South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Deng Huang
Haoming Xu
Ruiqiu Cao

Supervisor:
Qingyao Wu

Student ID：
201530211588
201530781357
201330610031

Grade:
Undergraduate or Graduate

December 26, 2017

Recommender System Based on Matrix Decomposition

Abstract— *Recommonder System applise statistical and knowledge discovery techniques to make the product recommon- -dations. The main idea of the system is CF(Collaborative Fitering).*

In this paper, we chose the Matrix Factorization(one way of the model based CF)technique to implement it. We imple- ment it both in ALS and SGD, compare them first. Then we study the choice of the two parameters---C and K.

I. INTRODUCTION

We do this experiment for the purpose of:

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

The main idea and formulas that use in the paper & the pseudocode of algorithm:

A. ALS

Objective function:

$$\mathcal{L} = \sum_{u,i}(r_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i)^2 + \lambda(\sum_u n_{\mathbf{p}_u}||\mathbf{p}_u||^2 + \sum_i n_{\mathbf{q}_i}||\mathbf{q}_i||^2)$$

Optimize $\mathbf{P}$ while **fixing** $\mathbf{Q}$:

The first order optimality:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{p}_u} = 0$$

$$\Rightarrow \sum_{r_{u,i}\neq 0}(\mathbf{q}_i\mathbf{q}_i^\top + \lambda n_{\mathbf{p}_u} I)\cdot \mathbf{p}_u = \mathbf{Q}^\top \cdot \mathbf{R}_{u*}^\top$$

$$\Rightarrow \mathbf{p}_u = (\mathbf{q}_i\mathbf{q}_i^\top + \lambda n_{\mathbf{p}_u} I)^{-1}\cdot \mathbf{Q}^\top \cdot \mathbf{R}_{u*}^\top$$

$\mathbf{R}_{u*}$ denotes the u-th row of rating matrix $\mathbf{R}$.

Update **all** $\mathbf{p}_u$ with the above formula.

Optimize $\mathbf{Q}$ while **fixing** $\mathbf{P}$:

The first order optimality:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}_i} = 0$$

$$\Rightarrow \sum_{r_{u,i}\neq 0}(\mathbf{p}_u\mathbf{p}_u^\top + \lambda n_{\mathbf{q}_i} I)\cdot \mathbf{q}_i = \mathbf{P}^\top \cdot \mathbf{R}_{*i}^\top$$

$$\Rightarrow \mathbf{q}_i = (\mathbf{p}_u\mathbf{p}_u^\top + \lambda n_{\mathbf{q}_i} I)^{-1}\cdot \mathbf{P}^\top \cdot \mathbf{R}_{*i}$$

$\mathbf{R}_{*i}$ denotes the i-th column of rating matrix $\mathbf{R}$.

Update **all** $\mathbf{q}_i$ with the above formula.

Algorithm 2 ALS Algorithm

1: **Require** rating matrix $\mathbf{R}$, feature matrices $\mathbf{P}$, $\mathbf{Q}$ and regularization parameter λ.

2: **Optimize** $\mathbf{P}$ while fixing $\mathbf{Q}$:

$$\mathbf{p}_u = (\mathbf{q}_i\mathbf{q}_i^\top + \lambda n_{\mathbf{p}_u} I)^{-1}\cdot \mathbf{Q}^\top \cdot \mathbf{R}_{u*}^\top.$$

3: **Optimize** $\mathbf{Q}$ while fixing $\mathbf{P}$:

$$\mathbf{q}_i = (\mathbf{p}_u\mathbf{p}_u^\top + \lambda n_{\mathbf{q}_i} I)^{-1}\cdot \mathbf{P}^\top \cdot \mathbf{R}_{*i}.$$

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

B. SGD

Objective function:

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

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**:

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

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

Objective function:

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

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 $\mathbf{P}$ and $\mathbf{Q}$ with **learning rate** α:

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

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

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**.

III. EXPERIMENT

1. Data Set

We use MovieLens-100k dataset in the experiment,consisting 100000 comments from 943 users out of 1682 movies.

2. Implentation

ALS:

```
def ALS_fit(self, epoch, r_matrix_train, \
            r_matrix_test, train_pre, test_pre, I, I_t, n_comments, n_comments_t):
    R = r_matrix_train
    E = np.eye(self.k)
    t_pre = train_pre

    it_num = []
    train_loss = []
    test_loss = []
    train_rmse = []
    test_rmse = []

    #开始进行交替最小二乘法
    for ep in range(epoch):

        print('ep' + str(ep+1) + ' success')
        print(self.loss(r_matrix_train, train_pre, I))
        it_num.append(ep)
        train_loss.append( self.loss(r_matrix_train, train_pre, I) )
        test_loss.append( self.loss(r_matrix_test, test_pre, I_t) )
        train_rmse.append( self.rmse(r_matrix_train, I, n_comments) )
        test_rmse.append( self.rmse(r_matrix_test, I_t, n_comments_t) )

        for i in range(self.P.shape[1]):
            if(t_pre[0][i] > 0):
                Q = I[i] * self.Q

                self.P[:,i] = \
                np.asarray( np.mat( (Q.dot(Q.T) + self.lamda * \
                t_pre[0][i] * E) ).I ).dot(Q).dot(R[i,:].T)
        I2 = I.T
        for i in range(self.Q.shape[1]):
            if(t_pre[1][i] > 0):
                P = I2[i] * self.P

                self.Q[:,i] = \
                np.asarray( np.mat( (P.dot(P.T) + self.lamda * \
                t_pre[1][i] * E) ).I ).dot(P).dot(R[:,i])

        if(ep>0 and abs(train_rmse[ep-1]-train_rmse[ep])<=0.001):
            break

    return it_num, train_loss, test_loss, train_rmse, test_rmse
```

SGD:

```
def SGD_fit(self, epoch, lr, r_matrix_train, \
            r_matrix_test, train_pre,test_pre, I, I_t, n_comments, n_comments_t):
    R = r_matrix_train
    users, items = R.nonzero()

    it_num = []
    train_loss = []
    test_loss = []
    train_rmse = []
    test_rmse = []

    #amount = len(users)
    for ep in range(epoch):

        print('ep' + str(ep+1) + ' success')
        #print(self.loss(r_matrix_test, test_pre, I_t))
        print(self.loss(r_matrix_train, train_pre, I))
        it_num.append(ep)
        train_loss.append( self.loss(r_matrix_train, train_pre, I) )
        test_loss.append( self.loss(r_matrix_test, test_pre, I_t) )
        train_rmse.append( self.rmse(r_matrix_train, I, n_comments) )
        test_rmse.append( self.rmse(r_matrix_test, I_t, n_comments_t) )

        for u,i in zip(users,items):
            e = R[u][i] - dot(self.P[:,u].T, self.Q[:,i])
            self.P[:,u] += lr * (e * self.Q[:,i] - self.lamda * self.P[:,u])
            self.Q[:,i] += lr * (e * self.P[:,u] - self.lamda * self.Q[:,i])

        if(ep>0 and abs(train_rmse[ep-1]-train_rmse[ep])<=0.001):
            break

    return it_num, train_loss, test_loss, train_rmse, test_rmse
```

3. Experiment:

Default setting

Set

k = 20, lamda = 0.1

lr = 0.01(for SGD)

Explanation about my loss

I set the loss functon to mutiply 0.5(loss = 0.5*loss),because I do not want to mutiply 2 when taking the derivative.

A. Just running the programm to See the result:

a. result

Set

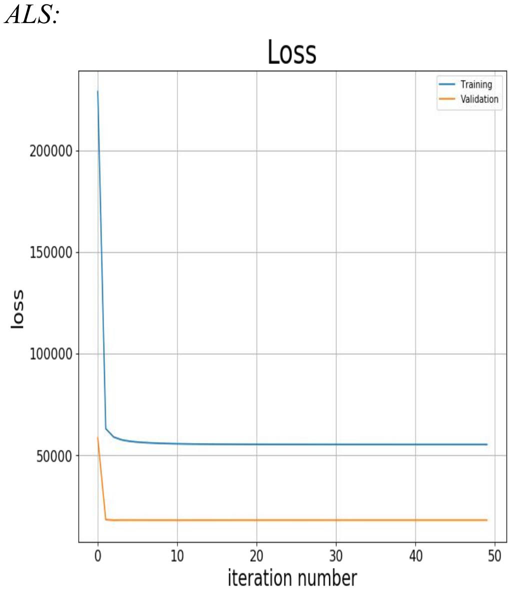
Iteration = 50

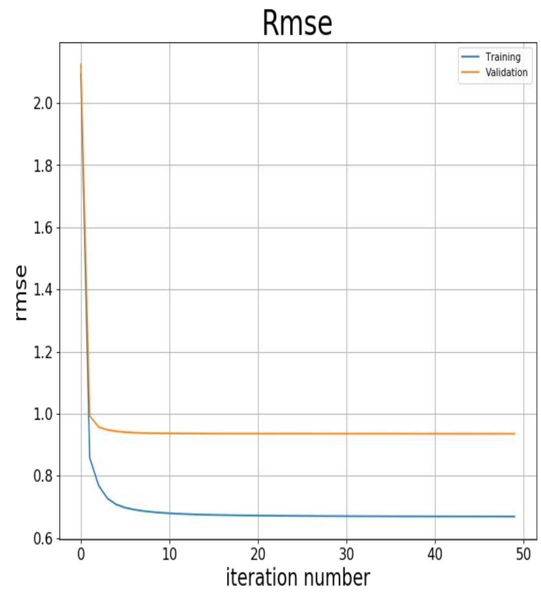
k = 20, lamda = 0.1

lr = 0.01(for SGD)

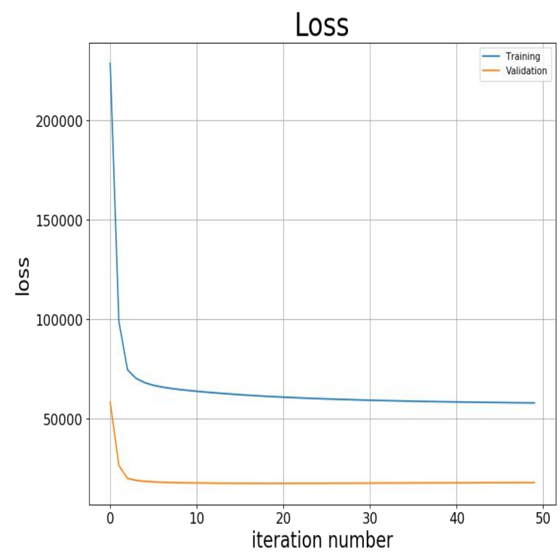
P,Q init with random
（42 seed）

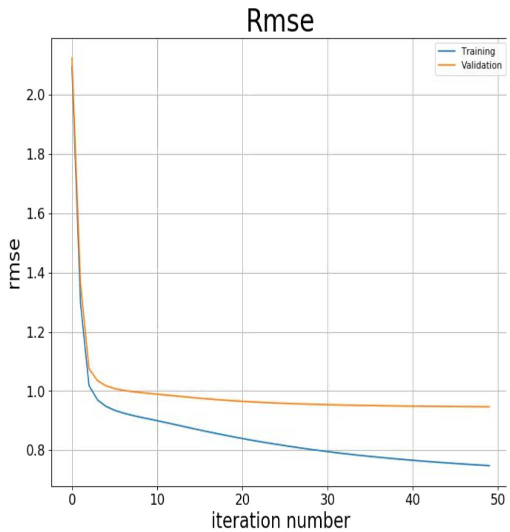
ALS:





SGD:





b. Analysis:

They can convergence after iteration.The implementation may be right.

B. Compare the ALS and SGD

a. Speed of convergence:

Set K = 5, 20, 50 to compare the speed:

ALS：

```
收敛耗时：
68.68455362319946
drawing,please wait
训练集上的rmse:
0.8021495994357293
验证集上的rmse:
0.9305916222210723
```

```
收敛耗时：
156.2533040046692
drawing,please wait
训练集上的rmse:
0.6688881807131867
验证集上的rmse:
0.9351614966958692
```

```
收敛耗时：
281.5046417713165
drawing,please wait
训练集上的rmse:
0.6113981481762778
验证集上的rmse:
0.9353843760822285
```

SGD：

```
收敛耗时：
494.3064298629761
drawing,please wait
训练集上的rmse:
0.9210679131296331
验证集上的rmse:
1.0440300339546669
```

```
收敛耗时：
493.03716468811035
drawing,please wait
训练集上的rmse:
0.7623435866607963
验证集上的rmse:
1.0297135890862759
```

```
收敛耗时：
523.23153424263
drawing,please wait
训练集上的rmse:
0.6907068964572054
验证集上的rmse:
0.9556509024129339
```

b. Analysis

b.1

We can see as the K increses, ALS’s speed increases,but SGD do not have much obvious change.

It accords with computational compexity:

ALS:

$$O(|\Omega|k^2 + (m+n)k^3)$$

SGD:

$$O(|\Omega|k)$$

b.2

But we can see ALS's conver--gence is faster.

So,in this experiment,what I set K just 1~50,ALS's speed is faster.

C. Test on parameters K,C:

Set K = 5,20,50

C = 0.01, 0.1, 0.3, 1

I set that if the train rmse change between two iteration is less than 0.001, it will be seen as conver--gence.

Do tests with ALS

a. The result table

lamda/k	0.01	0.1	0.3	1
5	0.775209063	0.803969643	0.94486837	1.355317513
	1.064854795	0.932579931	0.98822204	1.379623015
20	0.487880448	0.674630299	0.94466182	1.355349426
	1.236632499	0.935898293	0.98924916	1.379215598
50	0.195590682	0.616475335	0.94494501	1.355354654
	1.189968099	0.935655001	0.99589012	1.382268661
	the traning rmse is beyond			
	the validation rmse is below			
	t_rmse/v_rmse			

b. Analysis

The lamda cannot be set to a too larger value,or the train rmse and the validaton rmse will be unsatisfied both.

The resut of k,which indicate to the factor impacting the rating is different with my first assumpttion. K can not be too large too.

So,at last lamda = 0.3 and k = 5, may seen good.

IV. CONCLUSION

In the experiment we construct the reco--mmended system using MF. We have a deeper understand on MF after using two way to implement it and compare the ways

The SGD in this experiment is different with which in former threes.And we get the new understand on it after using it on this way.

At last, by applying what we learned to implement a real system, our engineering ability get much improved.