Movie Recommendation System Based On IMDB

Oliver Zhou
Computer Science and Engineering
the Ohio State University
OH, the United States
oliver.2866@osu.edu

Honghui Bao
Computer Science and Engineering
the Ohio State University
OH, the United States
bao.276@osu.edu

Jason Yao
Computer Science and Engineering
the Ohio State University
OH, the United States
yao.712@osu.edu

***Abstract*—Our goal is to recommend movie based on IMDB dataset without user information. We use K-means and GMM to cluster our data points and find high quality movie inside the cluster. The experiment shows that the revised methods have higher qualities than baseline method and can usually recommend properly.**

Keywords—recommendation system, content-based filtering, clustering, distance metric, gaussian mixture, k-means

I. Introduction

Recommendation systems are one of the most widespread application of machine learning algorithms in business. There are two major approaches to design a recommendation system: Collaborative filtering and Content-based filtering. Collaborative filtering is to find similarity of users. This method collects users' preference and activity like rating, review and view history, and calculates the similarity to other users to recommend products. Content-based filtering are to find the similarity of item attributes under different distance metrics and are not able to recommend new items dissimilar to item view in the past. And only look into movies' similarity may lead to bad quality movie.

This paper provides a revised approach of content-based for movie recommendation based on IMDB dataset. The key idea is to cluster movies beforehand and find popular and high rating movie inside the cluster. This way, the computation cost is greatly reduced, and we don't have to define a specific rule of similarity for votes and ratings while also make full use of these two features. Because of the natural property of clustering, we don't have to assign a higher weight prior on selected attributes and the update of attribute weight can be achieved automatically.

II. Task

A. Problem

The basic task is to recommend a number of new movies based on user's highly-rated movies. The necessity of doing this is because similarity of two movies are not defined by a single feature. Rating, genre and many other factors defines whether the two movies are "similar" or not. Predictions out of limited factors are often biased, say, a 19th century action movie may reveal if you simply look at genre but have no interest in vintage movies. So, combining and assigning weights to related features are indeed important.

B. Main Idea

The problem is mainly about clustering but also classification after. First we try to find similar movies and we use clustering to achieve this goal. After clustering, we output certain movies based on test sets. It's hard to judge a movie's quality simply based on its attribute and this is also not the mainly focus of our project. So what we assume here is that if a movie on IMDB is under a certain threshold of ratings and numvotes, the possible user will probably don't like this movie. So ratings and numvotes work as performance measure to tell whether the output movies are good recommendations or not. If one of its genres matches one of the genres of the test movie, the rating is not too low(under 6) and it's not that unpopular(numVote less than 100), then we call it a fair recommendation. If it's not, then we update the weight of the unqualified feature by adding according to a learning rate.

III. Data processing

A. Data Set

The dataset we obtained from IMDb Database, an international movie database that provide specific and comprehensive information about movies. The dataset had 7 subsets (basics, crew, episode, principals, rating, etc.) among which contained 9,229,989 items and 919658 of them have ratings. We considered features we assumed important for the comparison, which consist of rating, region, type, isAdult, startYear, numVote, director, actor and genre.

B. Preprocessing

This part is actually much harder than we thought, since IMDB spitted those feature into seven subsets, we have to combine them into one matrix. The very first thing we did was discarding those with type not "movie" and as a matter of fact, eighty percent of the whole set belongs to tv, game, documentary and others. Then we discard movies that have low ratings (under 6) or very few numbers of votes (under 100). These operations rapidly reduced the size of the dataset to thousand-level. Then we implemented one-hot coding for genre and region. The final feature matrix is now established.

IV. Algorithm

A. K-Means

We used k-means to cluster similar movies and made recommendation for a test new movie by finding its cluster centroid point. We initialize three different weight vectors which initialized fairly, emphasized rating and emphasized numVotes. We calculated variances of different number of clusters and found out that sixty-nine clusters fit the dataset best. Then we trained our dataset to adjust the corresponding weight vector. Like mentioned before, we increase weights of the unqualified features by different learning rates. For test data, we recommend three movies out of the three different initializations and learning rates. In these case, the movies we recommend must be either a high rating movie or a popular movie, and also related closely to the test data movie. The emphasis on high rating movies and popular movies can help us improve user's satisfaction about our recommended movies.

B. *Gaussian Mixture Model*

We also used Gaussian Mixture algorithm. In this part we follow the pattern of the k-means to train our weight vectors. The reason we chose Gaussian Mixture is because most of the features like ratings, numVote and genres are basically gaussian models in common sense. Due to different weight vectors, influences of different features vary.

C. *Baseline:*

Nearest Neighbor method is our baseline method. In this method we calculated the similarity between movies directly on the processed movie feature matrix. We use ball tree data structure for fast computation since our dimension is relatively high and our number of points are up to million. And the generated k nearest neighbor is our recommend movie sequence for a certain movie. The k is subject to change when compared with the other algorithms. Here we use euclidean metric which is the same as K-means and GMM for computation.

V. RESULT

1. Total performance on 100 test data:

Average Match and Qualification Percentage:
(Qualifying rating>=6.0, voteNum>=100, genres match at least one, region matches)
K-means: 58.33%
GMM: 58.33%
NN: 25%

2. Example:

Test data

Title	***Ratin g***	***VoteNum***	***Year***	***Genres***	***Region***
Golden Dawn Girls	6.4	170	2017	Docum entary	NO

K-Means Recommendation

Title	***Ratin g***	***VoteNum***	***Year***	***Genres***	***Region***
Listener	6.1	175	2015	Drama	JP
Xiaozhen's Story	8.8	175	2016	Drama	KR
Santiago	7.3	194	2018	Documen tary	FR

GMM Recommendation

Title	***Ratin g***	***VoteNum***	***Year***	***Genres***	***Region***
The Men's Room	9.2	53	2018	Documen tary	NO
Fractured	5.7	488	2016	Thriller	GB
The Wandering Earth	7.3	7073	2019	Sci-Fi	CN

NN Recommendation:

Title	***Ratin g***	***VoteNum***	***Year***	***Genres***	***Region***
Amor de Perdição	6.7	122	1979	Drama	PT
Yeva	8.0	167	2017	Drama	AM

We can see from the result that Nearest-Neighbor is the least accurate algorithm because movies may not cluster so obviously and clusters interfere a lot. So Nearest-Neighbor behaves poorly on borders. K-Means is much better than Nearest-Neighbor and has a stable performance across testsets. GMM's behavior is a little bit tricky. For some test data, GMM may result in perfect recommendations which qualifies all four evaluations, but behaves poorly for all three recommendations on others.

REFERENCES

[1] "https://www.imdb.com/interfaces/". Information courtesy of IMDb (http://www.imdb.com).

[2] Carlos, P. (2017). Recommender Systems — User-Based and Item-Based Collaborative FilteringI. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Christina, C. (2007). A hybrid movie recommender system based on neural networksR. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[4] Badrul, S. (2016). Item-Based Collaborative Filtering Recommendation AlgorithmsM. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.