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:
ChangxiZhu|YuzhongYe| YangyangZhao

Supervisor:
Qingyao Wu

Student ID：
201720144894|201721045763|201710106598

Grade:
Graduate

December 21, 2017

RECOMMENDER SYSTEM BASED ON MATRIX DECOMPOSITION

Abstract—The paper describes a recommender system trained by matrix decomposition algorithm. Recommender system is used to solve rating problems (i.e. user's attitude to some objects). These problems commonly have large-scale matrix though their data may be sparse. Matrix decomposition can performs well in these problems by divide a large matrix into some small matrix and deal with different algorithms.

I. INTRODUCTION

In order for us to further understand the principle of matrix decomposition and be familiar to the use of gradient descent, we experience the project "recommender system based on matrix decomposition" to construct a recommendation system under small-scale dataset, cultivating our engineering ability. In this paper, we implement a simple but effective approach named matrix decomposition. Furthermore, we utilize the Movielens-100K dataset , which consisting 10,000 comments from 943 users out of 1682 movies, stored in data/ml-100k/u.data, data was spilt into two files named u.base and u.test for training and testing respectively. We experience the project in PyCharm Community, which has built-in python package above.

Matrix decomposition:

Matrix decomposition is a factorization of a matrix into a product of matrices. There are many different matrix decompositions. In this problem, we have users matrix U and movies matrix V and we want to learn R=UV are the rating value for each user and each movie. So this decomposition problem also called UV Factorization.

ALS (alternate least squares):

Because we need to find the optimal solution with two sets of variables U and V, we can fixed one of them and learn another in alternately. It can be proved that this method can let both two matrix UV converged.

II. METHODS AND THEORY

According to the methods provided by teacher Wu (Steps 1-6 below) and we do it.

1. Read the dataset and divide it. Because the data has been divided we can read the training data and testing data directly:

def load_data(dir_name):

Param	Dir_name	the file or directory name entered
Return	return an training data and the testing data	

2. Populate the original scoring matrix against the raw data, and fill 0 for null values:

def padding_data(x ,y ,value):

Param	x	List of user_id
	y	List of item_id
	value	List of rating value, if not exist then set it as

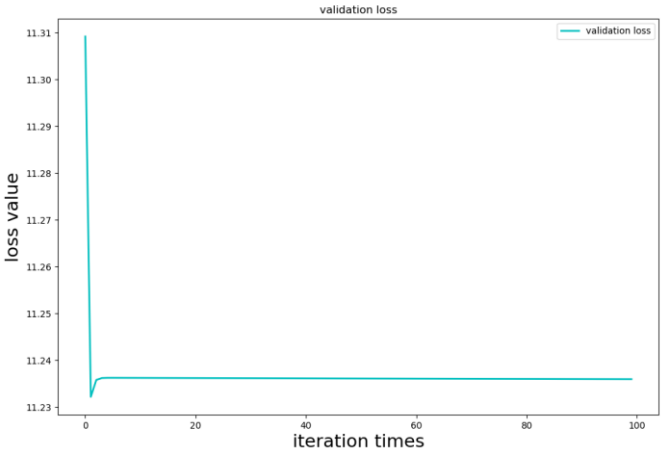
		a default value.(i.e. zero)
Return	return padded data x and y	

3. Determine the loss function and the hyper-parameter learning rate and the penalty factor. In this experiment we choose mean square loss and set penalty factor as 0.1. In task one (ALS) algorithm we don't need a learning rate but in task two (Gradient descent) we set it equals to 0.01.
4. Initialize the user factor matrix and the item (movie) factor matrix, which are (n-user)*k and (n-movie)*k matrix. Here k is the number of potential features. We set k equal to different value and show their performance.
5. Decompose the sparse user score matrix by ALS or GD algorithm. For ALS, we:
   a) 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.
   b) 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
   c) Calculate the Loss on the validation set, comparing with the Loss of the previous iteration to determine if it has converged.

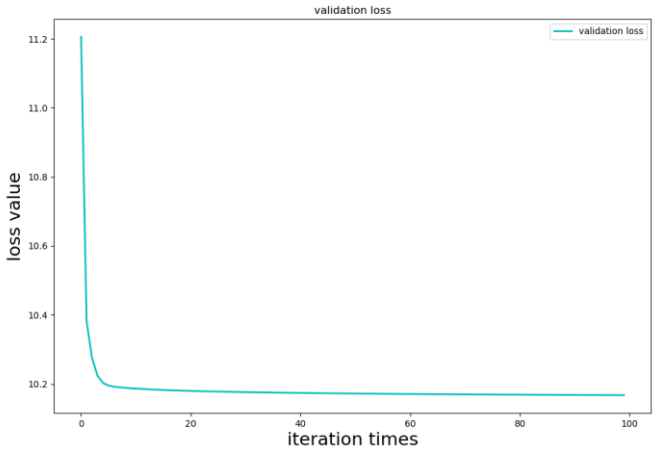
   And for GD:
   d) Select a sample from scoring matrix randomly
   e) Calculate this sample's loss gradient of specific row(column) of user factor matrix and item factor matrix
   f) Use SGD to update the specific row(column) of user factor matrix and movie factor matrix
6. Repeat step 5 in several times, get a satisfactory user factor matrix and an item factor matrix, Draw a Loss curve with varying iterations.

III. Experiment

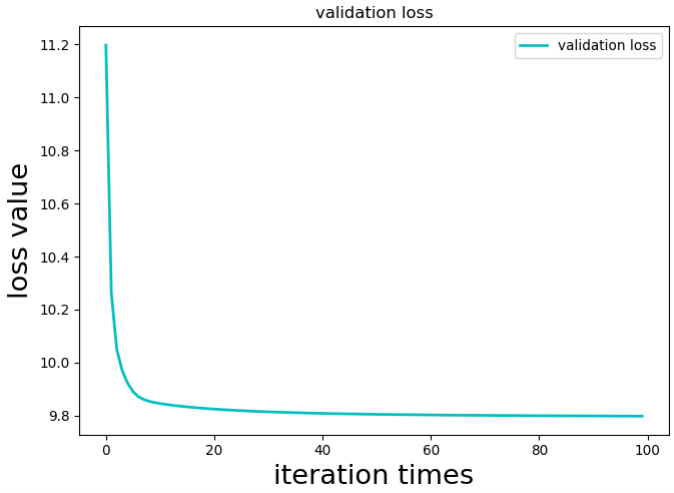
We did the experiment with different k value, and listed the ALS results of 1, 5, 10 below.



Validation L2-loss with k=1, ALS

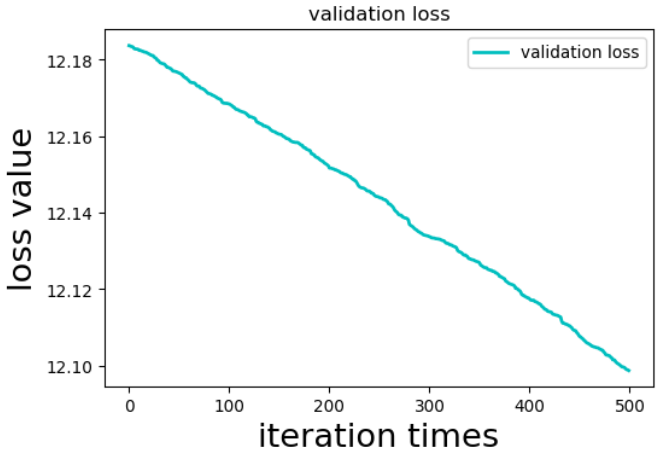


Validation L2-loss with k=5, ALS

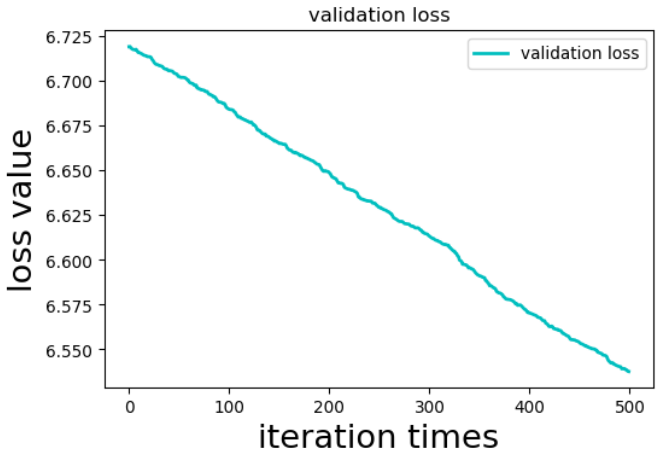


Validation L2-loss with k=10, ALS

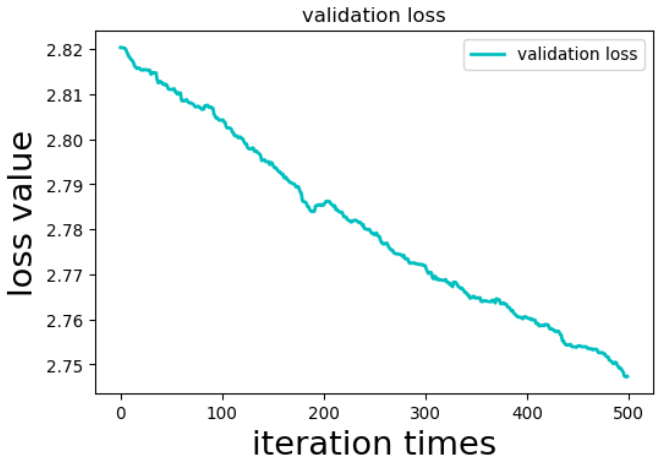
And with different k value, the SGD results of 1, 5, 10 below.



Validation L2-loss with k=1, SGD



Validation L2-loss with k=5, SGD



Validation L2-loss with k=10, SGD

IV. CONCLUSION

As we can see the pictures above, the potential value k obviously affect final result. When k is equal to one, the curve decline suddenly at the beginning, but then rise and keeps the loss upper than 10. When k is larger we need more iteration to convergence, while it can get better result and makes loss lower than 10.GD based algorithm performs similar to ALS based algorithm but may suffer from oscillation (Because of randomness of SGD) and need more iteration to achieve the best result. In a conclusion, because of the matrix decomposition's effectiveness in recommender system, we should do more research in it and cultivate our engineering ability.