South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Yuhong Zhang and ZhaoTao Lin

Supervisor:
Qingyao Wu

Student ID：
201530612309 and 201530613665

Grade:
Undergraduate

January 6, 2018

Recommender System Based on Matrix Decomposition

Abstract—

I. Introduction

Motivation

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

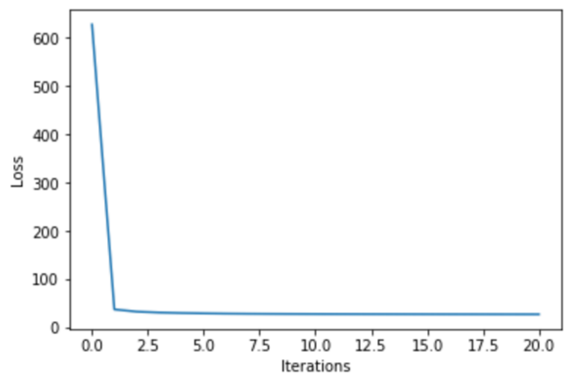
Using stochastic gradient descent method(SGD):

Read the data set and divide it (or use u1.base / u1.test to u5.base / u5.test directly). Populate the original scoring matrix against the raw data, and fill 0 for null values.
Initialize the user factor matrix and the item (movie) factor matrix , where K is the number of potential features.
Determine the loss function and hyperparameter learning rate and the penalty factor .
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 and ;
4.4 Calculate the loss of the validation on the validation set, comparing with the loss of the validation of the previous iteration to determine if it has converged.
Repeat step 4. several times, get a satisfactory user factor matrix and an item factor matrix , Draw a curve with varying iterations.
The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix .

III. Experiment

```
import numpy as np
from scipy.sparse import csc_matrix
import matplotlib.pyplot as plt
from numpy import random
import warnings
warnings.filterwarnings("ignore")
def load(path):
    with open(path) as f0:
        user = []
        item = []
        rate = []
        for i in f0:
            temp = i.split()
            user.append(temp[0])
            item.append(temp[1])
            rate.append(temp[2])
    return
np.array(user).astype(float),np.array(item).astype(float),np.array(rate).astype(float)

def rate_csc_matrix():
    path_base = "./u1.base"
    path_test = "./u1.test"
    user_base,item_base,rate_base = load(path_base)
    user_test,item_test,rate_test = load(path_test)
    rate_matrix_base = csc_matrix((rate_base, (user_base, item_base)), shape=(944, 1683)).toarray()
    rate_matrix_test = csc_matrix((rate_test, (user_test, item_test)), shape=(944, 1683)).toarray()
    u1_base = np.delete(rate_matrix_base,0,axis=0)
    u1_base = np.delete(u1_base,0,axis=1)
    u1_test = np.delete(rate_matrix_test,0,axis=0)
    u1_test = np.delete(u1_test,0,axis=1)
    return u1_base,u1_test

def loss(r, p, q, beta):
    L0 = np.sum((r - np.dot(p, q))**2)
    L1 = beta * (np.sum(p**2) + np.sum(q**2))
    loss_ = (L0 + L1) / (r.shape[0] * r.shape[1])
    return loss_

def grad(p, q, K, m, n, error, alpha, beta):
    for k in range(K):
        p[m, k] = p[m, k] + (2 * error * q[k, n] - beta * p[m, k]) * alpha
        q[k, n] = q[k, n] + (2 * error * p[m, k] - beta * q[k, n]) * alpha
    return p,q

def Recommend(base, test, K):
    alpha = 0.005
    beta = 0.02
```

```
    iteration = 30
    p = np.random.rand(base.shape[0], K)
    q = np.random.rand(K, base.shape[1])
    loss_record = []
    loss_ = loss(test, p, q, beta)
    loss_record.append(loss_)
    for i in range(iteration):
        for m in range (base.shape[0]):
            for n in range (base.shape[1]):
                if base[m][n] > 0:
                    error = base[m][n] - np.dot(p[m,:], q[:,n])
                    p,q = grad(p, q, K, m, n, error, alpha, beta);
        loss_ = loss(test, p, q, beta)
        loss_record.append(loss_)
        print(loss_)
    return p, q, loss_record

if __name__ == "__main__":
    u1_base,u1_test = rate_csc_matrix();
    K = 100
    p, q, loss_record = Recommend(u1_base, u1_test, K)
    plt.xlabel('Iterations')
    plt.ylabel('Loss')
    plt.plot(loss_record)
    plt.show()
    r = p*q
    print(r)
```



IV. CONCLUSION

In the machine learning experiment, we learned to realize recommendation system on simple small data set, is difficult at the beginning, after the check data and their research, gradually with the idea. I hope our machine learning ability will be stronger and stronger.