Recommender system Using Collaborative Filtering

Yuhan He(z5224997), Yuliang Wang(z5191313), Zihao Xu(z5184152)

August 11, 2019

1 Introduction

The rapid development of the Internet and the explosion of Information have made it increasingly more challenging and difficult to gain accurate and useful information promptly. Faced with the sheer amount of information, how to extract desired information as soon as possible definitely is a very important topic. Amongst the technologies to extract information, recommender system using collaborative filtering undoubtedly is a quite effective one, which has already been widely applied in social web and applications like YouTube, Reddit and Last.fm. Once users log in these webs and applications, those companies will try to recommend the popular contents to users. The collaborative filtering can calculate the collected datasets of users and make recommendations through data of users who shared a similar interest to current other users. Obviously, such a recommender system based on the existing dataset can make personalised recommendations through analysis of information collected previously(1).

Nowadays, Collaborative Filtering(CF) has been widely used in e-commerce and entertainment websites and applications. CF combines and extracts from the vast amount of data to perform a prediction and recommendation of appropriate items which are most likely to be favoured to the users or consumers based on the calculated similarity between them and other users. In terms of the features of CF, it is able to filter information which is difficult for filtering machine to analyze like artworks. And, it can utilize the shared experience, avoiding the inaccuracy and the incompleteness of contents analysis. Moreover, CF can filter information which is hard to be quantized. Another feature is that CF is able to recommend new information and find the incomplete similarity information which is unpredictable It provides a service for users to discover the undiscovered interests and preference.

As the mechanism of CF is based on the previous data of users which have been collected, it raised a problem called "cold-start". It is hard for it to do the prediction and recommendation to a new user/item. And sparsity and scalability are also a problem which CF will be faced with.

In the project, we focus on the application of CF in the movies recommender system. We aim

to adopt the memory-based approach and the Matrix factorization to implement the recommender systems, and try to evaluate the performance of recommender systems through some criterion.

2 Related Work

The implementation of CF came to realize with Tapestry and it is a Collaborative Filtering based recommender system. At the time, it is only able to extract and recommend with the knowledge collected from a community where people had already known each other(2). This is the earliest implementation of CF used in recommender system. Afterwards, several systems came out and they were more developed than Tapestry and they were capable of extracting information from a bigger dataset. There are many technologies added to implement the recommender systems, like Bayesian networks, clustering, and Horting. Schafer et al. succeeded in providing personalized recommendations and calculating the level of royalty from customers in a detailed way and examples of recommender system used in E-commerce(2). Above are the common technologies which are widely used in recommender systems.

The traditional Collaborative Filtering is based on neighborhood models. There are two kinds of neighborhood models, namely the user-oriented CF approach and an analogous item-oriented approach. Researchers have tried many methods to improve the performance in CF-based recommender systems. The performance of CF has been improved by introducing the knowledge graph(3). The results show that the new feature of graph significantly improve the performance of prediction accuracy in recommending. What's more, such an innovation also makes progress in the well-known problems of CF, cold start situation(3). Another innovation on CF is represented with Neural network-based CF, which gained significant progress in recommendation performance(4). The related work is not limited to the innovations mentioned above, and many researchers came up with different and innovative methods to improve CF performance.

Our aim is to implement several CF algorithms and compare the performance of the recommender systems through the root-mean-square deviation.

3 Methond

3.1 Memory-based CF

User rating data is used to calculate the similarity between users or items in memory-base approach. User-based and Item-based CF are two typical examples but there are differences between them. User-based CF pays more attention to a smaller community which share similar interests,

while Item-based CF can make more personalized recommendations to customers and have a good explanation for the recommendation(5).

The user-based CF focuses on similarity and recommend movies that users of similar interests rated highly in the dataset. The user-based CF takes a lot of memory to compute and store the similarity matrix, but still very effective. For the process, it would calculate the information for every user, which takes up the majority time of the recommender system.

For the similarity matrix, the most determinant factor is how to measure the similarity between two pairs of users. Three common metrics are usually used in the CF, namely Jaccard Similarity1, Cosine Similarity2, and Pearson Similarity3.

$$sim(a,b) = \frac{|N(a) \cap N(b)|}{|N(a) \cup N(b)|} \tag{1}$$

$$sim(a,b) = \frac{|N(a) \cap N(b)|}{\sqrt{|N(a)||N(b)|}} \tag{2}$$

$$sim(a,b) = \frac{cov(A,B)}{\sigma_A \sigma_B} = \frac{\sum_{i=1}^{n}(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \bar{A})^2 (B_i - \bar{B})^2}} \tag{3}$$

The sim stands for the similarity between a and b, and $N(x)$ stands for the movies of user x rates. $|N(x)|$ stands for the number of movies rated by user x.

We adopted the second similarity calculation approach2, namely Cosine Similarity. After the calculation of similarity matrix, the k which is a given value most similar users are selected. Then the intersection set of movie which is rated by the k most similar users is computed. Afterwards, we measure the priority of recommending these movies. If these movies have been rated by the most similar users, we set its priority higher. The process can be expressed in a formula below4:

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

Above, S stands for the set of k most similar users, $N(i)$ stands for the set of users who rated movie I previously, Sim stands for the similarity of users u and v, r_{vi} stands for the interest user v has on a movie i and here r_{vi} is set to 1.

For item-based CF, all the procedures are the same as user-based CF. The only difference is that the shape of matrix item-based CF used is the transpose shape of matrix user-based CF used.

3.2 Matrix Factorization

Basic matrix factorization model is a model-based collaborative filter. As the memory-based recommender systems are quite easy and straightforward to implement, there still exist some tricky problems such as the cold start problem.

Matrix Factorization (MF) models map both users and items to a joint latent factor space of dimensionality(6). The principle for the MF method is to factorize the original matrix into two or more matrices which can be used to discovered hidden latent features. In a short, the main method for MF is to form the used-feature and feature- item matrix. One advantage for the MF is that the preference of users and the feature of items are obtained. Another advantage is that the dimensionality of the matrix is reduced. The MF method aims to calculate and predict every user-item rating through learning during the iterations. During the implementation, we used the MF method to predict the rating which does not exist before(6).

Especially, we use Singular Value Decomposition (SVD). SVD can discover the features of users and items, and make predictions based on these features. Compare to other factorization algorithms, there is no restrictions for SVD and it is easier to implement(7)

The SVD can factorize a rectangular matrix $\boldsymbol{A_{n\times p} = U_{n\times k} S_{k\times k} V^T_{k\times p}}$ where $\boldsymbol{U^T U = I_{n\times n}}$, $\boldsymbol{V^T V = I_{p\times p}}$, $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal.

For the convenience of calculating the regression computation by gradient descent, the middle part of decomposition is ignored. $\boldsymbol{R_{u\times i}}$5 is the user-movie matrix and it is divided into two matrices $\boldsymbol{P_{u\times k}}$ and $\boldsymbol{Q_{k\times i}}$, which can be multiplied into the previous matrix $\boldsymbol{R_{u\times i}}$.

$$\boldsymbol{R_{u\times i} \approx P_{u\times k} \times Q_{k\times i} = \hat{R}_{u\times i}} \tag{5}$$

However, during the generating process of ratings, they must be influenced by some subjective factors which we can call "biases". There are many reasons for the biases such as the subjective preference of the user. To solve the problem and have a better model, we add the biases(6).

$$b_{ui} = b_u + b_i + \mu \tag{6}$$

The b_{ui} stands for the bias, and b_u and b_i mean the deviation value from the average rating μ. Therefore, taking account the bias b_{ui} into our matrix factorization, we can get the equation below:

$$\boldsymbol{\hat{R}_{u\times i} = q_i^T p_u} + b_u + b_i + \mu \tag{7}$$

The loss function8 is defined as the following, where λ is a parameter to avoid overfitting. We set

$\lambda = 0.002$.

$$E = \sum_{(u,i)\in k} (r_{ui} - \mu - b_u - b_i - \boldsymbol{q_i^T p_u})^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2) \tag{8}$$

For better performance, we try to minimize the loss function, which can be achieved by introducing the prediction error. Here a stochastic gradient descent optimization is used to minimize the loss function. For every user in the training set, the recommender system predicts r_{ui} and computes the associated prediction error e_{ui}9(6).

$$e_{ui} = \boldsymbol{\hat{R}_{u\times i}} - \boldsymbol{q_i^T p_u} \tag{9}$$

With the prediction error, we then have the following rules to update the biases and latent matrices (γ is the learning rate in gradient, we set $\gamma = 0.001$)(6)(7)(8).

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \tag{10}$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \tag{11}$$

$$p_u \leftarrow p_u + \gamma(e_{ui} \times q_i - \lambda p_u) \tag{12}$$

$$q_i \leftarrow q_i + \gamma(e_{ui} \times p_u - \lambda q_i) \tag{13}$$

The aim of SVD is to discover the hidden features behind the items, namely the movies in our project. What's more, SVD can discover the user's preference towards different features. Moving into the process of recommendation after we set the number of hidden features, the recommender system would predict a movie rating of a user or recommend the movies that are likely to be favoured.

3.3 SVD++

SVD++ is the derived from SVD and achieves better performance on prediction because of the introduction of implicit feedback information. The prediction made by SVD++ model can be defined as:

$$\tilde{r}_{ui} = \mu + b_u + b_i + \boldsymbol{q_i^T} \times \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} y_j \right) \tag{14}$$

Where $|R(u)|$ is the number of items rated by user u and y_i represents the characteristics of the items. Using this model, we are able to get a better user bias15.

$$\min_{P,Q} \sum_{r_{ui} \in R} \left[r_{ui} - \mu - b_u - b_i - \boldsymbol{q_i^T} \times \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right)^2 + \lambda(b_u^2 + b_i^2 + ||p_u||^2 + ||q_i||^2) \right] \tag{15}$$

Similar to the SVD, the SVD++ also have update the corresponding biases and matrices by minimizing the regularized squared error as much as possible(9).

3.4 Multilayer preceptron

Multilayer Perceptron(MLP) is under the class of feed forward artificial neural network which means each neuron receives input only from the neurons in the previous layer. MLP network consists of 1 input layer, 1 output layer, 1 or more hidden layers. Back propagation is used in MLP to do the training. MLP is an effective network and can be adjusted according to how accurate the desired result is. Due to the feature, MLP has been widely used and become the foundation of many state-of-the-art approaches. A example of MLP Network is shown in *figure 1*1.

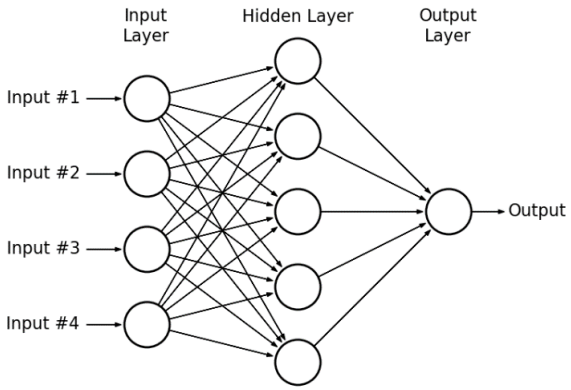

Figure 1: Multilayer preceptron Network

The activation function we used is the hyperbolic tangent function16. The range of hyperbolic tangent function is (-1, 1) which means the value of each neurons is in the range (-1, 1). The loss function we used is the mean squared error(mse)17.

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{16}$$

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{17}$$

In this project, the design of the MLP network structure is demonstrated as *figure 2*2(10).

In our design, we used one input layer, two fully connected hidden layers and one output layer.

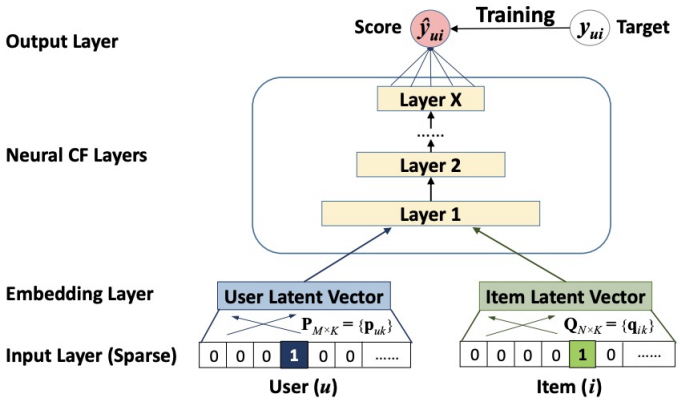

Figure 2: Neural collaborative filtering framework(10)

We randomly initialized user and movie input vector into two embedding vectors where each vector has 30 dimensions. Then, we used the first one hidden layer with activation function. The second hidden layer has a drop rate $\beta = 0.5$ to avoid overfitting. Finally is the output layer which has one single output neuron.

4 Experiments

The datasets we used to train each method are shown in *table 1*.4 There are two datasets in different data size, the MovieLens has 100k records and 1M records separately.

Datasets Stats			
Datasets	Number of users	Number of movies	Number of ratings
ML 100k	943	1682	100000
ML 1M	6040	3706	1000209

Table 1: Statistics of Datasets

The root-mean-square deviation(RMSE)18 is used as the evaluation metric to measure the differences between the values predicted by recommender systems and the actual values.

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|}\sum_{\hat{r}_{ui}\in\hat{R}}(r_{ui}-\hat{r}_{ui})^2} \tag{18}$$

4.1 Results

We have implemented SVD and MLP method, selecting user-based CF, item-based CF and SVD++ as comparison methods. What's more, 100k and 1M datasets are tested separately as comparison. In terms of allocation of the training set and testing set, 80% and 50% are allocated as training set

separately, while the corresponding remaining data are allocated as testing set. The performances are compared in terms of methods, sizes of datasets, and the allocation proportion of training and testing datasets.

The following table4.1 shows the average RMSE of each methond we used on datasets with different data size.

Methond \ Datasets	ML 100k		ML 1M	
	80%	50%	80%	50%
User-based	1.0171	1.0300	0.9768	0.9824
Item-based	1.0286	1.0479	0.9992	1.0122
SVD	0.9180	0.943	0.8720	0.875
MLP	1.0416	1.0788	0.9607	0.9840
SVD++	0.9195	0.9393	0.8612	0.8852

Table 2: Average RMSE for Datasets

1. SVD and SVD++ achieved best performance among the five methods.
2. With the size of dataset larger, RMSE gets smaller.
3. Under the same size of dataset, RMSE is smaller with larger training set.

5 Conclusion

Our project successfully implemented the recommender system using the five Collaborative Filtering Algorithms to predict the rating for the MovieLens datasets. The results show the SVD and SDV++ performed best amongst these algorithms.

While the MLP does not perform as well as expected, the reason may be attributed to the inappropriate matching of the data and the multilayer perceptron, leading to the relatively unsatisfactory results.

The MLP does not show satisfactory results as expected. The reasons may be insufficient number of layers implemented, as it is a trade-off between running time and performance. We can optimize and achieve smaller RMSE through adding more layers, which could need larger computation and more time.

SVD++ does not perform better in the dataset of 100k than SVD. The reason may be that the size of dataset is not quite large enough.

By comparison, user-based CF has relatively better results than item-based CF. Also, user-based CF need less computational cost than item-based CF.

References

[1] P. Resnick and H.-R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, pp. 56–58, 1997.

[2] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

[3] R. Mu and X. Zeng, "Collaborative filtering recommendation algorithm based on knowledge graph," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

[5] G. Linden, B. Smith, and J. York, "Recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, pp. 76 – 80, 2003.

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, pp. 30–38, 2009.

[7] C.-C. MA, "A guide to singular value decomposition for collaborative filtering," *IEEE Computer(Long Beach, CA)*, pp. 1–14, 2008.

[8] Byron_NG, "From SVD to Recommender System," 2018. https://www.cnblogs.com/bjwu/p/9358777.html, accessed 01/08/2019.

[9] Z. Xian, Q. Li, G. Li, and L. Li, "New collaborative filtering algorithms based on svd++ and differential privacy," *Mathematical Problems in Engineering*, vol. 2017, 2017.

[10] X. He, L. Liao, H. Zhang, N. L, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.