The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Ruiyong Li
Xin Shu
Xingyang Cai

Supervisor:
Qingyao Wu

Student ID：
201530612033
201530612705
201530611074

Grade:
Undergraduate or Graduate

December 9, 2017

Linear Regression, Linear Classification and Gradient Descent

Abstract—

I. INTRODUCTION

Motivation

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.

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.

II. METHODS AND THEORY

Matrix Decomposition; Recommender System; stochastic gradient descent(SGD)

III. EXPERIMENT

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.

```
train_path =  'C:/Users/47864/Desktop/ml-100k/ml-100k/u1.base'
test_path = 'C:/Users/47864/Desktop/ml-100k/ml-100k/u1.test'
train = np.loadtxt(train_path)
test = np.loadtxt(test_path)
R = np.zeros((943,1682))
for sample in train:
    R[int(sample[0])-1][int(sample[1])-1] = int(sample[2]) #the sparse user score matrix
```

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.

```
user_gradient = np.zeros(user_w.shape)
movie_gradient = np.zeros(movie_w.shape)
for row in range(user_w.shape[0]):
    col = int(random.choice(train[train[:,0]== (row+1)])[1])-1
    user_gradient[row] = -movie_w.T[col]*(R[row][col] -np.dot(user_w[row],movie_w.T[col]))

for col in range(movie_w.shape[1]):
    try:
        row = int(random.choice(train[train[:,1]== (col+1)])[0])-1
    except:
        pass
```

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

```
def compute_loss(user_w,movie_w,dataset):
    sum_loss = 0
    for sample in dataset:
        pred_rate = np.dot(user_w[int(sample[0]),:],movie_w[:,int(sample[1])])
        sum_loss += 1/2*(pred_rate-sample[2])**2
    return sum_loss/len(dataset)
```

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;

```
def compute_loss(user_w,movie_w,dataset):
    sum_loss = 0
    for sample in dataset:
        pred_rate = np.dot(user_w[int(sample[0]),:],movie_w[:,int(sample[1])])
        sum_loss += 1/2*(pred_rate-sample[2])**2
    return sum_loss/len(dataset)
```

   4.3 Use SGD to update the specific row(column) of $P_{n\_users,K}$ and $Q_{n\_item,K}$;

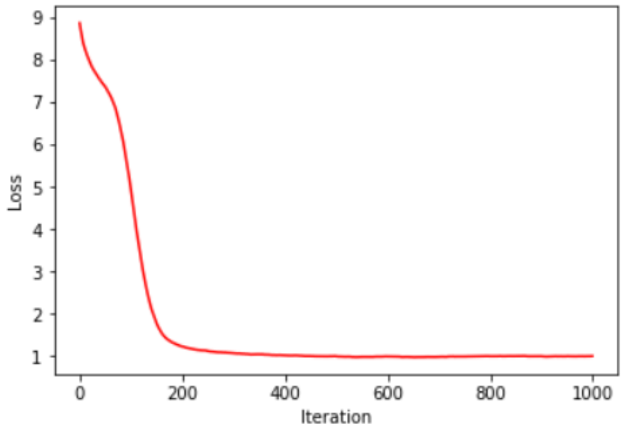
```
user_w -= learning_rate*user_gradient
movie_w -= learning_rate*movie_gradient
return user_w,movie_w
```

   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.

```
loss_list = []
loss = compute_loss(user_w,movie_w,test)
loss_list.append(loss)
print(loss)
```

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.



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

IV. CONCLUSION

From the $L_{validation}$ curve with varying iterations, we can find that the minimum loss is appeared(0.98002206146) when the iteration is about 543.