The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Anni Chen and Tianxin Cai and Shuyi Zhen

Supervisor:
Qingyao Wu

Student ID：
201530611142 and 201530611067 and 201530613771

Grade:
Undergraduate

December 9, 2017

Recommender System Based on Matrix Decomposition

Abstract—

In this experiment, we explore the construction of recommended system based on matrix decomposition. First, we populate the original scoring matrix against the raw data, and fill 0 for null values and initialize the user factor matrix and the item (movie) factor matrix, and the number of potential features k. Then we determine the loss function and the hyperparameter learning rate and the penalty factor and use alternate least squares optimization(ALS) to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix. The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix. Otherwise, we also use SVD to make matrix decomposition and get the good accuracy of the recommended system.

I. INTRODUCTION

A. Motivation

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

B. Dataset

1.Utilizing MovieLens-100k dataset.
2.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 1
THE EXAMPLE DATA IN u.data

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

3. u1.base / u1.test are train set and validation set respectively, seperated from dataset u.data with proportion of 80% and 20%. It also make sense to train set and validation set from u1.base / u1.test to u5.base / u5.test.

C. Experiment Step

The experiment code and drawing are both completed on jupyter.

Using alternate least squares optimization(ALS):
1.Read the data set and divide it (or use u1.base / u1.test to u5.base / u5.test directly). Populate the original scoring matrix R_{n_users,n_items} against the raw data, and fill 0 for null values.
2.Initialize the user factor matrix $P_{n\_users,K}$ and the item (movie) factor matrix $Q_{n\_item,K}$, where K is the number of potential features.
3.Determine the loss function and the hyperparameter learning rate η and the penalty factor λ.
4.Use alternate least squares optimization method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:
4.1 With fixd item factor matrix, find the loss partial derivative of each row (column) of the user factor matrices, ask the partial derivative to be zero and update the user factor matrices.
4.2 With fixd user factor matrix, find the loss partial derivative of each row (column) of the item factor matrices, ask the partial derivative to be zero and update the item

4.3 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, get a satisfactory user factor matrix P and an item factor matrix Q, Draw a $L_{validation}$ curve with varying iterations.
6.The final score prediction matrix $\hat{R}_{n\_users,n\_items}$ is obtained by multiplying the user factor matrix $P_{n\_users,K}$ and the transpose of the item factor matrix $Q_{n\_item,K}$.

II. METHODS AND THEORY

A. *Matrix Factorization*

Just as its name suggests, matrix factorization is to, obviously, factorize a matrix, i.e. to find out two (or more) matrices such that when you multiply them you will get back the original matrix. Matrix factorization goal is to make users - items evaluation matrix R into factor matrix and the item take the form of factor matrix, namely R = UV, where R is the n * m, and U is n x k, V is k * m. The intuitive representation is as follows:

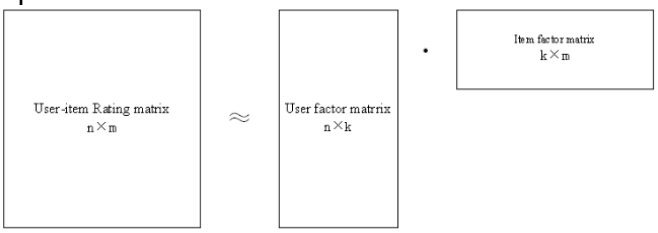

Fig.1. The intuitive representation of the matrix factorization

B. *ALS algorithm*

The abbreviation of ALS is alternating further squares, means alternating least squares; While ALS - WR is *alternating - further - squares with weighted - lambda - the abbreviation of regularization*, meaning the weighted regularized least squares method alternately. This method is often used in the recommendation system based on matrix factorization. For example: the user score for item matrix is decomposed into two matrices: one is the user preferences for items implicit characteristic matrix, the other one is items contains implicit characteristics of the matrix. In the process of the matrix factorization, lack of grading has been filled, so we can be based on the ratings to fill the most products recommended to the user.

ALS algorithm

- Loss function: $L_{u,i} = (r_{u,i} - \sum_{k=1}^{K} p_{u,k} q_{k,i})^2 + \lambda(\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2)$
- Gradient: $\begin{cases} \frac{\partial L_{u,i}}{\partial p_{u,k}} = -2(r_{u,i} - \sum_{k=1}^{K} p_{u,k} q_{k,i}) \cdot q_{k,i} + 2\lambda \cdot p_{u,k} \\ \frac{\partial L_{u,i}}{\partial q_{k,i}} = -2(r_{u,i} - \sum_{k=1}^{K} p_{u,k} q_{k,i}) \cdot p_{u,k} + 2\lambda \cdot q_{k,i} \end{cases}$
- Update: $\begin{cases} p_{u,k} \leftarrow p_{u,k} + \eta \cdot \frac{\partial L_{u,i}}{\partial p_{u,k}} \\ p_{k,i} \leftarrow q_{k,i} + \eta \cdot \frac{\partial L_{u,i}}{\partial q_{k,i}} \end{cases}$

Fig.2.ALS algorithm

III. EXPERIMENT

A. *Super parameter selection:*

k = 150,
penalty_factor = 0.01,
learning_rate = 0.0001,
epochs = 50

B. *Prediction results (best results):*

3) *The loss varying iterations:*

```
epoch 0
Loss of validation set: 113.902283355
--------------------
epoch 1
Loss of validation set: 35.0585044075
--------------------
epoch 2
Loss of validation set: 14.5872359815
--------------------
epoch 3
Loss of validation set: 7.28823520223
--------------------
epoch 4
Loss of validation set: 4.19330532673
--------------------
epoch 5
Loss of validation set: 2.7158200409
--------------------
epoch 6
Loss of validation set: 1.94297534958
--------------------
epoch 7
Loss of validation set: 1.50648813043
--------------------
epoch 8
Loss of validation set: 1.24250901716
--------------------
epoch 9
Loss of validation set: 1.07243335529
--------------------
epoch 10
Loss of validation set: 0.956201549683
--------------------
```

Fig.3. The loss with varying iterations

4) Loss curve with varying iterations:

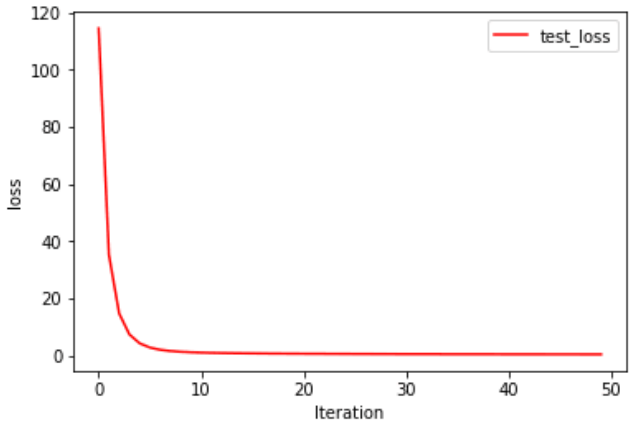

Fig.4. Loss curve with varying iterations in u1.test

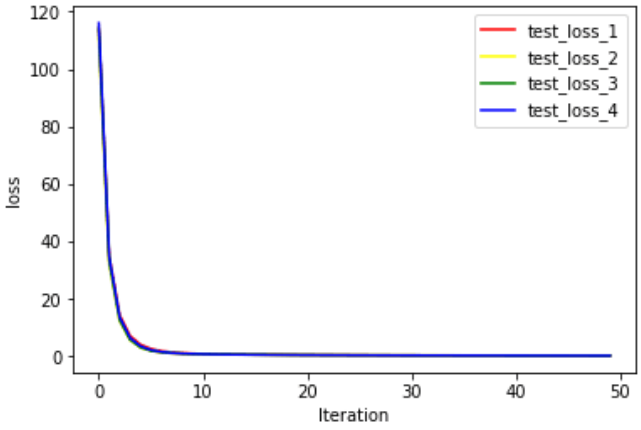

Fig.5. Loss curve with varying iterations in u1.test/u2.test/u3.test/u4.test

From the Loss curve we know, the loss in all the test sets is similar.

C. SVD

We also calculate k for the matrix factorization using by another method named SVD. Using the SVD we can get the result as follow:

1.The best k is 145.
2.The loss of the test data is as follow:

```
Loss of validation set 'u1.test': 0.194532227522
Loss of validation set 'u2.test': 0.194543605166
Loss of validation set 'u3.test': 0.192372269294
Loss of validation set 'u4.test': 0.192136987786
```

Fig.6. The loss of validation sets

IV. CONCLUSION

The results of the experiment show us the advantages of ALS that:
1. The algorithm can be more convenient parallelization:

In the process of using alternate least squares algorithm, when fixed user vector P [u], the algorithm can calculate the value of each dimension of the item vector Q [i] independently. In the same way, when fixed item vector Q [i], the algorithm can calculate the value of each dimension of the user vector P[u] independently. The method of independent calculation of each dimension of this vector provides the possibility for the large-scale parallelization of the algorithm design.
2. It is more advantageous for implicit data calculation

After inserting a large amount of implicit data, the original matrix of the training set is no longer sparse. In the case of sparse matrix, such as this experimental data, using ALS as the optimal solution will be the best choice.

Besides, with the data of this experiment, the result show that using the SVD is better than using the ALS algorithm, because the loss of SVD is 0.19 but the loss of ALS is 0.28 with 500 iterations.

V. REFERENCES

http://blog.csdn.net/sdujava2011/article/details/51387393