, DeepFM: A Factorization-Machine based Neural Network for CTR Prediction[J], 2017.

[7] Guo G, Zhang J, Yorkesmith N, TrustSVD: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings[J], 2015.

[8] HRendle S, Scaling factorization machines to relational data[C], Proceedings of the 39^{th} international conference on Very Large Data Bases. VLDB Endowment, 2013.

[9] Wang R, Fu B, Fu G, et al, Deep & Cross Network for Ad Click Predictions[J], 2017.

[10] Steffen Rendle, Factorization Machines, ICDM, 2010.