An Factorization-Based Solution for Item Recommendation
CS420 Coursework: Item Recommendation

Zhenjia Xu
xzj, 515030910637
Shanghai Jiao Tong University
xuzhenjia1997@gmail.com

May 20, 2017

Abstract

Item Recommendation is a classic task of machine learning. There're various methods in this area. I have tried most of these methods, and finally choose matrix factorization[1] to solve this task. To get a better performance, I combined several results under different parameters. The overall performance of my model is not bad. The RMSE in the final test data was 1.32592.

1 Introduction

1.1 Background

In this task, our goal is to conduct a comprehensive recommendation system to predict the preference score of the given user on the specific items.
The strategies for recommendation system can be divided into two categories, namely, content filtering and collaborative filtering. Collaborative filtering is more suitable for this task. The two primary areas of collaborative filtering are neighborhood methods and latent factor models. The method I used to solve this task is matrix factorization, which is based on latent factor models.

1.2 Formulation

- Constructint the score matrix D from the training data.
- Extracting latent vector for each user and item.
- The predicted score of the given user on the specific items is:

$$\hat{r}_{u,i} = \mu + b_u + b_i + \boldsymbol{p}_u^T \boldsymbol{q}_i$$

- Training object:

$$\min_{P,Q} \sum_{r_{u,i} \in D} \frac{1}{2}\Big(r_{u,i} - (\mu + b_u + b_i + \boldsymbol{p}_u^T \boldsymbol{q}_i)\Big)^2 + \frac{\lambda}{2}(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2)$$

Table 1: Notations and descriptions

Notation	Description
D	The score matrix.
$\boldsymbol{p}_u$	The latent vector of the user.
$\boldsymbol{q}_i$	The latent vector of the item.
$r_{u,i}$	The true score of the given user on the specific items.
$\hat{r}_{u,i}$	The predicted score of the given user on the specific items.
$L(u, i, r_{u,i})$	The loss function of the objective.
λ	The hyperparameter of regularization.
μ	The global bias.
b_u	The user bias.
b_i	The item bias.

- The goal of the task is to minimize the RMSE of the prediction on the test data.

2 Methodology

2.1 Data Processing

I used one-hot encoding to process time information and features of the item.

2.2 Matrix Factorization

Matrix Factorization has several learning algorithms, such as stochastic gradient descent (SGD), alternating least-squares (ALS) optimization and Bayesian inference using Markov Chain Monto Carlo (MCMC). In these learning algorithms, I choose MCMC, because it has the best accuracy.

2.3 Ensemble

By matrix factorization method, I got several results under different parameters, and I choosed a simple but useful method to combine these results, that is, weighted average. Furthermore, the coefficients are different under different scores.

2.4 Some other methods

I also tried some other methods, such as k-nearest neighbors algorithm (KNN)[3], alternating least-squares (ALS), stochastic gradient descent (SGD), but their performance are not good enough.

3 Experiments and Results

3.1 Matrix Factorizsation

I used libFM[2] to implement matrix factorization method. The two most important hyper-parameters are number of iteration (n_{iter}) and dimention (dim). Because The complexity is closely related these two parameters, $n_{iter} * dim$ can't be too large. Here are several results

under different parameters.
By comparison, we can find that the dimention can't be too large nor too small. If the dimention is very small, the model is too simple to describe the characteristics of user and item. In contrast, if the dimention is very large, the model will be very likely to overfit.

id	n_{iter}	dim	$RMSE$
1	20	800	1.34607
2	30	700	1.34358
3	45	700	1.34255
4	60	600	1.34227
5	100	500	1.34215
6	150	300	1.34204
7	200	300	1.34058
8	250	300	1.34058
9	275	250	1.33990
10	300	200	1.34032
11	350	200	1.34063
12	400	200	1.34222
13	500	150	1.34420
14	800	120	1.35307

Table 2: RMSE under different parameters

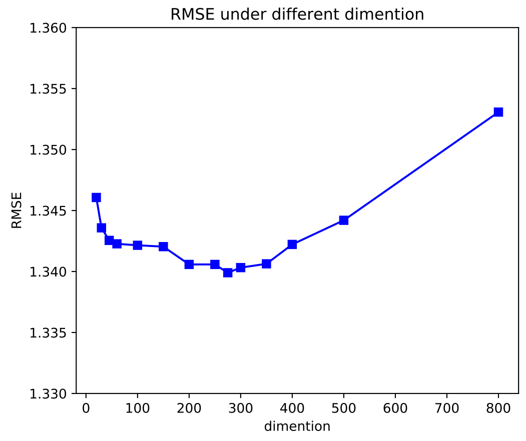

Figure 1: Comparison of different results

4 Ensemble

I used weighted average method to combine these results, which were gotten under different parameters.

$$\hat{r} = \sum_{i=1}^{14} w_i \hat{r}_i$$

Firstly, the weight of each result was set to the same, and the RMSE was to 1.32824, which was a great improvement.
Secondly, each result is given a different weight based on their performance. The better performances, the greater the weight. In this way, the RMSE was 1.32784.
Thirdly, I used some tricks to improve the peoformance. These tricks are quite effective, but I can give a formal explanation.

- By accident, I found that let the sum of the weights be slightly greater than one can improve the performance. Through several attempts, I found $\sum_{i=1}^{14} w_i = 1.025$ can get the best performance, and the RMSE was 1.32616.

  My explanation: If a user like an item very much, he will give a score of 5 without hesitation. However, each model has a different emphasis, and the average of these results will be lower than 5. As a result, the predicted score should be slightly raised.

- If the above explanation is true, the predicted score should be slightly raised only if the score is large. I tried to change the magnitude of raise according to the score, that is, the larger the score, the larger the magnitude of raise. The RMSE is 1.32584, which was a small improvement. (I find a bug in my code when I am writing the report, so the RMSE was 1.32592 during the competition)

Model	$RMSE$
Single model	1.33990
Average with same weight	1.32824
Average with different weight	1.32784
Trick1	1.32616
Trick1 + Trick2	1.32584

Table 3: RMSE of different models

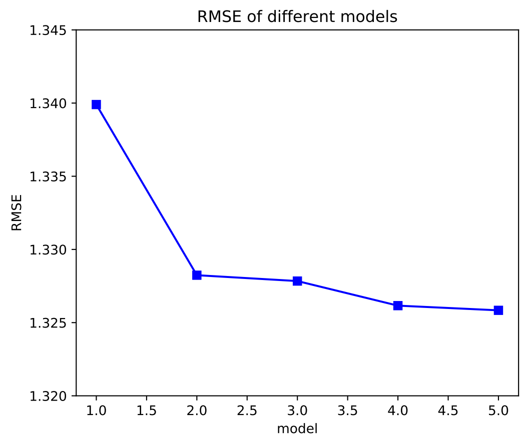

Figure 2: Comparison of different models

5 Discussion and Future works

Through this course work, I have mastered some basic methods of data mining, especially in the area of recommendation system. I have also thought of some other ideas but have no time to implement, for example, combine the existing features to get new features. Although the course work was finished, I will try these ideas when I have time. In summary, I enjoyed the pleasure given by the improvement of the performance and I will try my best to complete the next course work.

References

[1] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[2] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[3] Wikipedia. K-nearest neighbors algorithm — wikipedia, the free encyclopedia, 2017. [Online; accessed 19-May-2017].