Recommender system using collaborative filtering

COMP9417 Machine Learning Project

**Abstract**

Collaborative filtering algorithm are widely used in the recommendation system of movie website, which is based on the analysis of a large number of user's historical behavior data. Traditional collaborative filtering algorithm has it defects such as data sparsity, cold start and scalability. This paper will focus on how to recommend movie to user accurately. We use singular value decomposition(SVD) on matrix of users or hobby, which will provide some hidden demands of user. The experimental results show that the advantage of Collaborative filtering algorithm with SVD.

**Key words**

singular value decomposition, collaborative filtering algorithm, similarity

**Introduction**

With the development of Internet, infotech is part of the techno logical revolution and that is with us now. The data of human grows exponentially. It brings convenience to users, but it also led directly to the “information overload”. It means the information is larger than the demand of user and some redundant data will disturb the use of useful information. On the one hand, people get more and more information, on the other hand, people have to spend more time and energy to search for information which is helpful to them, which makes the phenomenon of information overload is becoming more and more serious. “Information overload” can be solved by traditional information retrieval technique to some degree. However, it cannot provide user with information which is based on their hobby. It will be the same result for different users. How to support the information which the user is interested in become a project that need to be settled.

Recommender system is based on the analysis of a large number of user's historical behavior data. It set a metric on the hobby of user(comments, share, favorites) and get the relationship between any two users. Then it will recommend the movie to those who has similarity hobby. If the hobby of user has changed, the result of recommender system would change. It will have different weights on different features.

**Related work**

**cosine similarity:**

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures cosine of the angle between them.

**MSE:**

We use MSE to predict the error in order to test the accuracy of the different algorithm.

**Base Line predictor:**

is trying to fit the model to be less error, is to avoid over fitting. is the set of users, is the set of items, is the one which we want to predict. is the real value of test data.

So , .

**Singular-value decomposition(SVD)**:

where

is a diagonal matrix which are weights correspond to each feature.

is a matrix with each row represent to each user.

is a matrix with each column represent to each movie.

Thus, we could just omit the low weight part by and just keep p% of the weight, which p% is the parameter we would like to find.

Furthermore, needs to prune to j columns, by doing this, the matrix of will be thinner, and less data need to compare. This reduce the time of computation.

**Collaborative filtering(CF):**

This approach uses user rating data to compute the similarity between users or items. This is used for making recommendations. This was an early approach used in many commercial systems. It's effective and easy to implement. Typical examples of this approach are neighbourhood-based CF and item-based/user-based top-N recommendations. For example, in user based approaches, the value of ratings user 'u' gives to item 'i' is calculated as an aggregation of some similar users' rating of the item:

where 'U' denotes the set of top 'N' users that are most similar to user 'u' who rated item 'i'. Some examples of the aggregation function includes:

where k is a normalizing factor defined as ,and

where is the average rating of user u for all the items rated by u.

The neighborhood-based algorithm calculates the similarity between two users or items, and produces a prediction for the user by taking the weighted average of all the ratings. Similarity computation between items or users is an important part of this approach. Multiple measures, such as Pearson correlation and vector cosine based similarity are used for this.

The Pearson correlation similarity of two users x, y is defined as

where is the set of items rated by both user x and user y.

The cosine-based approach defines the cosine-similarity between two users x and y as:[4]

The user based top-N recommendation algorithm uses a similarity-based vector model to identify the k most similar users to an active user. After the k most similar users are found, their corresponding user-item matrices are aggregated to identify the set of items to be recommended. A popular method to find the similar users is the Locality-sensitive hashing, which implements the nearest neighbor mechanism in linear time.

**cold start:**

when a new user coming, we do not have much information about him or her, thus, in this situation we need to generate a start data for these users. We could first provide the global popular movie for these users and after period of time. we get some information of these users, and we update this users weight.

**Baseline Predictors:**

CF models try to find the relationship between users and items. And it predicts rating values according this kind of relationship. However, much of rating values have been observed that beside the relationship between users and items, it also been effected by either users or items, independently. Therefore, most of CF data would result in large user and item biases, which means some users are more likely to give higher ratings while some would like to give lower ratings. It is also true for items. For example, in the movie rating system, some movie has high quality, so it is obviously that it would have higher ratings than those which are terrible film. In this case, these high-quality movies would have high bias.

To observe this kind of effects, which is independently of the interaction between users and items, we will use baseline predictors (also known as biases). Much of bias would be captured by these predictors. Thus, it would be vital to build a correct and suitable model so that we could get higher accuracy in result. And this kind of model would isolate the interaction between users and items. By this way, it would result in a model which are more appropriate to represent bias of users and items.

Thus, a baseline prediction for an unknown r\_ui which is denoted by (b\_ui ) and associate with the bias with users and items would be:

The parameters and indicate the observed deviations of user u and item i, respectively, from the average. And b indicates the average rating over all movies.

Implementation

In this project, we use several different algorithms which are related to collaborative filtering.

Base line predictor:

1. Choosing matrix:

First step, we try to use dense matrix in this project, actually, it does work in the 100kb small dataset. But after transfer to a 500mb large movie dataset, the memory leaks and computer stop running. This makes us to choose a sparse matrix, which will store all dataset efficiently.

1. Reduce dimension:

In the dataset ‘rating.csv’, there are four dimensions ‘userid, movieid, rate, time’. In the beginning, we could choose a model depending on time. , this formula could make item depending on time, with time elapsed the influence of the record could be gradually reduced, if σ is higher, the influence will be more longer and vice versa. However, according to Occum’s razor theory. we would like to make the algorithm simpler. So, we reduce the dimension of ‘time’ by just applying aggregation on ‘userid, movieid’.

1. Model choosing:

The first model: Base line predictor, which is a Netflix 100 million-dollar algorithm. The formula is , is the bias of user, is the bias of movies, b is the average bias of the matrix. is what we would like to predict. Which means if we like to predict a value , it depends on the global average value and current user average value and current movie average value. Moreover, . Similarly, we can get .

Reference: chapter 5 Advances in Collaborative Filtering

1. The second model: Collaborative filtering and svd. The reason we would like to add svd is that, the matrix is to sparse, this will generate many redundant, so we decompose the original matrix in the form of . Next, we select the p% of the and try to do the similarity on . This do not help much to reduce much of time(we will explain it in the result). Finally, we use the collaborative filtering algorithm by applying on data which are getting from
2. Error testing:

We choose 0.1% of data for testing, which is not part of training data to cross validation. Furthermore, we test it on different algorithm and try to find the best one.

Results

**Time complexity:**

We compare these algorithms by time consumption and MSE error, So, below are some graph related to the result by using 100k dataset of movie(ratings.csv).

Above four graphs compare the time consumption of Baseline predictor and Collaborative filtering which denoted as ”CF”. We could know that Base line has an extremely fast speed which has a time complexity of O(1), while CF has a time complexity of O(n3). Moreover, we reduce the original matrix with svd by taking the weight of . Although dimension has been reduced, the time consumption doed not reduced much.

Above graph compare collaborative filtering and advanced collaborative filtering (baseline predictor) with different sample size. The blue one is base line predictor and the orange one is traditional collaborative filtering.

It is obvious that user-base collaborative filtering takes longer to predict, while Base line predictor does not need much time to predict.

**Note: In the code which we provide, by using 500mb movie dataset(ratings.csv). it takes few second to get the baseline predictor, however, it will take hours to get traditional collaborative filtering.**

Thus, we suggest of using the advanced collaborative filtering (base line predictor) in the real case.

**MSE:**

Next, we test the error by these two algorithms.

Below are four graphs of the error between and base line predictor and collaborative filtering by using 100k dataset of movie(ratings.csv).

We can find out that base line predictor always has a lower MSE.

Conclusion

There are also several disadvantages with traditional collaborative filtering. Its performance decreases when data gets sparse, which occurs frequently with web-related items. This hinders the scalability of this approach and creates problems with large datasets. Although it can efficiently handle new users because it relies on a data structure, adding new items becomes more complicated since that representation usually relies on a specific vector space. Adding new items requires inclusion of the new item and the re-insertion of all the elements in the structure.

There are two major problems with user-based algorithms:

1. Data sparsity. A large-scale e-commerce recommendation system generally has plenty of items, and it could be the case that less than 1% of the items the user would like to buy. The overlap between items purchased by different users is very low, resulting in the algorithm being unable to find a user's neighbors, that is, user who have similar preferences.

2. Algorithm scalability. The computation of the most similar users algorithm increases with the increase in the number of users and items. Thus, it is not suitable for use in the situation that there is a large amount of data need to process.

The advantages with base line filtering include: the exploitability of the results, which is an important aspect of recommendation systems; easy creation and use; easy facilitation of new data; content-independence of the items being recommended; good scalability with co-rated items. It more than 1000 times faster than traditional collaborative filtering, this could make a better user experience. Moreover, it has a lower MSE, which means that it is more accurate.

The disadvantage of base line filtering is that, when a new user coming. We need to compute the most popular item for of the original matrix. This is O(n2) which is related to b, bi and bu, the time consumption will be much larger but it is still better than traditional collaborative filtering.

Future Work

This project is about recommending system. We use two methods, they are baseline predictors and user based collaborative filtering. The square error of CF we get now is about 1.2. And we hope that we could do something to reduce it to less than about 0.6 in the future.

For baseline predictors, we only use the ratings given by users for different items. It is a naïve approach but it has good result in predicting the ratings. And the bias, or say the users preference would be changed by time. It is also true for items. For example, some movies would be popular and become hot topic so that lots of people would watch it. In result, for this kind of popular movies, it could have high bias during some time. Then it would reduce because there are more new movies. And some items would have strong relationship with other items. For example, Harry Potter is a series movie. If a user doesn’t like or never watch Harry Potter I or Harry Potter II, it would be likely this user doesn’t like to watch other Harry Potter movies. In a word, if we could change bias with time and add weights which represent this strong relationship, it is possible to get a higher accuracy.

For the user based collaborative filtering approach, we use SVD do reduce dimension and similarity between users to predict ratings. While for a sparse matrix, it would be difficult to do SVD. And it is also really time consuming to compute a large matrix. It also need lots of space to store the matrix. Thus, we need to find some way to improve the efficiency and calculate ratings in a more suitable to get a higher accuracy.

References

Ricci, F, Rokach, L and Shapira, B 2015, *Recommender systems handbook,* 2nd edn. Springer, London. ISBN: 978-0-387-85819-7

Koren Y., Bell R. (2015) Advances in Collaborative Filtering. In: Ricci F., Rokach L., Shapira B. (eds) Recommender Systems Handbook. Springer, Boston, MA

Bell, R., and Koren, Y., “Lessons from the Netflix Prize Challenge”, SIGKDD Explorations 9 (2007), 75–79.

Bell, R.M., Koren, Y., and Volinsky, C., “Modeling Relationships at Multiple Scales to Improve Accuracy of Large Recommender Systems”, Proc. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007.

Bennet, J., and Lanning, S., “The Netflix Prize”, KDD Cup and Workshop, 2007. www.netflixprize.com.

Canny, J., “Collaborative Filtering with Privacy via Factor Analysis”, Proc. 25th ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR’02), pp. 238–245, 2002.

Herlocker, J.L., Konstan, J.A., and Riedl, J., “Explaining Collaborative Filtering Recommendations”, Proc. ACM Conference on Computer Supported Cooperative Work, pp. 241–250, 2000.

Herlocker, J.L., Konstan, J.A., Borchers, A., and Riedl, J., “An Algorithmic Framework for Performing Collaborative Filtering”, Proc. 22nd ACM SIGIR Conference on Information Retrieval, pp. 230–237, 1999.

Hofmann, T., “Latent Semantic Models for Collaborative Filtering”, ACM Transactions on Information Systems 22 (2004), 89–115.

Kim, D., and Yum, B., “Collaborative Filtering Based on Iterative Principal Component Analysis”, Expert Systems with Applications 28 (2005), 823–830.

Koren, Y., “Factor in the Neighbors: Scalable and Accurate Collaborative Filtering ”, ACM Transactions on Knowledge Discovery from Data (TKDD),4(2010):1–24.

Linden, G., Smith, B., and York, J., “Amazon.com Recommendations: Item-to-Item Collaborative Filtering”, IEEE Internet Computing 7 (2003), 76–80.

Marlin, B.M., Zemel, R.S., Roweis, S., and Slaney, M., “Collaborative Filtering and the Missing at Random Assumption”, Proc. 23rd Conference on Uncertainty in Artificial Intelligence, 2007.

Paterek, A., “Improving Regularized Singular Value Decomposition for Collaborative Filtering”, Proc. KDD Cup and Workshop, 2007.

Salakhutdinov, R., Mnih, A., and Hinton, G., “Restricted Boltzmann Machines for Collaborative Filtering”, Proc. 24th Annual International Conference on Machine Learning, pp. 791–798, 2007.

Sarwar, B.M., Karypis, G., Konstan, J.A., and Riedl, J., “Application of Dimensionality Reduction in Recommender System – A Case Study”, WEBKDD’2000.

Takács G., Pilászy I., Németh B. and Tikk, D., “Matrix Factorization and Neighbor based Algorithms for the Netflix Prize Problem”, Proc. 2nd ACM conference on Recommender Systems (RecSys’08), pp. 267–274, 2008.

Toscher, A., Jahrer, M., and Legenstein, R., “Improved Neighborhood-Based Algorithms for Large-Scale Recommender Systems”, KDD’08 Workshop on Large Scale Recommenders Systems and the Netflix Prize, 2008.

Wang, J., de Vries, A.P., and Reinders, M.J.T, “Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion”, Proc. 29th ACM SIGIR Conference on Information Retrieval, pp. 501–508, 2006.