South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
袁澍民 and 刘志杰

Supervisor:
Qingyao Wu

Student ID：
201530613528 and201530612392

Grade:
Undergraduate

January 5, 2018

Matrix-based recommendation system

Abstract—

I. Introduction

Today,I try to solve a question which named Matrix-based recommendation system, There are the terminals of this the question:
1.Explore the construction of recommended system
2.Understand the principle of matrix decomposition
3.Proficiency in the use of gradient decline
4.Implement a recommendation system on a simple, 5.small-scale data set to develop engineering capabilities

Data set description
1.Use MovieLens-100k data set
2.u.data - Consists of 10,000 users rated 16,000 movies by 943 users. Each user scored at least 20 movies. Users and movies numbered consecutively from number one. The data is sorted randomly.
3.Data sets u1.base / u1.test to u5.base / u5.test are training sets and test sets that divide the u.data data set into 80% / 20% proportions.
4. According to their own assessment methods can also be divided training set and validation set

II. Methods and Theory

Optimization using stochastic gradient descent method

1.Read the data and divide the data set (or use u1.base / u1.test to u5.base / u5.test). Fill the original scoring matrix with the raw data, zeroes for null values.
2.Initialize the user factor matrix and item (movie) factor matrix, where is the number of potential features.
3.Determine the loss function and determine the super-parameter learning rate and penalty coefficient.
4.The random user score matrix is decomposed by stochastic gradient descent method to get user factor matrix and item (movie) factor matrix:
4.1 randomly select a sample in the user rating matrix;
4.2 Calculate the sample loss function value of the user factor matrix of a row (column) and article factor matrix of a row (column) gradient;
4.3 Gradient descent Update one row (column) and one row (column);
4.4 Calculated on the validation set, compared with the previous iteration to determine whether the convergence.
5.Repeat step 4. A number of times, get a satisfactory user factor matrix and item factor matrix, draw the curve with changes in the number of iterations.
6.Multiplying the user factor matrix with the transpose of the item factor matrix yields the final score prediction matrix.

III. Experiment

Using loss function :

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

After regularization:

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

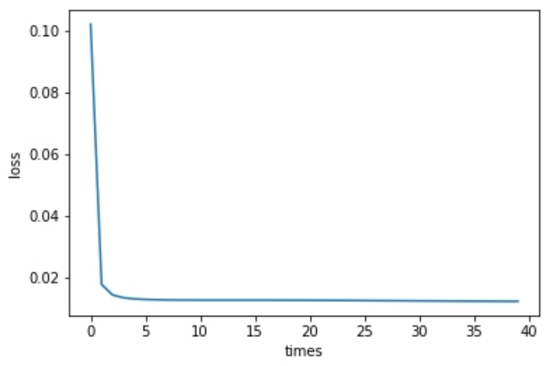
Optimization algorithm based on gradient descent:

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

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

```
import numpy as np
import pickle
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_svmlight_file
import time
import matplotlib.pyplot as plt

def r_file( file_name):
    file = open( file_name)
    data = file.read()
    file.close()
    data = np.array( data.split( ))
    data = data.reshape( [int(len(data)/4) , 4] )
    return data

def calculate( matrix , data):
    i=0
    j=0
    for x in range(data.shape[0]):
        i=int(data[x][0])-1
        j=int(data[x][1])-1
        matrix[i][j] = data[x][2]
    return matrix

def makeMatrix():
    r_b = np.zeros(943*1682).reshape(943,1682)
    r_t = np.zeros(943*1682).reshape(943,1682)
    #这里只用了 u1
    data1 = r_file('ml-100k/u1.base')
    data2 = r_file('ml-100k/u1.test')
    calculate(r_b,data1)
    calculate(r_t,data2)
```

```
        return r_b, r_t

    def matrix_factorization(R_b, R_t, P, Q, K, steps=400, alpha=0.0005, beta=0.02):
        Q = Q.T
        L_list = []
        for step in range(steps):
            for i in range(len(R_b)):
                for j in range(len(R_b[i])):
                    if R_b[i][j] > 0:
                        e = R_b[i][j] - np.dot(P[i,:],Q[:,j])
                        for k in range(K):
                            P[i][k] = P[i][k] + (2 * e * Q[k][j] - beta * P[i][k])*alpha
                            Q[k][j] = Q[k][j] + (2 * e* P[i][k] - beta * Q[k][j])*alpha
            if step%10 == 0:
                l = 0
                for i in range(len(R_t)):
                    for j in range(len(R_t[i])):
                        if R_t[i][j] > 0:
                            l = l + pow(R_t[i][j] - np.dot(P[i,:],Q[:,j]), 2)
                            for k in range(K):
                                l = l + (beta/2) * (pow(P[i][k],2) + pow(Q[k][j],2))#正则化
                l = l/(R_b.shape[0]*R_b.shape[1])
                L_list.append(l)
                print(l)
                if l < 0.001:
                    break
        return P, Q.T, L_list

    r_b , r_t = makeMatrix()
    N = len(r_b)
    M = len(r_b[0])
    K = 2
    P = np.random.rand(N,K)
    Q = np.random.rand(M,K)
    P1, Q1, L_v = matrix_factorization(r_b, r_t, P, Q, K)

    plt.xlabel('times')
    plt.ylabel('loss')
    time_list=[i for i in range(40)]
    line1=plt.plot(time_list,np.array(L_v))
    plt.show()
    error = 0
    R1 = np.dot(P1, Q1.T)
    print(R1)
```



Reference formula source link:
https://www.cnblogs.com/kobedeshow/p/3651833.html

IV. CONCLUSION

Through this experiment, I tried to optimize the matrix-based recommender system by stochastic gradient descent algorithm and compare the differences between ALS and SGD from the learning, and spent more time on understanding these formulas. Thank you for another partner Give me an idea of the formula, and promote us to experiment together.