华南理工大学

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Qi Chen, Jing Liu, Yihan Zheng

Supervisor:
Qingyao Wu

Student ID:
201720144924, 201720145099,
201720145105

Grade:
Graduate

December 23, 2017

Reconmender System Based on Matrix Decomposition

***Abstract*—Collaborative filtering (CF) approaches proved to be effective for recommender systems in predicting user preferences in item selection using known user ratings of items. A major approach for this problem is matrix factorization (MF). In this report, we aim to conduct some experiments for further understanding of this algorithm. In experiment, we choose Stochastic Gradient Descent (SGD) to optimized the loss function.**

I. Introduction

RECOMMENDATIONS can be generated by a wide range of algorithms. While user-based or item-based collaborative filtering methods are simple and intuitive, matrix factorization techniques are usually more effective because they allow us to discover the latent features underlying the interactions between users and items. Of course, matrix factorization is simply a mathematical tool for playing around with matrices, and is therefore applicable in many scenarios where one would like to find out something hidden under the data.

For convenient, we design a loss function using matrix decomposition and use stochastic gradient descent (SGD), which is one of the most popular algorithms to perform optimization, to optimize the loss function.

The main purposes of this report can be concluded as the following:

- Explore the construction of recommended system.
- Understand the principle of matrix decomposition.
- Be familiar to the use of gradient descent.
- Construct a recommendation system under small-scale dataset, cultivate engineering ability.

II. Methods and Theory

In this section, we will give a complete introduction to matrix decomposition algorithm.

A. Matrix Decomposition

Having discussed the intuition behind matrix factorization, we can now go on to work on the mathematics. Firstly, we have a set U of users, and a set D of items. Let $\mathbf{R}$ of size $|U| \times |D|$ be the matrix that contains all the ratings that the users have assigned to the items. Also, we assume that we would like to discover K latent features. Our task, then, is to find two matrics matrices $\mathbf{P}$ (a $|U| \times K$ matrix) and $\mathbf{Q}$ (a $|D| \times Kmatrix$) such that their product approximates $\mathbf{R}$:

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}} \tag{1}$$

In this way, each row of $\mathbf{P}$ would represent the strength of the associations between a user and the features. Similarly, each row of $\mathbf{Q}$ would represent the strength of the associations between an item and the features. To get the prediction of a rating of an item d_j by u_i, we can calculate the dot product of the two vectors corresponding to u_i and d_j:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj} \tag{2}$$

Now, we have to find a way to obtain $\mathbf{P}$ and $\mathbf{Q}$. One way to approach this problem is the first intialize the two matrices with some values, calculate how 'different their product is to $\mathbf{M}$, and then try to minimize this difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference.

The difference here, usually called the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 \tag{3}$$

Here we consider the squared error because the estimated rating can be either higher or lower than the real rating.

To minimize the error, we have to know in which direction we have to modify the values of p_{ik} and q_{kj}. In other words, we need to know the gradient at the current values, and therefore we differentiate the above equation with respect to these two variables separately:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij} q_{kj} \tag{4}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij} p_{ik} \tag{5}$$

Having obtained the gradient, we can now formulate the update rules for both p_{ik} and q_{kj}:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj} \tag{6}$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik} \tag{7}$$

Here, α is a constant whose value determines the rate of approaching the minimum. Usually we will choose a small value for α, say 0.0002. This is because if we make too large a step towards the minimum we may run into the risk of missing the minimum and end up oscillating around the minimum.

Furthermore, to avoid overfitting, we add a parameter β and modify the squared error as follows:

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^{K} (||P||^2 + ||Q||^2) \tag{8}$$

TABLE I
FORMAT OF THE DATA IN MOVIELENS-100K DATASET

user id	tme id	rating	timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

In other words, the new parameter β is used to control the magnitudes of the user-feature and item-feature vectors such that $\mathbf{P}$ and $\mathbf{Q}$ would give a good approximation of $\mathbf{R}$ without having to contain large numbers. In practice, β is set to some values in the range of 0.02. The new update rules for this squared error can be obtained by a procedure similar to the one described above. The new update rules are as follows.

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik}) \quad (9)$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj}) \quad (10)$$

B. Stochastic Gradient Descent

Stochastic gradient descent (SGD) in contrast performs a parameter update for each training example $x^{(i)}$ and $y^{(i)}$:

$$\theta = \theta - \eta \nabla_\theta L(\theta; x^{(i)}; y^{(i)}) \quad (11)$$

III. EXPERIMENTS

In this section, we conduct some experiments for further understanding of matrix decomposition algorithm.

A. Dataset

We use MovieLens-100k dataset, which consists 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 are stored randomly as followings:

In this dataset, u1.base and u1.test are train set and validation set respectively, seperated from the whole dataset u.data with proportion of 80% and 20%.

B. Implementation

In this experiment, we use stochastic gradient descent(SGD) algorithm to optimize the loss function. The steps of our experiments are as the following:

1) Read the data set from indicated folder. We use u1.base as trianing data and u1.test as testing data directly. For convenient, we populate the original scoring matrix R against the raw data and fill 0 for null values.
2) Initialize the user factor matrix P and the item (movie) factor matrix Q, where K is the number of potential features.
3) Determine the loss function and hyperparameter learning rate η and the penalty factor λ.
4) Use the stochastic gradient descent method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:
   - Select a sample from scoring matrix randomly;
   - Calculate this sample's loss gradient of specific row(column) of user factor matrix and item factor matrix;
   - Use SGD to update the specific row(column) of P and Q;
   - Calculate the $L_{validation}$ on the validation set, comparing with the $L_{validation}$ of the previous iteration to determine if it has converged.
5) Repeat step 4 several times until the validation loss has converged to get a satisfactory user factor matrix P and an item factor matrix Q.
6) The final score prediction matrix $\hat{R}$ is obtained by multiplying the user factor matrix P and the transpose of the item factor matrix Q.

C. Experimental Results

In this section, we conduct several experiments to compare the performance using different hyper-parameters in this matrix decompositoin algorithm. In all of these experiments, we set learning rate $\alpha = 0.002$, weight of regularization $\beta = 0.02$ and run algorithms for 100 epochs. For measuring performance, we utilize Mean Absolute Error (MAE) index which is mean of all difference of elements in $\mathbf{R}$ and $\hat{\mathbf{R}}$.

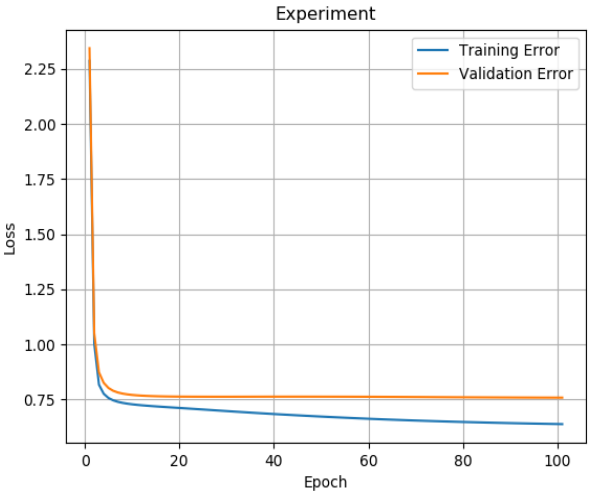

Fig. 1. We use Stochastic Gradient Descent (SGD) to optimize loss function mentioned before using hyper-parameter $K = 5$.

In Figure 1, obviously, we observe that validation loss firstly decreases during training loss minimizing. However, with time goes on, the validation loss almost no changes even the trianing loss declining.

As shown in Figure 2, we choose 6 values of K and conduct experiments respectively. Finally, we get these results together to compare performance with using different values. As we can see in Table II, when the value of K increases, the value of

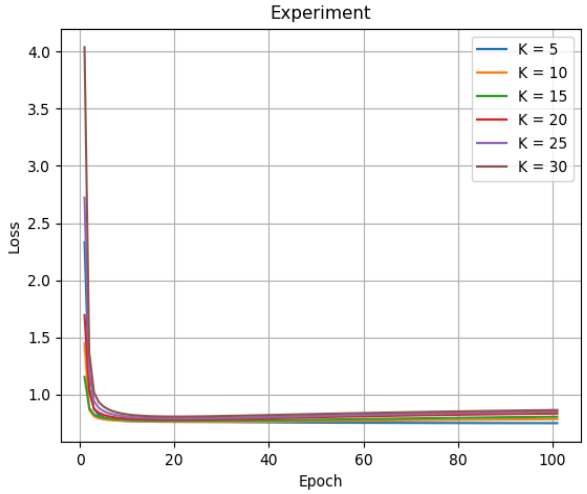

Fig. 2. For further understanding, we conduct another experiment to compare the impact of different values of K. The chosen values of K are $[5, 10, 15, 20, 25, 30]$.

TABLE II
IN THIS TABLE, WE SHOW THE FINAL VALUES OF VALIDATION LOSS USING DIFFERENT VALUES OF K.

value of K	5	10	15	20	25	30
validation loss	**0.7513**	0.7864	0.8069	0.8353	0.8519	0.8654

validation loss also increases. That means the performance of this algorithm becomes worse if value of K is too large.

IV. CONCLUSION

In this reprot, we have discussed the intuitive meaning of the technique of matrix factorization and its use in collaborative filtering. And we also conduct several experiments for recognizing the performance of this method. In fact, there are many different extensions to the above technique. That could achieve more promising performance than this basic method.