The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Chaoxian Zhang and Yuchen Zhou and Wende Zhu

Supervisor:
Mingkui Tan

Student ID：201530613597 and 201530613887 and 201530613948

Grade:
Undergraduate

December 21, 2017

Recommender System Based on Matrix Decomposition

***Abstract*—In today's big data age, a huge amount of information makes it difficult for users to get the information they want. And recommending systems help them to get what they want efficiently.Taking the user - film score matrix as an example, the matrix decomposition is to predict the missing values in the scoring matrix and then recommend it to the user in some way according to the predicted value.**

I. Introduction

In this experiment, the algorithm based on matrix decomposition assumes that the user's evaluation of the film is influenced by multiple recessive factors. The original high latitude score matrix is decomposed into two low latitude matrix products. The two low latitude matrices can be regarded as user characteristic matrix and item characteristic matrix. Then, through training we can make it adopts to approximate the original scoring matrix and fill in the blanks.

II. Methods and Theory

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 hyperparameter learning rate η and the penalty factor λ.

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.

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 $P_{n\_users,K}$ is obtained by multiplying the user factor matrix $P_{n\_users,K}$ and the transpose of the item factor matrix $Q_{n\_item,K}$.

III. Experiment

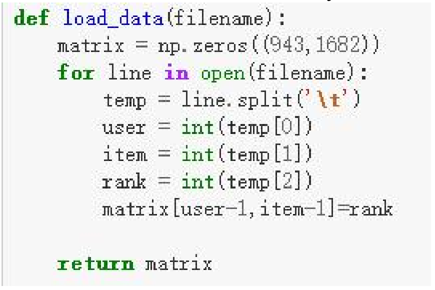
A. Dataset

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

B. Implementat

Firstly, we ues load_data() function to load data(the function is shown in Fig. 1.),. In this experiments, we use u1.base / u1.test to u5.base / u5.test directly.



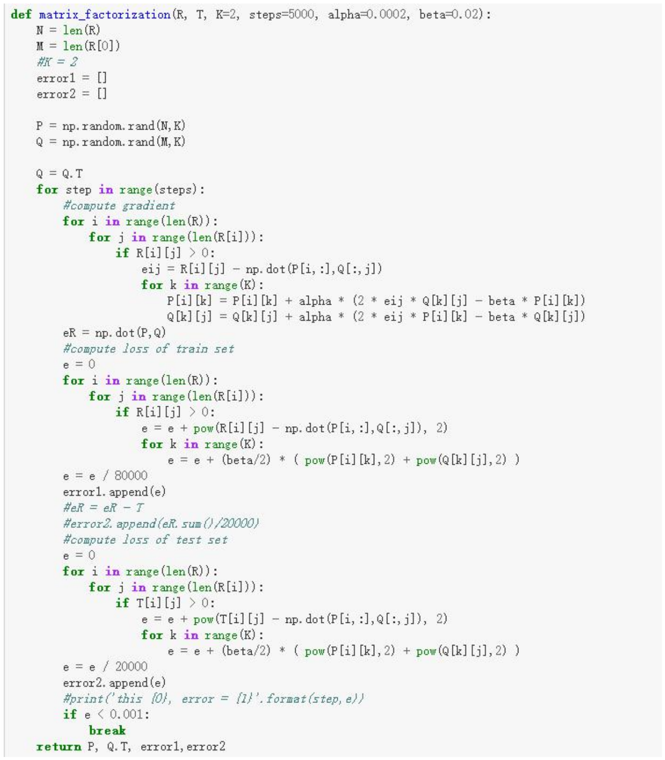
```
def load_data(filename):
    matrix = np.zeros((943,1682))
    for line in open(filename):
        temp = line.split('\t')
        user = int(temp[0])
        item = int(temp[1])
        rank = int(temp[2])
        matrix[user-1,item-1]=rank

    return matrix
```

Fig. 1. Load datas.

We set the learning rate as 0.0002(alpha), and set the regularization parameter penalty factor λ as 0.02(beta). The loss function is :

$$f=\frac{1}{2}\sum_{u=1}^{m}\sum_{j\in P_u}\left(\hat{\boldsymbol{R}}_{uj}-\boldsymbol{R}_{uj}\right)^2+\frac{\lambda}{2}\left(\|\boldsymbol{P}\|_F^2+\|\boldsymbol{Q}\|_F^2\right)$$

Then, we use matrix_factorization() function to train it. We can see the function in Fig. 2..



```python
def matrix_factorization(R, T, K=2, steps=5000, alpha=0.0002, beta=0.02):
    N = len(R)
    M = len(R[0])
    #K = 2
    error1 = []
    error2 = []

    P = np.random.rand(N,K)
    Q = np.random.rand(M,K)

    Q = Q.T
    for step in range(steps):
        #compute gradient
        for i in range(len(R)):
            for j in range(len(R[i])):
                if R[i][j] > 0:
                    eij = R[i][j] - np.dot(P[i,:],Q[:,j])
                    for k in range(K):
                        P[i][k] = P[i][k] + alpha * (2 * eij * Q[k][j] - beta * P[i][k])
                        Q[k][j] = Q[k][j] + alpha * (2 * eij * P[i][k] - beta * Q[k][j])
        eR = np.dot(P,Q)
        #compute loss of train set
        e = 0
        for i in range(len(R)):
            for j in range(len(R[i])):
                if R[i][j] > 0:
                    e = e + pow(R[i][j] - np.dot(P[i,:],Q[:,j]), 2)
                    for k in range(K):
                        e = e + (beta/2) * ( pow(P[i][k],2) + pow(Q[k][j],2) )
        e = e / 80000
        error1.append(e)
        #eR = eR - T
        #error2.append(eR.sum()/20000)
        #compute loss of test set
        e = 0
        for i in range(len(R)):
            for j in range(len(R[i])):
                if T[i][j] > 0:
                    e = e + pow(T[i][j] - np.dot(P[i,:],Q[:,j]), 2)
                    for k in range(K):
                        e = e + (beta/2) * ( pow(P[i][k],2) + pow(Q[k][j],2) )
        e = e / 20000
        error2.append(e)
        #print('this {0}, error = {1}'.format(step,e))
        if e < 0.001:
            break
    return P, Q.T, error1,error2
```

Fig. 2. Train and get the The final score prediction matrix

The last, we train u1.base and test u1.test, the loss is shown in the follwing picture Fig. 3..

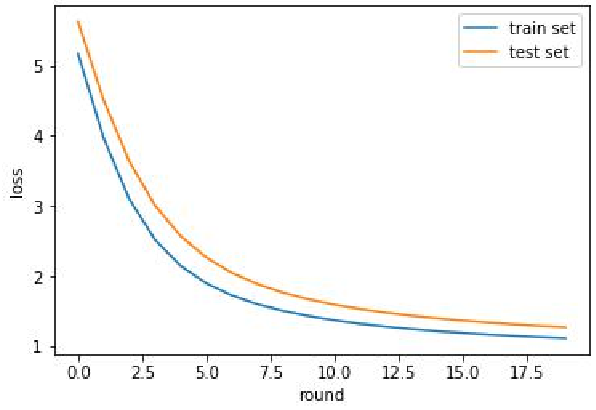


Fig. 3. The loss update with the round times

IV. CONCLUSION

In this experiment, we finished it with the SGD method. When the training is about 18 times, the change of the loss value is less than 0.001. And now, we get a simple recommender system. This system only takes into account the feature of the movie itself and the user preferences,if we consider the relationship between the users and the social situation, then the system will be more perfect. Up to now, we have finished 4 projects.I have to say that machine learning is an interesting subject ! It is exciting to see that the loss value decline,though it is hard, it is funny.