The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

December 25, 2017

Author:
Yingjie Lin Yang Lin Dongcheng Mai
林英杰 林杨 麦栋铖

Supervisor:
Mingkui Tan

Student ID：
201530612286 and 201530612279 and 201536612525

Grade:
Undergraduate

Face Classification Based on AdaBoost Algorithm

Abstract—This experiment focus on building a Recommender System Based on Matrix Decomposition.

I. INTRODUCTION

This experiment is for:

1. Explore the construction of recommended system.
2. Understand the principle of matrix decomposition.
3. Be familiar to the use of gradient descent.
4. Construct a recommendation system under small-scale dataset, cultivate engineering ability.

II. METHODS AND THEORY

Algorithm 4 SGD Algorithm

1: **Require** feature matrices $\mathbf{P}$, $\mathbf{Q}$, observed set Ω, regularization parameters λ_p, λ_q and learning rate α.
2: **Randomly** select an observed sample $r_{u,i}$ from observed set Ω.
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 and 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**.

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 $\mathbf{R}$.
- λ_p and λ_q are **regularization parameters** to avoid overtting.

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
3. u1.base / u1.test are train set and validation set respectively, seperated 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. Experiment Step

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 against the raw data, and fill 0 for null values.
2. Initialize the user factor matrix and the item (movie) factor matrix , where is the number of potential features.
3. Determine the loss function and hyperparameter learning rate and the penalty factor .
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 and ;
   4.4 Calculate the on the validation set, comparing with the of the previous iteration to determine if it has converged.
5. Repeat step 4. several times, get a satisfactory user factor matrix and an item factor matrix , **Draw a curve with varying iterations**.
6. The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix .

C. Implementation

Source code:

```
import numpy as np
import pandas as pd

header=['user_id','item_id','rating','timestamp']
df = pd.read_csv('ml-100k/u.data',sep='\t',names=header,enc
n_users = df.user_id.unique().shape[0]
n_items = df.item_id.unique().shape[0]
print('Number of users ='+ str(n_users)+ ' | Number of movi
from sklearn.model_selection import train_test_split
train_data,test_data=train_test_split(df,test_size=0.25)
train_data= pd.DataFrame(train_data)
test_data= pd.DataFrame(test_data)

R = np.zeros((n_users, n_items))
for line in train_data.itertuples():
    R[line[1]-1, line[2]-1] = line[3]

T = np.zeros((n_users, n_items))
for line in test_data.itertuples():
    T[line[1]-1, line[2]-1] = line[3]
```

```
I = R.copy()
I[I > 0] = 1
I[I == 0] = 0

I2 = T.copy()
I2[I2 > 0] = 1
I2[I2 == 0] = 0
```

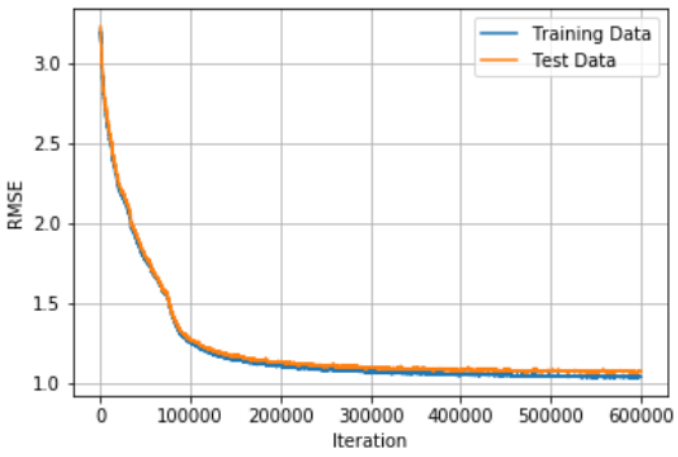
```
#predict the unknown ratings
def prediction(P,Q):
    return np.dot(P.T,Q)
```

```
lmbda = 0.1 # Regularisation weight
k = 2  # Dimensionality of the latent feature space
m,n = np.shape(R)  # Number of users and items
n_epochs = 8  # Number of epochs
gamma = 0.05 # Learning rate
```

```
P = 3 * np.random.rand(k,m) # Latent user feature matrix
Q = 3 * np.random.rand(k,n) # Latent movie feature matrix

def rmse(I,R,Q,P):
    return np.sqrt(np.sum((I * (R - prediction(P,Q)))**2)/len(R
#     return np.sqrt(np.mean(I * (R - prediction(P,Q))**2)) •
```

```
L_train = []
L_validation= []
#Only consider non-zero matrix
users,items = R.nonzero()
for epoch in range(n_epochs):
#     count = 0
    for (u,i) in zip(users,items):
#         if count<50000:
        e = R[u,i] - prediction(P[:,u],Q[:,i])  # Calculate error fo
        P[:,u] = P[:,u] + gamma * (e * Q[:,i] - lmbda * P[:,u])  # U
        Q[:,i] = Q[:,i] + gamma * (e * P[:,u] - lmbda * Q[:,i])  # U
        train_rmse = rmse(I,R,Q,P) # Calculate root mean squared erro
        test_rmse = rmse(I2,T,Q,P)# Calculate root mean squared erro
        L_train.append(train_rmse)
        L_validation.append(test_rmse)
#         L_train[count]=train_rmse
#         L_validation[count]=test_rmse
#         count+=1
```

```
import matplotlib.pyplot as plt
%matplotlib inline
# Check performance by plotting train and test errors
plt.plot( L_train, label='Training Data')
plt.plot( L_validation, label='Test Data')
plt.xlabel('Iteration')
plt.ylabel('RMSE')
plt.legend()
plt.grid()
plt.show()

# calculate prediction matrix
R = pd.DataFrame(R)
R_hat = pd.DataFrame(prediction(P, Q))

# compare true ratings of user 17 with predictions
ratings = pd.DataFrame(data=R.loc[16, R.loc[16,:] > 0]).head(n=5)
ratings['Prediction'] = R_hat.loc[16, R.loc[16,:] > 0]
ratings.columns = ['Actual Rating', 'Predicted Rating']
ratings
```



	Actual Rating	Predicted Rating
0	4.0	3.281555
6	4.0	3.442064
8	3.0	3.360856
12	3.0	3.496814
99	4.0	3.487000

IV. CONCLUSION

In this experiment we chose to use SGD method to complete the experiment, we encountered some problems, but in the end are discussed and access to relevant information to solve, by setting the parameters again and again, and finally we get a relatively good Convergence and loss.

Through this experiment, we see the SGD algorithm for the versatility of machine learning, but also deeply understand his importance. We understand the way to implement the recommender system, mastered the principle of matrix decomposition, and we gained a lot through this experiment.