University of New South Wales

School of Computer Science and Engineering

COMP9417

Machine Learning and Data Mining

Recommender System

Xiaowei Zhou z5108173

Jiexin Zhou z5199357

Ximing Fan z5092028

8 August 2019

Table of Contents

Introduction

With the rapid development of the Internet, more and more people get knowledge, enrich interests and discover entertainment through the Internet. Hence, implementation of high-quality content recommendation system becomes many websites' leveraging power of data, boosting the sales and views such as Netflix, Amazon, YouTube. The accuracy of systems impacts the website famous and the number of views.

In this project, we will implement a recommender system with Collaborative Filtering based on movie rating, using the data from Movie Lens to predict and recommend movies that specific user may prefer.

The task of this project is to develop and evaluate a collaborative filtering model based on user similarity of movie ratings to provide movie recommendation to users.

In this report, we will explain the theorem basis of the recommendation system, mathematical formula we use for developing the model, the process of experimentation on the model, and result and evaluation of the implemented system.

Related work

Recommender system

The essence of a recommendation system is to use adequately database to make recommendations to customers interesting's and distinguish users from items by entities and productions. Hence, recommendations are also made based on the relationship between the user and the items. In a Collaborative-Based model, ratings from multiple users are evaluated to predict unknown ratings. In a Content-Based model, predicted ratings are modeled from the rating and content from user's previous preference.

Content-Based Recommendation Model

In Content-Based recommender system, the description of items or movies is needed to describe the item's characteristic judged by system for providing recommendation. For the item, it will combine with rating and content as input to the model. For example, a movie, should have data of ratings from users, as well as set of actors, director, plot of story, genre, etc. By processing and modelling content information, recommender system can recommend related items or movies to users' preferences based on content directly.

However, too much content may make the recommendation inaccurate, such as large number of actors, mixed genre, high dispersion of rating of story. This will lead to the inaccurate recommendation. Some more complexed algorithms are required to avoid these issues.

Collaborative-Based Recommendation Model

In general, the collaborative filtering system is, assuming that for a group of users having similar preferences, one's preferred item may also be preferred by others. It is aimed to predict the new interactions based on previous ones. There are two kinds of methods to achieve this function.

Memory-base

In memory-based method, there are two type of algorithms to achieve: based on user or item. In this project, we prepare use the User-based collaborative filtering to implement the recommender system.

User-based collaborative filtering

In this method, the user identifies clusters and utilizes the other users who interact with them and have common interests. In addition, according to the similar users, compute the weight of similarity and recommend similar items.

Item-based collaborative filtering

This is the method to predict items rating on other things from the user set, it is similar with Use-based collaborative filtering. The difference is that it will compute the similarity of all movies or items, considering preference as a whole group of items.

Model-base

In model-based methods, machine learning and data mining methods are used in the context of predictive models. We could use the user-item interact to build a model to make predictions of top k movies or items.

Dataset

For this assignment, ml-1m dataset from MovieLens is used as the source of our prediction model. This dataset contains 1,000,209 anonymous ratings for 3,883 movies(movie id ranges from 1 to 3952) from 6,040 MovieLens users, with at least 20 ratings from each user. The dataset is available at: *http://files.grouplens.org/datasets/movielens/ml-1m.zip.*

Implementation

The movie recommendation software includes six parts of implementation which provides all information for model training, performance evaluation, performance visualization and model application.

Part 1. Load and Reformat Data

In this part, ratings.dat file from data-1m dataset is read in through read_csv function from pandas' library. The data file will be format into data frame with four columns: "userID", "movieID", "rating", "timestamp". A constant random seed is set to 9417 for result replay.

Part 2. Rating Matrix

In this part, a rating matrix is established as: userID for each row, movieID for each column, ratingMatrix[userID, movieID] for the rating of a user to a specific movie with default value 0.

Part 3. Train/Test Splitting

In this part, 10% of existed ratings for each user will be separated from original rating matrix to become a test set of data. The remaining 90% of rating information will be training set. The dataset guaranteed that each user has at least 20 ratings, which also guaranteed that we have at least 18 training data and 2 testing data for each user. We use the numpy function random.choice with no replacement to select ratings for test data set and update their value in training data set to be 0.

Part 4. Similarity Matrix

In this part, similarity matrix based on training data set is established using Cosine Similarity Function as following formula:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

For our training rating matrix R, we construct the similarity matrix as:

$$\frac{R \cdot R^T}{\|R\| \times \|R^T\|}$$

using numpy.dot and numpy.diagonal function.

There is additional work to ensure the similarity matrix working properly. To avoid divided by zero error, and let all movies participate into prediction, we add 1*10^(-6) to each item of R.dot(R.T).

Part 5. Model Training and Evaluation

In this part, the model is trained with consideration of top k users rating and evaluated by its Mean Squared Error from testing matrix.

The k-value we select to train and evaluate include: 5, 10, 20, 50, 100, all user. We predict an unknown rating of a user c to movie m by aggregating the ratings of top K users c' that are most similar to user c who have rated movie m.

We use the weighted sum for the rating P[userID, movieID] - prediction of rating of user userID on movie movieID, as the following formula:

$$P[userID, movieID] = \frac{S[userID, :][top_k] \cdot T[:, movieID][top_k]}{\sum S[userID, :][top_k]}$$

S for similarity matrix, T for test rating matrix, top_k for index of top_k similar users.

The top_k similar users are sorted by quicksort of similarity matrix except the user himself.

For k = number of users, the formula can be simplified as:

$$P = \frac{S \cdot T}{\sum S^T}$$

After we got the prediction matrix from training set, we can calculate Mean Squared Error of predicted ratings for items in test data set. We select a proper k value for next part recommendation.

Part 5.5 Visualization

We use matplotlib.pyplot to plot the Mean Squared Error of different k values. This helps with selecting proper k-value for recommendation.

Part 6. Recommendation and Output

In this part, we use the whole data set as prediction source. A new similarity matrix based on all ratings is established with same algorithm from Part 4, and we calculate the predicted ratings of a specific user ID entered from stdin and give the top 10 predicted rating movies for the user, excluded movies he/she already rated.

The algorithm is a simplified version for one user ID from Part 5 top_k prediction.

The movie IDs in the dataset are not continuous, which require us to do additional process to transfer matrix index into real movie ID and movie Information. This functionality is done in movieID_matrix_correction function.

The output will be array of top 10 predicted high rating movie of the user.

Experimentation

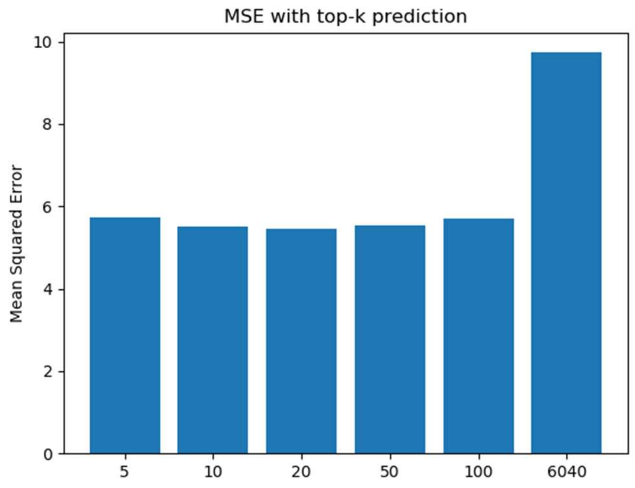
In this assignment, the key point of coming out proper recommendation is to select a high-performance k value for top-k similarity.

All data showed in this section, is generated with input "input.txt" to the program, and data is exported to "output.txt". These files are attached with the assignment submission. The instruction of using the sample files and the program is provided.

The performance measure is the Mean Squared Error of prediction based on top-k similar users from training set and the real marking in test set, lower is better.

We first use all users to determine predicted rating for unrated movies for each user, in which k = 6040. It is expected to have poor performance since users obviously would not be like all others.

We then use top 5, 10, 20, 50, 100 for training, and plot their Mean Squared Error to the testing set. These settings are using same training test and test set.



K-value	5	10	20	50	100	6040
MSE	5.743	5.514	5.461	5.541	5.707	9.733

According to the results above, we choose k = 20 for best performance recommendation.

After k-value is decided, we use this k value and full rating matrix for recommendation. The software accepts recommendation request for a specific user ID and give his/her predicted top 10 rating movies that he/she did not rated.

In sample input/output we tried User ID = 22, showing the following recommendations:

```
[[[259 'Kiss of Death (1995)' 'Crime|Drama|Thriller']]
  [[355 'Flintstones, The (1994)' "Children's|Comedy"]]
  [[592 'Batman (1989)' 'Action|Adventure|Crime|Drama']]
  [[526 'Savage Nights (Nuits fauves, Les) (1992)' 'Drama']]
  [[1386 'Terror in a Texas Town (1958)' 'Western']]
  [[3470 'Dersu Uzala (1974)' 'Adventure|Drama']]
  [[540 'Sliver (1993)' 'Thriller']]
  [[2027 'Mafia! (1998)' 'Comedy|Crime']]
  [[1616 'Peacemaker, The (1997)' 'Action|Thriller|War']]
  [[2114 'Outsiders, The (1983)' 'Drama']]]
```

The ratings of user 22 from ratings.dat shows high preference on comedy, drama and thriller, and some preference on Action and Adventure. Our prediction worked pretty well, provided movies in these categories which has high probability that user 22 may prefer. Also, we correctly implemented the filtering that all recommended movies have not be rated by the user.

Conclusion

This study was completed to provide User-based collaborative filtering recommendation of movies. Users with their ratings to different movies are used to compare with others to find similar users and predict the potential rating of movies they did not rate. A recommendation of movies is provided based on predicted rating of specific user. The recommendation result is also consistent with potential content-based prediction, showing similar trend of user preferred categories of movies.

However, this collaborative filtering method has a lot of issue, including "cold start" situation, which is the data source for constructing similarity matrix is not large enough, and "new user" situation, which is a new user may not get high performance recommendations due to their small amount of ratings made. The dataset we use is relatively large and each user have a considerable enough amount of rating records. In a real productive environment, "cold start" and "new user" situation will have more impact and will affect recommendation a lot during the early stage - rating information not enough.

Reference

Aggarwal C.C. (2016) An Introduction to Recommender Systems. In: Recommender Systems. Springer, Cham. DOI= https://doi.org/10.1007/978-3-319-29659-3_1

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI= http://dx.doi.org/10.1145/2827872