Recommendation System Report

Weishi Guo(wg7qn), Tianyang Luo(tl2sf)

1 Abstract

Nowadays more and more companies try to bid on Google advertisements to show their products, for example, Amazon's advertisements will show up in your Chrome browser based on your purchase history or browsing history, in order to motivate you to buy more Amazon products. The data shows us that by implementing a recommendation system has increased Amazon's revenue by 35 percent.

2 Introduction

The main goal of recommendation system is that, given a UserItemRating matrix R where each row represents a user i, each column represents an item j and $R_{i,j}$ represents the rating of user i to item j. However there are some users who have no ratings on some items, and therefore we want to build some models to approximate these missing ratings.

In our project, we did some researches on some famous recommendation algorithm that can be used to help us approximate missing information. The details of these algorithms will be discussed in section 3-7 and we showed our test results of implementing these algorithms on a data MovieLens in section 9.

3 Related Works

There are two types of typical Collaborative Filtering Algorithms: Memory-based and Model-based Collaborative Filtering. We will explain them in details in the following sections.

3.1 Memory-based Collaborative Filtering

Memory-based Recommendation is directly based on previous ratings in the stored matrix that describes user-item relations. There are two famous memory-based collaborating recommendation:(1)User-based CF: Users with similar previous ratings for items are likely to rate future items similarly. (2)Item-based CF: items that have received similar ratings previously from users are likely to receive similar ratings from future users [4].

The key idea of these two memory-based collaborative filtering method is that they depend on the similarity between users and items. The similarity can be measured by methods such as Cosine Similarity and Pearson Correlation Coefficient[5].

3.2 Model-based Collaborative Filtering

The model-based CF assumes that an underlying model governs how users rate items. The famous example of model-based CF is by using matrix factorization method such as SVD, SVD++.

One major difference between Memory-based and Model-based models is that memory-based model is in some way deterministic because it is based exactly on the original matrix is. On the other hand, the model-based models can be trained with some hyper-parameters and the final accuracy is determined by both original matrix and the hyper-parameters are been tuned.

3.3 Challenges of recommendation systems

- 1.The sparsity of the data: Both Model-based and memory-based CF methods rely heavily on previous data. However the density of available ratings in commercial recommendation systems is often less than 1 percent [5].

- 2. Users with no ratings Memory-based CF method depends on the similarity between one user i to other users. However, if the user i has no ratings on any item, then the similarity can not be calculated and therefore Memory-based CF does not work in this case.

- 3.Users' relationship In reality, the users' relationship are very important. Think about when you want to buy an item, you will ask your friends for recommendations. But, in the traditional CF methods do not consider the social networks of users.

4 Simple Matrix Factorization

4.1 Implementation of Simple Matrix Factorization

The naive matrix factorization [6] is to factorize the rating matrix $R_{|U|X|I|}$ where $|U|$ denotes the number of users and $|I|$ denotes the number of items by finding out two matrix P and Q such that $\hat{R} = PQ^T$. Finally you can approximate $\hat{R} = PQ^T$ with a objective function:

$$min \sum_{i,j} |R_{i,j} - \hat{R}_i j|, i \in |U|, j \in |I| \quad (1)$$

By using this method, each row $P_i \in R^{1XK}$ of P represents the latent feature of user i, each row of $Q_j \in R^{1XK}$ represents the latent feature of item j. K denotes the dimension of latent feature vector. The rating of user i to item j can be represented as:

$$\hat{R}_{i,j} = P_i Q_j = \sum_{k=1}^{K} p_{i,k} q_{k,j} \quad (2)$$

The difference $e_{i,j}$ between predicted rating and actual rating can be represented as:

$$e_{i,j}^2 = (R_{i,j} - \hat{R}_{i,j})^2 = (R_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j})^2 \quad (3)$$

Then we can calculate the gradients of $e_{i,j}^2$ with respect of $P_{ik}, Q_{k,j}$ as following:

$$\frac{\partial e_{i,j}^2}{\partial P_{i,j}} = -2 e_{i,j} Q_{k,j} \quad (4)$$

$$\frac{\partial e_{i,j}^2}{\partial Q_{i,j}} = -2 e_{i,j} P_{i,k} \quad (5)$$

Then we can use gradients to update $P_{i,k}, Q_{k,j}$ and $k \in |K|, i \in |U|, j \in |I|$ with a learning rate α as followings:

$$\forall P_{i,k} \in P, P'_{i,k} \leftarrow P_{i,k} + 2\alpha * e_{i,j} Q_{k,j} \quad (6)$$

$$\forall Q_{k,j} \in Q, Q'_{k,j} \leftarrow Q_{k,j} + 2\alpha * e_{i,j} P_{i,k} \quad (7)$$

4.2 The Challenge of Naive Matrix Factorization

For the implementation of naive matrix factorization method, it is necessary to set the missing values to be some number initially in order to make $R = PQ^T$ feasible. Normally, people will initialize missing value to be zero, and the reason is that we only need to we are not really trying to come up with P and Q such that we can produce R exactly. Therefore, even though we set missing entries of R to be 0, we mainly focus on minimise the errors of the available user-item pairs.But we believe there are some problems of by initializing with zero, because each entry will impact the results of P and Q that we will get. Also, by setting the missing value $R_{u,i} = 0$ is equivalent to we assume the user u gives a rating of 0 on the item i. Therefore, the results of P,Q will not be accurate or reasonable in some way.

So we think it would be reasonable if we set missing value $R^*_{u,i}$ to be the average rating of item i by equation:

$$R^*_{u,i} = \frac{\sum_u^{|U|} R_{u,i} * I(R_{u,i})}{\sum_u^{|U|} I(R_{u,i})} \quad (8)$$

$I(R_{u,i})$ is a indicator, and it equals to 1 if $R_{u,i}$ can be observed and equals to 0 if $R_{u,i}$ is missing. We will show our test results of these two different settings' influence on the accuracy in the following section.

5 SVD Method

The Singular Value Decomposition(SVD) recommendation algorithm was popularized by Simon Funk during the Netflix Prize and the team won 100M dollars because of their SVD recommendation model increased the movie prediction accuracy [2][3].

The key idea of this model is that we can define a latent feature vector $u \in R^{nx1}$ for an movie j. The intuition of thinking about this movie latent feature vector is that every movie's preference by a user can be determined by this movies' type, release date and so on. We can also define a latent feature vector $v \in R^{nx1}$ for user latent feature vector.

Then the rating of user i on movie j can be represented by

$$RatingsMatrix[user_i][item_j] = u^T v \quad (9)$$

5.1 SVD mathematical description

We define that:

- M is User-Item-Rating-Matrix
- U is user latent feature matrix
- I is item latent feature matrix
- Σ=diag$(\sigma_1, ..\sigma_k)$ which is diagonal matrix of singular values

Then M can be calculated as $M = U\Sigma I$ by SVD as shown in below figure.

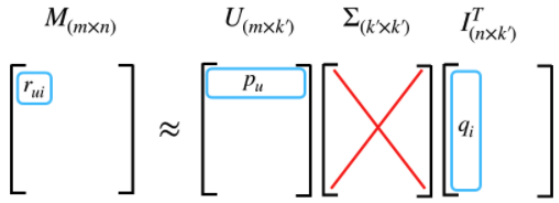


Figure 1: Figure: SVD[1]

By this SVD representation, each user is associated with a row vector p_u which measures the user u's potential feature vector in k dimension, and each item is associated a column vector q_i which measures the item's potential feature vector in k dimension. Then the results of $p_u q_i$ gives us an sense of the interaction between user u and item i.

Even though there is still a σ_1 term, the singular value only provides a scaling factor instead of feature factor, so we can ignore the singular value term, because we only care about the interaction between user u and item i in this case.

5.2 SVD algorithm implementation

1.Setup

The approximation of $\hat{r}$ can be represented by

$$\hat{r}_{u,i} = \mu + b_i + b_u + p_u q_i \tag{10}$$

μ denotes the overall average rating, b_u and b_i denote he observed deviations of user u and item i. If the is no rating from user u on item i, then b_u, b_i, p_u, q_i are initialized with 0.

2.Objective function

There are four hyper-parameters that we need to tune: b_i,b_u are the deviations, $p_u q_i$. Also we want to include a regularization term denoted R to prevent over-fitting issue, therefore the objective function is

$$\min_{b^*,q^*,p^*} \sum_{(u,i)\in K} (r_{u,i} - \mu - b_i - p_u q_i) + R \tag{11}$$

$$R = \lambda(b_i^2 + b_u^2 + ||qi||^2 + ||p_u||^2) \tag{12}$$

3.Gradient descent

During each iteration of training, given a rating $r_{u,i}$, we want to predict $\hat{r}_{u,i}$, and the difference between $r_{u,i}$ and $\hat{r}_{u,i}$ is denote as $e_{u,i} = r_{u,i} - \hat{r}_{u,i}$. Finally we can calculate gradient of four parameters by:

$$\frac{\partial \mathcal{L}}{\partial b_u} = \gamma(\lambda * b_u - e_{u,i}) \tag{13}$$

$$\frac{\partial \mathcal{L}}{\partial b_i} = \gamma(\lambda * b_i - e_{u,i}) \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial q_i} = \gamma(\lambda * q_i - e_{u,i} p_u^T) \tag{15}$$

$$\frac{\partial \mathcal{L}}{\partial p_u} = \gamma(\lambda * b_u - e_{u,i} q_i^T) \tag{16}$$

6 SVD++ Method

6.1 Description of SVD++

The previous SVD method only considers about explicit data such as user's rating on one item. However, there are some implicit information such the users' rating history that can be used to make the model more accurate. The SVD++ is a model that is improved based on SVD. One major difference is that, in SVD++, there is a parameter R(u) which is a list of user u's ratings that are available [7]. The prediction is represented as equation:

$$\hat{r}_{u,i} = \mu + b_i + b_u + q_i(+|R(u)|^{-\frac{1}{2}} \sum_j \in R(u) y_j) \tag{17}$$

where y_j is like a weight that can be trained.

6.2 Advantages of SVD++

There are several advantages of SVD++:

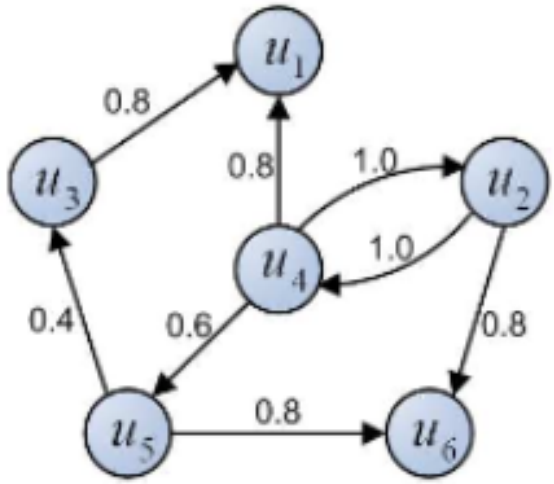
- **Explainable:** The traditional SVD is difficult to explain why recommend a item to a user, but SVD++ model can identify the users' previous rating information and the influence weight of each previous rating of user u can be calculated by SVD++
- **Accuracy:** The accuracy of SVD++ is higher than SVD, which is mainly because it considers about users previous rating information.

7 Probabilistic Matrix Factorization Method

The major problem of the methods that we discussed in the previous several sections is that those methods did not consider about the Social Network of users. The social network is very important, simply because when people want to buy some items, they will likely to ask their friends for recommendation.

7.1 Description

In paper [8], the authors proposed probabilistic matrix factorization method that combines the userItemRating matrix with social network graph as shown below. It is a directed graph, and each directed edge(u_i, u_j) means how much user i trust user j.



(a) Social Network Graph

Figure 2: Figure: Social Network Graph

The key idea of this is that we want to factorize social network matrix by $U^T Z$ and factorize user item rating matrix by $U^T V$. The matrix U denotes user latent feature matrix, V denotes latent feature matrix, and Z denotes factor matrix in the social network graph.

The probabilistic model depends on Bayes posterior distribution. There are three necessary posterior distributions that we need to calculate:

- (1) Social Network Matrix posterior distribution

$$p(U, Z|C, \sigma_C^2, \sigma_U^2, \sigma_Z^2) \propto p(C|U, Z, \sigma_C^2)p(U|\sigma_U^2)p(Z|\sigma_Z|\sigma_Z^2)$$

- (2) User-item Matrix rating posterior distribution

$$p(U, V|R, \sigma_R^2, \sigma_U^2, \sigma_V^2) \propto p(R|U, V, \sigma_R^2)p(U|\sigma_U^2)p(V|\sigma_V^2)$$

- (3) Social Recommendation posterior distribution

$$p(U, V, Z|C, R, \sigma_U^2, \sigma_V^2, \sigma_C^2, \sigma_Z^2)$$

where C is the observed social network matrix, R is the observed user-item rating matrix, σs are the pre-defined variances for matrix .

The key point is that we can combine social network matrix posterior and user-item matrix rating posterior to get social recommendation posterior. After we get posterior distribution of $p(U, V, Z|C, R, \sigma_U^2, \sigma_V^2, \sigma_C^2, \sigma_Z^2)$, recall that maximize the log-posterior is equivalent to minimize the sum-of-squared-error(MSE). Therefore, we can calculate the gradients of $\frac{\partial \mathcal{L}}{\partial U_i}, \frac{\partial \mathcal{L}}{\partial V_j}, \frac{\partial \mathcal{L}}{\partial Z_k}$, and finally we will get three matrices U,V,Z. Then the user u's rating on item i can be predicted as $U_u^T V_i$.

7.2 Advantages and Disadvantages of Probabilistic Matrix Factorization

The major advantage of this model is that it considers about social network's influence which makes this model to be more accurate. However, there are several disadvantages, such as it assumes user item matrix and social networks follows Gaussian Distribution and in reality it is hard for us to tell the actual distribution of data. Also, even though we can get social network of users, it would be hard for us to say how much the user i trust user j. For example, there is still a chance such that user i can still follow user j because user i hate or not trust user j.

8 Evaluation Method

8.1 Data set

MovieLens 100k data is a dataset which contains information such as users' age, occupation etc and also movie's type, release date. We use this dataset to do some experiments.

8.2 Test methods

8.3 1.Model Performance Measure

We found that Root Mean Squared Deviation(RMSD) is good indicators for measuring

our model. The lower error means the higher accuracy of our models. So we implemented RMSD equation as:

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x_i})^2}{N}} \quad (18)$$

2.Memory-based models

For two memory-based models, since these two methods are heavily depending on all ratings of a user, therefore we implemented these two methods on all 100k data. Assume the observed data as $r_i, j \in Observed$. After we get the approximated rating matrix $\hat{R}$, we can calculate RMSD by equation

$$RMSD = \sqrt{\frac{\sum_{i,j \in Observed}(r_{i,j} - \hat{r_{i,j}})^2}{N}} \quad (19)$$

$$N = 100000 \quad (20)$$

2.Model-based models

There are three model-based models that we implemented:(1)Simple matrix factorization,(2)SVD matrix factorization, (3) SVD++ matrix factorization.

The 100k data was splitted into 75k training dataset and 25k testing dataset. Both training dataset and testing dataset can be represented as a list of $(userindex, itemindex, rating)$ but with different size.

By using 75k training data set T, we tuned our model with some hyper-parameters such as latent feature dimension K, epoch number, learning rate. Then we can get the latent feature vectors of each user and each item based on 75k data. Finally, we can use these latent feature vectors to approximate the $\hat{r}_{i,j}, i, j \in$testing dataset, and the RMSD can be calculated.

9 Test Results

9.1 Memory-based Models' results

As shown in figure 3, we test our user-based, item-based model on 100k data with several different neighbor sizes 3,6,9,12,15 to find the influence of neighbor size on loss. The results showed that as neighbor size increases from 3 to 15, the loss decreases. This tells us as we considering more and more similar users with user i, the prediction of user i's ratings will more accurate. However, there is a threshold of neighbor size because as neighbor size larger than threshold, some non-similar users will be considered in the model.

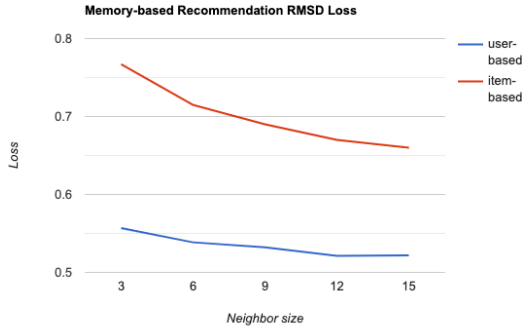


Figure 3: Figure: Memory-based Model RSMDLoss

9.2 Model-based models' results

The figure 4 shows the RSMD loss of SVD, SVD++, Simple Matrix Factorization method. We trained these models on the same training and testing dataset, as well as same epoch number(=20), learning rate(=0.0005), but with different latent feature vector dimension K. Overall, the Simple Matrix Factorization method has larger loss than SVD, we believe this is because SVD's good performance in feature extraction field, and there are several mathematical researches showed that. SVD++ has the lowest loss which is because SVD++ not only can extract feature vectors as well as SVD, but also it considers user's previous rating information. In terms of K, we found that as latent feature vector dimension increases, the loss increases for all three models. This coincides our intuition, each user should have only about 10-20 preference aspects.

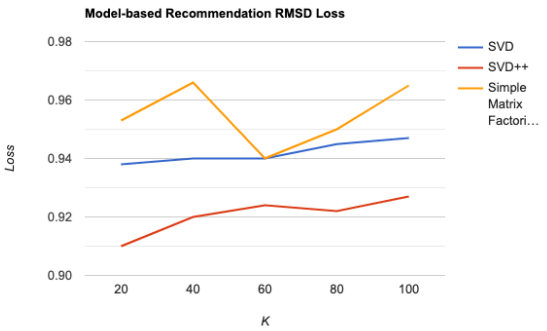


Figure 4: Figure: Model-based Model RSMDLoss

9.3 Influence of setting of missing value

As we discussed before, all matrix factorization methods assumes missing rating value of each user as 0 when calculating the feature vectors for train-

ing data set. So one question would be why not set each user i's missing ratings as the global mean rating or user i's average rating. Then we conducted an experiments to find out what is the influence of the setting of these missing rating values. The results are shown in figure 5.

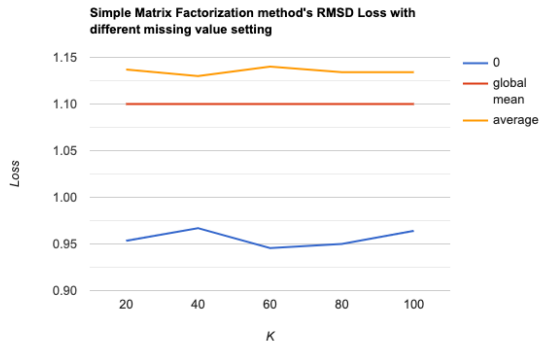


Figure 5: Figure: Simple matrix Fctorization with different missing value settings

The results showed that when we set all missing value to be 0 give us the best accuracy (minimum loss). This result explains why people normally set missing value to be 0 when they are using matrix factorization method. The reason that we think is that when we set the missing value to be non-zero, it actually provides the model some "fake information" to do the approximation of latent feature vectors. When missing values are set to 0, the model will focus on finding the latent features to maximize the observed rating information errors.

9.4 Social Network

As we talked before, probabilistic matrix factorization model can be used to increase the accuracy, because it considers about both social network and user-item ratings. However, the MovieLens do not provide users' trust relations. But some clustering methods such as K-means can be applied to find communities of users and we can assume users in the same group will trust each other. We implemented K-means to cluster users into 19 groups which is because there are 19 types of movies, so we want to cluster users with same interested movie types into same group.

Each user is represented as a vector $v_i = [r_{i,1}, r_{i,2}...r_{i,j}]$ where $r_{i,j}$ means the rating of user i on movie j. Because it is hard to visualize the clustering results of multidimensional matrix, so we used Principal Component Analysis on the matrix to plot the clustering results(figure 6).

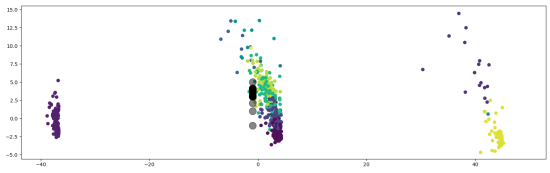


Figure 6: Clustering Result: each color represents one community

10 Conclusion and Future work

In this paper, we talked about several different ways to approximate the missing rating values of a user-item matrix. We analyzed the advantages and disadvantages of each model and how can we improve the model. However, the major issue is that we do not have the ground truth for those missing values and therefore there is no perfect way to say whether the model is good or not. Another big issue is that there are some known features of users and items are not been used, for example the users' age, occupation etc.

For the future work, we would like to find out a way to include those known features of users and items in the current several models. Also, the reason of why set missing value to be 0 when we are training the model-based model would be another interesting topic to discover.

References

[1] Hadian, S. (2019, July 16). Deploying a recommender system for the movie-lens dataset – Part 1. Retrieved May 09, 2021, from https://blog.codecentric.de/en/2019/07/recommender-system-movie-lens-dataset

[2] S. (2006, December 11). Netflix Update: Try This at Home. Retrieved from https://sifter.org/ simon/journal/20061211.html

[3] Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems handbook. New York: Springer.

[4] Li, J. (n.d.). Slide: Lecture 16. Recommendation I.pptx. Retrieved from http://www.ece.virginia.edu/ jl6qk/index.html

[5] Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the Tenth International Conference on World Wide Web - WWW 01. doi:10.1145/371920.372071

[6] Matrix Factorization: A Simple Tutorial and Implementation in Python. (n.d.). Retrieved from

http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/#source-code

[7] Koren, Y. (2008). Factorization meets the neighborhood. Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD 08. doi:10.1145/1401890.1401944

[8] Ma, H., Yang, H., Lyu, M. R., & King, I. (2008). SoRec. Proceeding of the 17th ACM Conference on Information and Knowledge Mining - CIKM 08. doi:10.1145/1458082.1458205

[9] MovieLens. (2021, March 02). Retrieved from https://grouplens.org/datasets/movielens/