华南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Biquan Wang, Li Li, and Yuchen Li

Supervisor:
Qingyao Wu

Student ID:
201821038853, 201820137894, and 201821038741

Grade:
Graduate

January 2, 2019

Recommender System Based on Matrix Factorization

***Abstract*—Matrix factorization can be used to discover latent features underlying the interactions between two or more different kinds of entities. So, one obvious application is to predict ratings in recommender system. In tihs paper, we use matrix factorization with stochastic gradient descent to gnerate a predict matrix from discrete data. For futher understanding, we try to show the result with different learn rate and different latent feature number K, and try to find the impact of these two prameters.**

I. Introduction

MATRIX factorization is one of the most popular methods for collaborative filtering recommendation, and stochastic gradient descent is the most common optimize way for black-box optimize problem. RMSE(Root mean Square Error)is a commonly used regression evaluation standard.

This paper aim to show the process of matrix factorization, we try two show the result with different learn rate and different latent feature number K, and try to find the impact of these two prameters.

II. Methods and Theory

In this part, we try to explain the process of matrix decomposition from idea to implementation. At the end, we explan the loss function and the update function of the factor matrix with stochastic gradient descent.

Example for this data set, we have a set U of users, and a set of D of movies. $\boldsymbol{R}$ is the rating matrix size $|U| \times |D|$. And assume that we want to find K latent features. So, what we should do is to find two matrix: $\boldsymbol{P}(|U| \times K)$ and $\boldsymbol{Q}(|D| \times K)$, such that their product approximates $\boldsymbol{R}$:

$$\boldsymbol{R} \approx \boldsymbol{P} \times \boldsymbol{Q}^T = \hat{\boldsymbol{R}}$$

In this way, $\boldsymbol{P}$ represent the strength of the associations between a user and the latent features, at the same times, $\boldsymbol{Q}$ represent the strength of the associations between a movie and the latent features.

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

In this experiment, We use gradient descent to get suitable $\boldsymbol{P}$ and $\boldsymbol{Q}$: initialize the two matrices with some value, calculate how "different" their product $\hat{R}$ to R, and try to minimize the difference interactively.

The difference here we called loss between the predict rating and real rating can use following function to calculate, we use squared error because the predict rating can be either higher or lower than real rating:

$$loss_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

Add regularization to avoid overfitting:

$$loss_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^{K} (||\boldsymbol{P}||^2 + ||\boldsymbol{Q}||^2) \quad (1)$$

In gradient descent, we need to know the gradient of p_{ik} and q_{kj}, so, we differentiate the above function with respect to these two variables separately:

$$\frac{\partial loss_{ij}^2}{\partial p_{ik}} = 2 loss_{ij} q_{kj} - \beta p_{ik}$$

$$\frac{\partial loss_{ij}^2}{\partial q_{kj}} = 2 loss_{ij} p_{ik} - \beta q_{kj}$$

So, the update function with learn rate η is:

$$p'_{ik} = p_{ik} + \frac{\partial loss_{ij}^2}{\partial p_{ik}} = p_{ik} + \eta(2 loss_{ij} q_{kj} - \beta p_{ik}) \quad (2)$$

$$q'_{kj} = q_{kj} + \frac{\partial loss_{ij}^2}{\partial q_{kj}} = q_{kj} + \eta(2 loss_{ij} p_{ik} - \beta q_{kj}) \quad (3)$$

RMSE:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (4)$$

III. Experiments

A. Dataset

Utilizing MovieLens-100k dataset. u.data – Consisting 10,000 comments from 943 users out of 1682 movies. At least, each user comment 20 videos. Users and movies are numbered consecutively from number 1 respectively. The data is sorted randomly

TABLE I
Snapshot of input data

userid	movieid	Rating	timestamp
196	242	3	881250949
ratings[n,0]	ratings[n,1]	ratings[n,2]	ratings[n,3]
12	203	3	879959583

B. Experiments steps

The steps of our matrix factorization are as the following:

1. Use loadtxt function in numpy library to load the MovieLens-100k and divided into train set and valid set.
2. Populate the original scoring matrix R against the row data, and fill 0 for null values.
3. Initialize the user factor matrix P and the item(Movie) factor matrix Q and K is the number of latent features.

4. Get a mini-batch data from train set randomly.
5. Update factor matrix P with function (2), and update factor matrix Q with fuction (3).
6. Calculate the loss of validation set with function (1).
7. Repeate setp 4-6 for n times, and return the losses of validation set.

C. Experiments Results

In this experiment we try to compare the impact of learn rate and latent factor K, so, the other hyper parameters is same: batch-size=10000, epoch=200, plenty=0.02.

Figure 1 is the result of $K = 2$, and different learn rate, we can find out that, with the increase of learn rate, the loss decrease faster. We try two use a large enough learn rate such as 0.02, 0.2 etc, but the program will overflow, so we can not show the overfitting result. Figure 2 is the result of $K = 3$, and different learn rate.

From Figure 1 and 2, we chose learn rate 0.005 as a local best learn rate and compare the result with different K. From figure 3, we can see, the loss begin with smaller number, with increase of latent factor K.

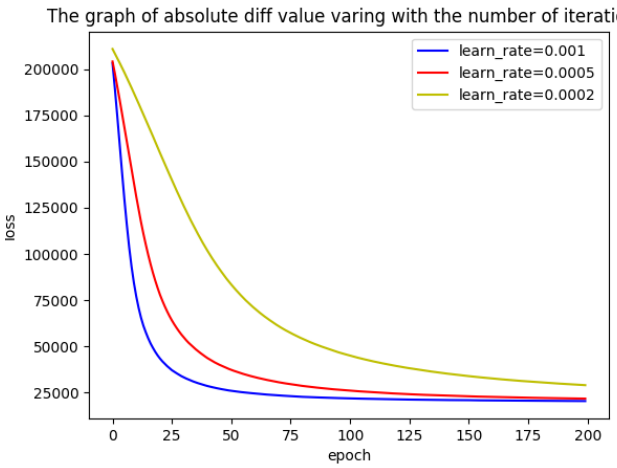

Fig. 1. Losses of different learn rate with k=2

Figure 4 and 5 shows the distribution of rating before algorithm, and figure 4 is the input matrix, witch contins 0(no rating).

Figure 6,7,8 shows the distribution of rating of predict matrix with K=3,6,9, from these pictures we can see that, there is no 0 rating, and all data tends to be normally distributed.

For further compare the impact of different latent feature, Figure 9 shows the RMSE value of predict matrix with different value of K. We can find out when K=12 the RMSE value is smallest.

IV. Conclusion

In tihs paper, we use matrix factorization with SGD to gnerate a predict matrix from discrete data. For easy to understanding, we compare the result with different learn rate and different latent feature number K, and try to find the

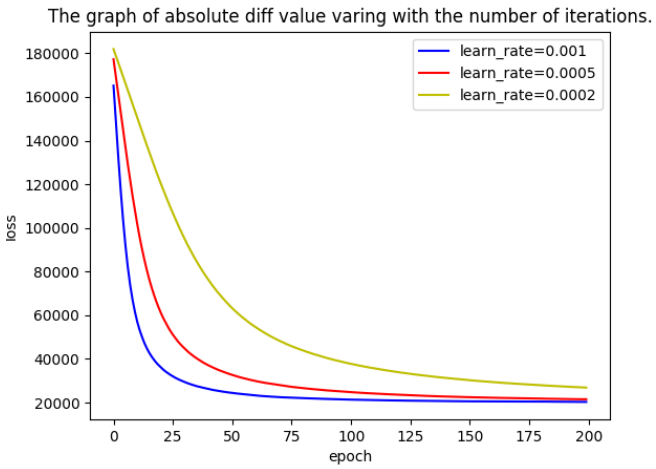

Fig. 2. Losses of different learn rate with k=3

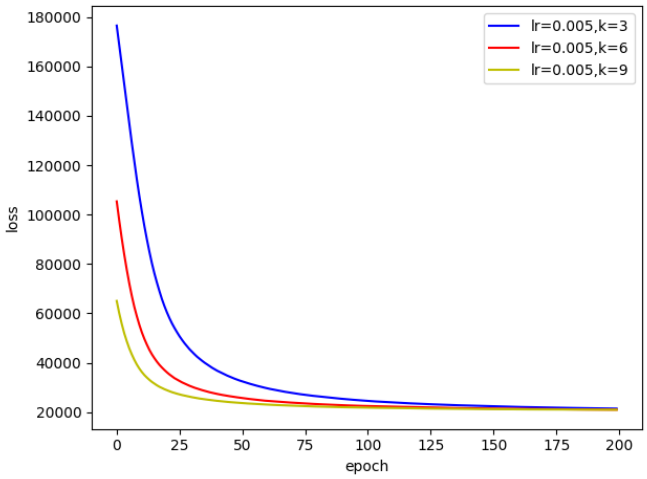

Fig. 3. Losses of different K learn rate 0.005

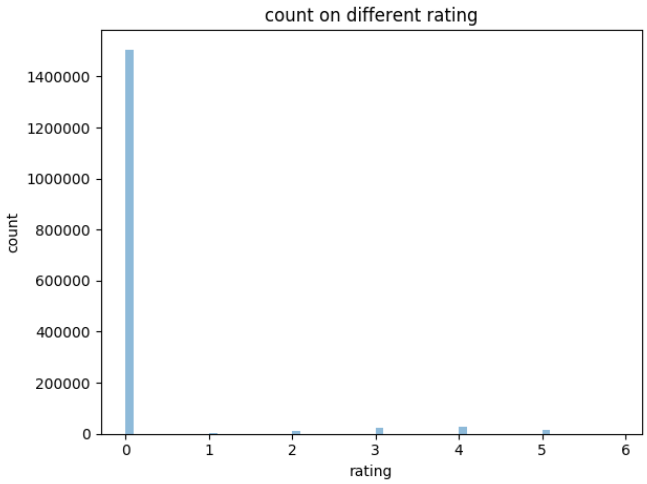

Fig. 4. Rating distribution with 0

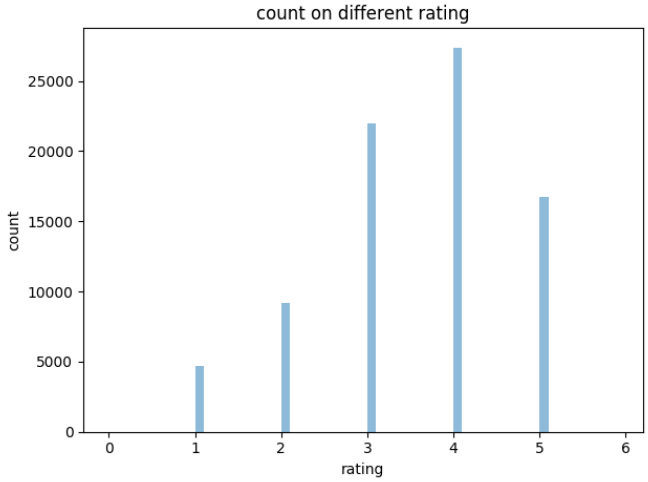

Fig. 5. Rating distribution without 0

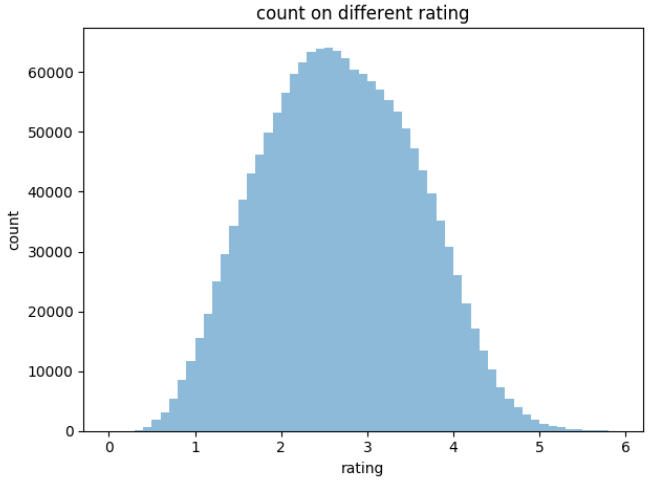

Fig. 6. Rating distribution after MF with K=3

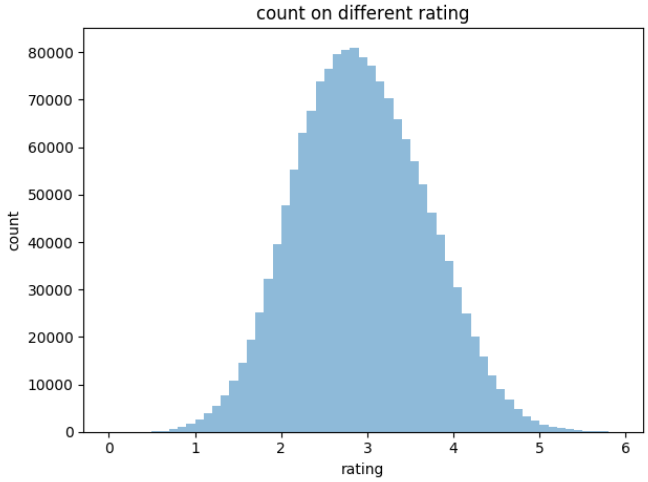

Fig. 7. Rating distribution after MF with K=6

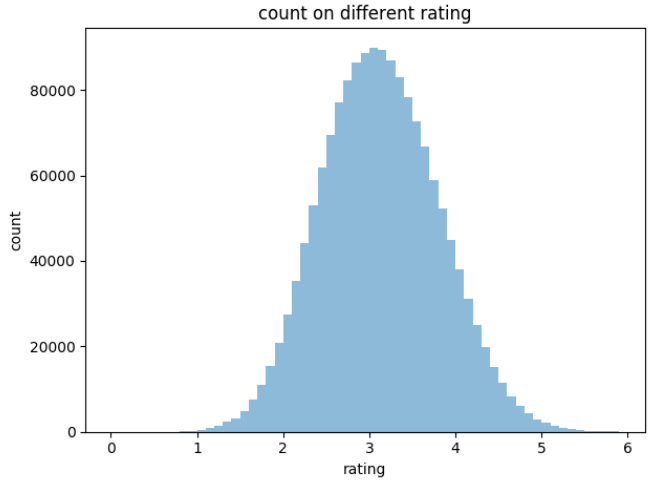

Fig. 8. Rating distribution after MF with K=9

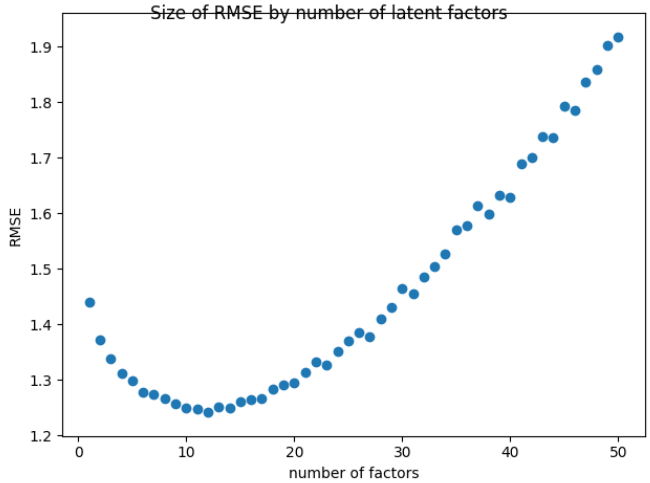

Fig. 9. RMSE with different latent feature number K

impact of these two prameters. We just finish the key step of a whole recommender system because of the time limitation, and we will compare other algorithm such as collaborative filtering, etc.