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:
Xiaoyu Luo,Weimin Luo and Dongdong Zhou

Supervisor:
Qingyao Wu

Student ID：
201530612507,201530613061 and 201530613849

Grade:
Undergraduate

December 29, 2017

Recommender System Based on Matrix Decomposition

I. Abstract

Now Recommender System is very popular and plays a major role in browsing information. And given that we have been learned the related theory, In this experiment we are about to construct a Recommendation System by using the principle of Matrix Decomposition under small-scale dataset, which will be optimized by SGD, and combine the theory with the actual project.

II. INTRODUCTION

With the continuous expansion of the scale of e-commerce, the number and types of products are rapidly increasing, and customers need to spend a lot of time to find the products they want to buy. This browsing a large number of unrelated information and product process will undoubtedly make the drowning in the information overload problem of consumers continue to drain.

In order to solve these problems, personalized recommendation system came into being. Personalized recommendation system is based on massive data mining based on a high-level business intelligence platform to help e-commerce Web site for its customers to provide fully personalized shopping decision support and information services. In this experiment we are about to use Matrix Decomposition to construct a Recommendation System.

Matrix Decomposition decomposition of a matrix into two or more matrices. For the above user-product matrix (score matrix), it is denoted by Rm × n. We can decompose it into the product of two or more matrices. Suppose decomposed into two matrices Pm × k and Qk × n, we want to make the product of matrices Pm × k and Qk × n restore the original matrices Rm × n:

$$R_{m\times n} \approx P_{m\times k} \times Q_{k\times n} = \hat{R}_{m\times n}$$

Among them, the matrix Pm × k represents the relationship between m users and k topics, while the matrix Qk × n represents the relationship between k topics and n products.

The square of the error between the original scoring matrix Rm × n and the reconstructed scoring matrix R ^ m × n may be used as the loss function, that is:

$$\mathcal{L} = (r_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i)^2 + \lambda_p ||\mathbf{p}_u||^2 + \lambda_q ||\mathbf{q}_i||^2$$

Finally, the minimum of the sum of all non-"-" losses needs to be solved.

For the above squared loss function, it can be solved by the gradient descent method. The core step of the gradient descent method is:

Randomly select an observed sample $r_{u,i}$.

Calculate the **prediction error**:

$$E_{u,i} = r_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i$$

Calculate the **gradient**:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{p}_u} = E_{u,i}(-\mathbf{q}_i) + \lambda_p \mathbf{p}_u$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}_i} = E_{u,i}(-\mathbf{p}_u) + \lambda_q \mathbf{q}_i$$

Update the feature matrices P and Q with learning rate α：

$$\mathbf{p}_u = \mathbf{p}_u + \alpha(E_{u,i}\mathbf{q}_i - \lambda_p \mathbf{p}_u)$$

$$\mathbf{q}_i = \mathbf{q}_i + \alpha(E_{u,i}\mathbf{p}_u - \lambda_q \mathbf{q}_i)$$

Using the above process, we can get the matrices Pm × k and Qk × n, so that the product j can be scored for user i:

$$\sum_{k=1}^{K} p_{i,k} q_{k,j}$$

This paper is organized as follows. In Section 3, we describe our proposed algorithm. Implementation issues are investigated in Section 4. In Section 5, we show the results,gains and inspirations.

III. METHODS AND THEORY

SGD is to minimize the following objective function:

$$\mathcal{L} = \sum_{u,i \in \Omega} (r_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i)^2 + \lambda_p ||\mathbf{p}_u||^2 + \lambda_q ||\mathbf{q}_i||^2$$

$$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m]^\top \in \mathbb{R}^{m \times k}$$

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n] \in \mathbb{R}^{k \times n}$$

$r_{u,i}$ denotes the actual rating of user u for item i.

Ω denotes the set of observed samples from rating matrix R

λ_p and λ_q are regularization parameters to avoid overfitting.

Algorithm 4 SGD Algorithm

1: **Require** feature matrices $\mathbf{P}$, $\mathbf{Q}$, observed set Ω, regularizat parameters λ_p, λ_q and learning rate α.
2: **Randomly** select an observed sample $r_{u,i}$ from observed set
3: Calculate the **gradient** w.r.t to the objective function:

$$E_{u,i} = r_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{p}_u} = E_{u,i}(-\mathbf{q}_i) + \lambda_p \mathbf{p}_u$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}_i} = E_{u,i}(-\mathbf{p}_u) + \lambda_q \mathbf{q}_i$$

4: **Update** the feature matrices $\mathbf{P}$ and $\mathbf{Q}$ with learning rate α a gradient:

$$\mathbf{p}_u = \mathbf{p}_u + \alpha(E_{u,i}\mathbf{q}_i - \lambda_p \mathbf{p}_u)$$
$$\mathbf{q}_i = \mathbf{q}_i + \alpha(E_{u,i}\mathbf{p}_u - \lambda_q \mathbf{q}_i)$$

5: **Repeat** the above processes until **convergence**.

IV. 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.

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	88397596

3. 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.

4. You can also construct train set and validation set according to your own evaluation method.

B. Implementation

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.

```
def get_data(filename):
    header = ['user_id', 'item_id', 'rating', 'timestamp']
    df = pd.read_csv(filename, sep = '\t', names = header)
    n_users = df.user_id.unique().shape[0]
    n_items = df.item_id.unique().shape[0]
    return df,n_users,n_items
```

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.

```
k = 2
p = np.random.rand(m,k)
q = np.random.rand(n,k)
steps=50
e = []
```

3.Determine the loss function and hyperparameter learning rate and the penalty factor λ .

```
def loss(R, P, Q, K, beta=0.02):
    Q = Q.T
    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) )

return e
```

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.

```
def matrix_factorization(R, P, Q, K, alpha=0.0002, beta=0.02):
    Q = Q.T
    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])

    return P, Q.T
```

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.

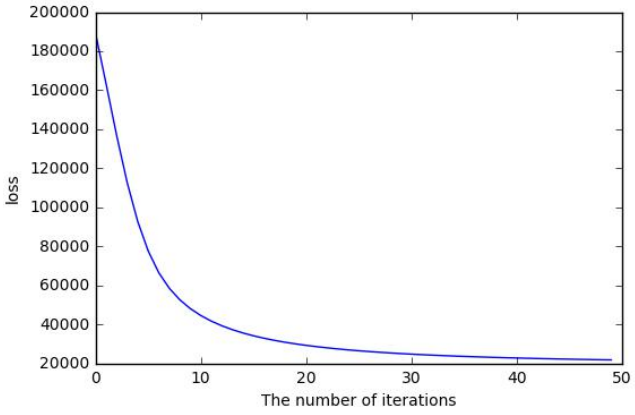
```
steps=50
e = []
for step in range(steps):
    nP, nQ = matrix_factorization(train_data_ndarray, p, q, k)
    e.append(loss(test_data_ndarray,nP,nQ,k))
    #if e[step] < 0.001:
```

6.The final score prediction matrix R_{n_users,n_item} is obtained by multiplying the user factor matrix $P_{n\_users,K}$ and the transpose of the item factor matrix $Q_{n\_item,K}$.

```
print(np.dot(nP,nQ.T))
```

V. CONCLUSION

The experimental results showed in the following figure:



The result is still satisfactory.

Through this experiment,we learn how to construct a recommendation system under small-scale dataset, cultivate engineering ability,which is significant for our future learn and work life.What's more,we also appreciate the charm of Recommendation System that can make the drowning in the information overload problem of consumers continue to drain.We will work harder to learn the science related to machine learning, strive to climb the heights of science and try hard for the progress of mankind.

After accomplishing the experiment,I learn that in the future research of Recommendation System, in addition to improving the algorithm, we should also consider the following three aspects of the Recommended System.

First, the stability of the algorithm itself.Although SGD algorithm is superior to other algorithms on average but its own experimental results are not more stable than other algorithms, which requires us to pay attention to the stability of the algorithm in the future theoretical research and practical applications

Second, improve the evaluation of the effectiveness of the recommendation system or recommended algorithm.

Third,Recommended System is not only used in the field of Internet, in the future often involve information filtering unknown sources, can be regarded as a recommendation.