
# The Experiment Report of Machine Learning

**SCHOOL:** SCHOOL OF SOFTWARE ENGINEERING

**SUBJECT:** SOFTWARE ENGINEERING

Author:
Qiang Yuan, Yihui Zhu, Junpeng Su

Supervisor:
Mingkui Tan

Student ID：201530613511,
201530613955, 201530612743

Grade:
Undergraduate

December 25, 2017

# Recommender System Based on Matrix Decomposition

**Abstract— It is an experiment of machine learning about Recommender System Based on Matrix Decomposition. In this paper, we propose that use alternate least squares optimization or stochastic gradient descent method to complete the experiment.**

## I. INTRODUCTION

In the experiment, we try to use matrix decomposition to construct a recommender system under small-scale dataset, and we choose to utilize both alternate least squares optimization and stochastic gradient descent method to complete the experiment, adjust parameters to get the right results and smoother loss curve.

The experiment is to help us to explore the construction of recommended system, understand the principle of matrix decomposition, be familiar to the use of gradient descent, and construct a recommended system under small-scale dataset so that we cultivate engineering ability, combine the theory with the actual project and experience the complete process of machine learning.

We use python3 as the environment of experiment, including python package: sklearn, numpy, jupyter, matplotlib.

The results have loss curves of validation loss and the final score prediction matrix. We wish to see that these methods have higher accuracy, rapider convergence, smoother loss curve, and better learning and diagnostic properties to complex data.

## II. METHODS AND THEORY

We are going to use dataset to populate the original scoring matrix, then use method to decompose the sparse user score matrix, get the user factor matrix and item factor matrix, draw the loss curve and calculate the final score prediction matrix. Then we will talk about those methods and theory.

To begin with, Recommender System applies statistical and knowledge discovery techniques to the problem of making product recommendations. 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. So matrix decomposition becomes the most widely used algorithm.

Matrix factorization is to factorize a matrix, i.e. to find out two (or more) factor matrices such that when you multiply them you will get back the original matrix. Let the original scoring matrix R of size $|U| \times |D|$, and find two factor matrices P (a $|U| \times K$ matrix) and Q(a $|D| \times K$ matrix) such that their product approximates R:

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

Square error is usually selected as the loss function of the model.

ALS for MF is to minimize the following objective function:

$$\text{error}(y_{\text{truth}}, y_{\text{pred}}) = \sum_{i,j} \frac{1}{2}\left(y_{\text{truth}_{i,j}} - y_{\text{pred}_{i,j}}\right)^2 + \frac{\lambda}{2}\left(\sum_i n_{p_i}||P||^2 + \sum_j n_{q_j}||Q||^2\right)$$

$\lambda$ is regularization parameter to avoid overfitting.

$n_{p_i}$ and $n_{q_j}$ denote the number of total ratings on user i and item j, respectively.

Optimize P while fixing Q:

$$P_i = \left(Q_j Q_j^T + \lambda n_{P_i} I\right)^{-1} \cdot Q^T \cdot R_{i*}^T$$

$R_{i*}$ denotes the i-th row of rating matrix R.

Update all $P_i$ with the above formula.

Optimize Q while fixing P:

$$Q_i = \left(P_i P_i^T + \lambda n_{Q_j} I\right)^{-1} \cdot P^T \cdot R_{*j}^T$$

$R_{*j}$ denotes the j-th column of rating matrix R.

Update all $Q_j$ with the above formula.

SGD for MF is scalable to large-scale datasets.

The formula is $\text{error}(y_{\text{truth}}, y_{\text{pred}}) = \sum \frac{1}{2}\left(y_{\text{truth}_{i,j}} - y_{\text{pred}_{i,j}}\right)^2 + \frac{\lambda}{2}\sum_{l=1}^{K}(||P||^2 + ||Q||^2)$

We introduce regularization to prevent over-fitting.

$$y\_pred_{i,j} = P_i Q_j^T$$

The derivative is

$$\frac{\partial \text{error}_{i,j}}{\partial P_i} = -\left(y_{\text{truth}_{i,j}} - y_{\text{pred}_{i,j}}\right) Q_j + \lambda P_i$$

$$\frac{\partial \text{error}_{i,j}}{\partial Q_j} = -\left(y_{\text{truth}_{i,j}} - y_{\text{pred}_{i,j}}\right) P_i + \lambda Q_j$$

Update the feature matrices P and Q with learning rate $\alpha$:

$$P_i = P_i + \alpha\left(\left(y_{truth_{i,j}} - y_{pred_{i,j}}\right) Q_j - \lambda P_i\right)$$

$$Q_i = Q_i + \alpha\left(\left(y_{truth_{i,j}} - y_{pred_{i,j}}\right) P_i - \lambda Q_j\right)$$

Only part of the rate matrix which included in the train data is chosen to be the sample of stochastic gradient descent.

Donate the speed of loss reduction when the gradient of item factor matrix is calculated before the update of user factor matrix as s1. Donate the speed of loss reduction when the gradient of item factor matrix is calculated after the update of user factor matrix as s2. The experiment result shows that s2 is larger than s1.

## III. EXPERIMENT

In this section, we analyze the method of matrix decomposition for recommendations, and we also investigate how to update factor matrices to improve the performance of algorithm.

**3.1 Data sets and data analysis:**

This experiment utilizes MovieLens-100k dataset which named u.data, and it consists of 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. Its format is as shown below.

| 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 |

u1.base / u1.test are train set and validation set respectively, separated 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.

**3.2 Experimental steps:**

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 hyper-parameter learning rate η and the penalty factor $\lambda$.
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 fixed 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 fixed 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 $Q_{n_{item},K}$.

*Using stochastic gradient descent method (SGD):*

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 hyper-parameter learning rate η and the penalty factor $\lambda$.
4. Use the stochastic gradient descent method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:
   4.1 Select a sample from scoring matrix randomly;
   4.2 Calculate this sample's loss gradient of specific row(column) of user factor matrix and item factor matrix;
   4.3 Use SGD to update the specific row(column) of $P_{n_{users},K}$ and $Q_{n_{item},K}$;
   4.4 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 $Q_{n_{item},K}$.

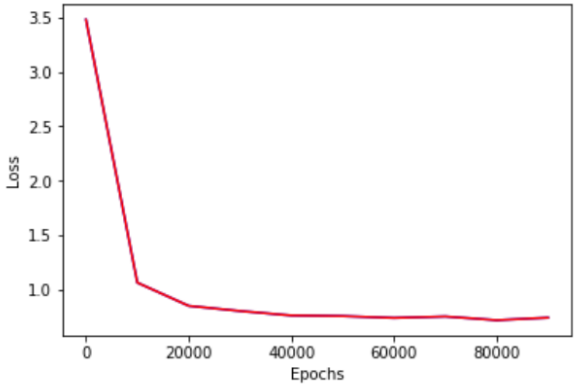
**(We choose to use both alternate least squares optimization and stochastic gradient descent method to complete the experiment. )**

Sort out the experimental results and complete the experimental report.

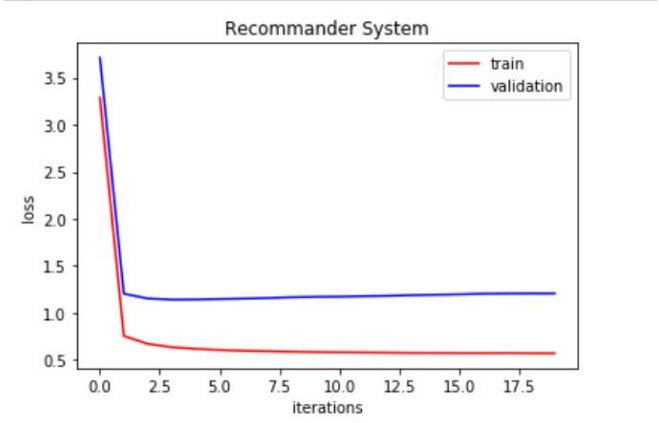
**3.3 Experimental results and curve:**

The below are our experimental results and the accuracy curves of models.

The validation loss in SGD is as below:




The validation and training loss in ALS is as below:




## IV. CONCLUSION

In this experiment, we use a sparse feature subset and matrix decomposition to construct a recommender system, and we utilize both ALS and SGD to accomplish the experiment. There are some gains in the middle of our tests. Primarily, we learn the theory of matrix decomposition and the construction of recommendation system, then we practice the method in the experiment, we are beginning to have some insight into the method. Next, we get familiar with the use of gradient descent, and try the ALS optimization so that we unite the knowledge and the actual project, experience the process of machine learning. Furthermore, we see the difference between SGD and ALS, ALS is easier to parallelize than SGD, and it's converges faster. However, SGD has less storage complexity and less computational complexity than ALS.