DSCI 553
10/10/20
Jie Ji, Tao Xiong, Zhongze Chen

Final Report: Clustering CF recommendation system

Description

Collaborative filtering algorithms are widely used in recommendation systems, especially for movie platforms. In essence, CF algorithms assume that users with similar past behaviors will have similar preference on items, so it will look for users' past behaviors and decisions made by similar users to recommend items to active users. However, the normal CF algorithm has some weaknesses such as cold start, sparsity, first rater and popular bias. To address some of these issues, we came up with an idea to merge clustering methods with the CF algorithm. Instead of measuring similarities between every pair of users or items, our new clustering CF algorithm will cluster both items and users and then store the similarities between users and user clusters and cluster labels for each item. Finally, the algorithm can predict a user's preference or rating by calculating the weighted average ratings in different clusters.

The clustering CF algorithm has two major advantages. The first one is that it can solve the sparsity problem. For example, in movie platforms, there are millions of users and hundreds of thousands of movies. However, users usually would only watch and rate a small part of the whole movies database, so the user-rating pairs are in a very sparse matrix, making it very difficult to detect similarity between two users. With the clustering CF algorithm, it will first cluster similar items or movies into a group, reducing the hundreds of thousands of items into tens of clusters and making it very easy for two users to watch and rate movies in the same cluster. The second major advantage of the clustering CF algorithm is that it can greatly improve the memory efficiency. For the normal CF algorithm, it will measure and store the similarities between every pair of users ($O(N^2)$). For the new clustering CF algorithm, it will only measure and store similarities between users and clusters ($O(NxK)$, where $K << N$), thereby largely reducing the memory usage.

Innovation

The normal CF algorithm can work well when the size of users and items is small. However, scalability becomes a key problem of the algorithm as it uses a great amount of memory resources to increase the accuracy of prediction. In fact, it is actually a trade-off between accuracy and memory usage. Also, the extra accuracy by more memories is limited. Therefore, the major innovation of our new clustering CF algorithm is to sacrifice a small amount of accuracy but save a great amount of memory. Another very important innovation is that we cluster both users and items. As mentioned before, clustering items can efficiently solve the issue of sparsity because each user's basket is represented by item clusters rather than individual items. Clustering users can help to further reduce the memory usage because we only need to calculate the distance between users and the centroid of each cluster. Therefore, our new algorithm is useful in scenarios of large databases.

Data and Data Cleaning

The dataset is from Kaggle and is mainly about user ratings on MUBI platform, which could be described as Netflix for art movies. There are two major datasets including movie data and rating data. The movie data includes more than 220 thousand movies and 10 fields such as movie ID, title, releasing year and directors. The rating data has around 15 million rating records from 451 thousand unique users. As the dataset is well structured and very clean, so we didn't spend much time on data cleaning, we only removed the records with unreasonable value like the releasing year before 1700. The only major problem with the movie data is that it has no feature of genre, but this variable is very important for movie clustering. At first, we used the url feature to scrape genres for each movie, but web scraping didn't work since IP got blocked after scraping several tens of movies. To solve the problem, we used another dataset from IMDb and attempted to merge two datasets. However, we found only 60% of movies have perfect match because it's very often the movie title is little different in two datasets. Then we used fuzzy match on titles for movies released in the same year. Finally, 95% of movies have a match.

Hypothesis

Our project is established on the following hypotheses. First, we assume that similar movies will be clustered into the same movie clusters. Second, we assume that users with same tastes will be

clustered into the same user clusters and their ratings for the same movie will be similar. Lastly, we assume that a user cluster can be represented by the centroid of the cluster. In other words, our hypothesis is that we can predict a user's rating on a movie by similar users' ratings on similar movies.

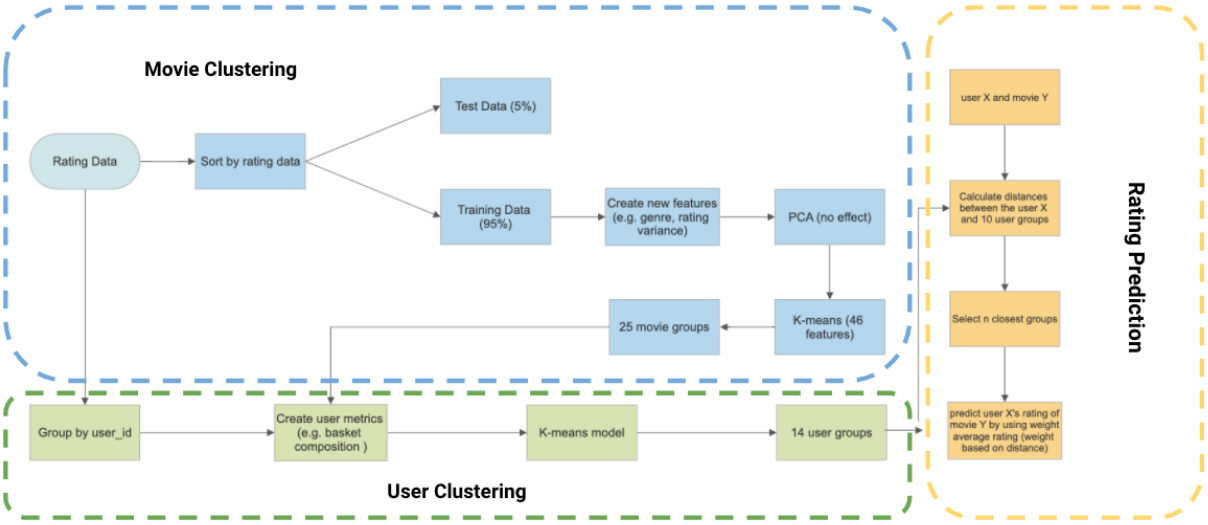
Literature Review

There are some applications of clustering in recommendation systems (Urszula Kużelewska, 2014). The usually used clustering models are k-means and hierarchical clustering. However, the result of traditional k-means is not ideal as the centroid is calculated by averaging the values in each cluster. With a modified algorithm for centroid like most frequent representative values or a different clustering method like SCuBA, the performance of the recommendation system can improve a lot.

Method

After preparing the data, we first sorted the rating data by the date and used the first 95% records as the training data and the rest 5% as test data. Then, we began to cluster the movie based on 46 variables like movie genres, movie release year, popularity and average ratings. Also, we tried the PCA method to reduce the dimensions, but the data variance is explained evenly by all the components, which means the effect of PCA is negligible, so we still used all 46 features as the input of modeling. Next, we used the K-means to cluster the movie, which is an unsupervised model and can divide data into k groups. We tried different k from 1 to 50 and plotted the distortions under different k. Although the plot has no obvious elbow, we thought the group number of 25 is reasonable, so finally we divided all movies into 25 groups.

After clustering the movie, we did a similar process to cluster the users. We first grouped the rating records by user id and created new variables for each user such as the total number of watched movies and the average rating. Also, we created some new variables based on movie clusters. For example, we think users will have different preferences on movie genres, so we calculated the proportion of different genres to represent a user's overall basket. Then we input all features into the model and used the K-means model to divide users into 14 groups.



Model Flow Chart

Evaluation and Results

In order to evaluate the performance of our clustering CF model, we also implemented a model based CF algorithm on our movie rating dataset as our baseline predictor. However, it is impossible for us to implement model based CF on the entire dataset. Model based CF requires to store a utility matrix which is M x N large (M is the number of movies and N is the number of users) before we factorize it into two smaller matrix components. We cannot implement it because our PC memory is not large enough, but we are able to implement model based CF on a 2% size of the entire dataset.

$$r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$$

Baseline Model

We sampled 2% users of the total 450,000 users, which is around 9,000 users. Then we only kept the rating records which are related to these sampled users. We would implement both CF models on this sample size dataset to see whether our clustering CF model would outperform the baseline predictor. Then we would implement only our clustering CF model on the entire dataset (memory not large enough for model based CF) to see whether our algorithm still works well on

a very large dataset. For both sample size dataset and the entire dataset, we sorted the data by rating date and made the first 95% data as training data and the left 5% as test data.

For our clustering CF algorithm, there are two major matrices, a User - User Cluster distance matrix D and a User Cluster - Movie Cluster average rating matrix S. The way our clustering CF algorithm predicts user X's rating on a movie is: first, the model finds which movie cluster does a movie belong to, say movie cluster t; then, it will find the average rating of each user cluster on all of the movies in movie cluster t, and calculates a weighted average based on user X similarity to each user cluster (here we calculates the Euclidean distance between X and each user cluster, and use 1/distance as the similarity; and at last, the algorithm will add the user bias to the formula.

$$r_{xi} = \sum 1/D_{xk} \times S_{kt} \Big/ \sum 1/D_{xk} \; + b_x$$

Clustering CF Model

Here is an example. We want to predict user X's rating on the movie Harry Potter. The movie belongs to movie cluster T and we have 3 user clusters. The distances between user X to three user clusters are 1.5, 2 and 0.5, and the average ratings of the 3 clusters on movie cluster T are 3, 5 and 4. And the user bias is 0.2. The prediction is 4.15 and is calculated as follows.

We use MSE and false positive rate to measure model performances. On the 2% size sample data, the baseline predictor has a MSE of 0.868, which beats our clustering CF model by around 0.01 in MSE. However, in this movie rating scenario, MSE is not as important as false positive rate. We need to make sure that users would like the movie that we recommend to them, but we wouldn't worry too much if a user do like a movie and we didn't make the recommendation. If a user rates a movie for 4 or 5 stars, then we define that the user likes this movie. And for both models, if the prediction score is larger than or equal to 4, then we will recommend this movie to the user.

The false positive rates for both models are almost the same, only 0.2% in difference. However, baseline predictor only gives around 4,600 recommendations while clustering CF algorithm gives 6,100 recommendations without decreasing accuracy. Therefore, we will prefer clustering CF model for such scenario.

2% size data	Model Based CF	Clustering CF
MSE	0.868	0.879
False Positive	0.157	0.159
# Recommendations	4636	6102

The clustering CF model works well on a relatively small size data, so we try it on larger data, such as 5%, 10% and 20% size of the entire dataset, and the algorithm still gives us similar results. Finally, we tried it on the entire dataset and get a result of 0.872 MSE and 16.2% false positive rate. Therefore, our results support our hypothesis.

Conclusion

This project gives us the opportunity and experience to build a recommendation system on very large dataset. Instead of building a recommendation system that is clearly elaborated in class, we combined clustering algorithm and user based CF and build our own model. It's really hard in the beginning and we've run into a lot of unexpected problems during the project, but we tried so hard and finally we succeeded in building the model and the results are good.

However, there are still a lot of improvements that we can make in the future. 1) Time bias is not included in our model. We could include time bias to make more precise prediction. 2) Add more features for clustering, both movie clustering and user clustering. For movies, we can utilize features like movie language, movie country, director and actor. These features are categorical but has too many classes to convert into dummy variables, so we could apply target encoding on these features and make them numeric. For users, we can add features like user gender, user age, user membership or not and so on. By adding right features, it is more likely for us to cluster users into the right group. 3) Try different number of clusters. For this situation, we chose

moderate number of clusters based on the elbow method graph. However, since there is no obvious elbow in the graph and one of our goals is to recommend movies to some users of unique tastes, increasing the number of clusters, for both users and movies, might give us more accurate recommendations.

Reference

1. Alguliyev, Rasim M, et al. “Efficient Algorithm for Big Data Clustering on Single Machine.” IEEE, IET, 12 Mar. 2020, ieeexplore.ieee.org/document/9032014.
2. Sharma, Ritu, et al. Collaborative Filtering-Based Recommender System: Approaches and Research Challenges. 13 July 2017, ieeexplore.ieee.org/document/7977363.
3. Wu, J., Chen, L., Feng, Y., Zheng, Z., & Zhou, M. C. (2012, September 12). Predicting Quality of Service for Selection by Neighborhood-Based Collaborative Filtering. Retrieved from https://ieeexplore.ieee.org/abstract/document/6301755
4. Liu, Haifeng, et al. “Context-Based Collaborative Filtering for Citation Recommendation.” IEEE, IEEE, 28 Sept. 2015, ieeexplore.ieee.org/abstract/document/7279056.
5. Honda K., Sugiura N., Ichihashi H., Araki S. (2001) Collaborative Filtering Using Principal Component Analysis and Fuzzy Clustering. In: Zhong N., Yao Y., Liu J., Ohsuga S. (eds) Web Intelligence: Research and Development. WI 2001. Lecture Notes in Computer Science, vol 2198. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45490-X_50
6. Chen, Tung-Shou, et al. A Combined K-Means and Hierarchical Clustering Method for Improving the Clustering Efficiency of Microarray. IEEE, 21 Feb. 2006, ieeexplore.ieee.org/abstract/document/1595432.