The Experiment Report of Machine Learning

SCHOOL: South China University of Technology

SUBJECT: SOFTWARE ENGINEERING

Author:

Peiwei Chang, Zhiwei Wu and Jingjing Hu

Supervisor:

Mingkui Tan

Student ID：

201530611128, 201530613092 and 201530611609

Grade:

Undergraduate

December 28, 2017

Recommender System Based on Matrix Decomposition

Abstract—Recommender systems have been proven to be valuable means for web online users to cope with the information overload and have become one of the most powerful and popular tools in electronic commerce. In this paper, we realize the Matrix Factorization (MF) algorithm through Alternating Least Square (ALS) and Stochastic Gradient Descent (SGD)

I. INTRODUCTION

The motivation of this exploratory study is:

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

Matrix Factorization

Give a rating matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$, with sparse ratings from *m* users to *n* items. Assume rating matrix **R** can be factorized into the multiplication of two low-rank feature matrices $\mathbf{P} \in \mathbb{R}^{m\times k}$ and $\mathbf{Q} \in \mathbb{R}^{k\times n}$.

Then determine an objective function.

Squared error loss: $\mathcal{L}(r_{u,i}, \hat{r}_{u,i}) = (r_{u,i} - \hat{r}_{u,i})^2$

Binary hinge loss: $\mathcal{L}(r_{u,i}, \hat{r}_{u,i}) = max(0, 1 - r_{u,i}\hat{r}_{u,i})$

Note:

$r_{u,i}$ denotes the actual rating of user *u* for item *i* and $\hat{r}_{u,i}$ denotes the prediction.

Alternating Least Square (ALS) is to minimize the following objective function:

$$\mathcal{L} = \sum_{u,i}(r_{u,i} - p_u^T q_i)^2 + \lambda\left(\sum_u n_{p_u}||p_u||^2 + \sum_i n_{q_i}||q_i||^2\right)$$

Note:

$\mathbf{P} = [p_1, p_2, \dots, p_m]^T \in \mathbb{R}^{m\times k}$

$\mathbf{Q} = [q_1, q_2, \dots, q_n] \in \mathbb{R}^{k\times n}$

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

λ is regularization parameter to avoid overfitting.

n_{p_u} and n_{q_i} denote the number of total ratings on user u and item i, respectively.

General Steps of ALS:

1. Require rating matrix **R**, feature matrices **P**, **Q** and regularization parameter λ.
2. Optimize **P** while fixing **Q**.
   The first order optimality: $\frac{\partial \mathcal{L}}{\partial p_u} = 0$
   $$\Rightarrow \sum_{r_{u,i}\neq 0}\left(q_i q_i^T + \lambda n_{p_u} I\right)\cdot p_u = \mathbf{Q}^T \cdot \mathbf{R}_{u*}^T$$
   $$\Rightarrow p_u = \left(q_i q_i^T + \lambda n_{p_u} I\right)^{-1}\cdot \mathbf{Q}^T \cdot \mathbf{R}_{u*}^T$$
   Note:
   $\mathbf{R}_{u*}$ denotes the *u*-th row of rating matrix **R**.
   Then update all p_u with the above formula.
3. Optimize **Q** while fixing **P**.
   The first order optimality: $\frac{\partial \mathcal{L}}{\partial q_i} = 0$
   $$\Rightarrow \sum_{r_{u,i}\neq 0}\left(p_u p_u^T + \lambda n_{q_i} I\right)\cdot q_i = \mathbf{P}^T \cdot \mathbf{R}_{*i}^T$$
   $$\Rightarrow q_i = \left(p_u p_u^T + \lambda n_{q_i} I\right)^{-1}\cdot \mathbf{P}^T \cdot \mathbf{R}_{*i}^T$$
   Note:
   $\mathbf{R}_{*i}$ denotes the *i*-th column of rating matrix **R**.
   Then update all q_i with the above formula.
4. Repeat the above processes until convergence.

Stochastic Gradient Descent(SGD) is to minimize the following objective function:

$$\mathcal{L} = \sum_{u,i\in\Omega}(r_{u,i} - p_u^T q_i)^2 + \lambda_p||p_u||^2 + \lambda_q||q_i||^2$$

Note:

$\mathbf{P} = [p_1, p_2, \dots, p_m]^T \in \mathbb{R}^{m\times k}$

$\mathbf{Q} = [q_1, q_2, \dots, q_n] \in \mathbb{R}^{k\times n}$

$r_{u,i}$ 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.

General Steps of SDG:

1. Require feature matrices **P, 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.
   Calculate the prediction error:
   $$E_{u,i} = r_{u,i} - p_u^T q_i$$
   Calculate the gradient:
   $$\frac{\partial \mathcal{L}}{\partial p_u} = E_{u,i}(-q_i) + \lambda_p p_u$$

$$\frac{\partial \mathcal{L}}{\partial q_i} = E_{u,i}(-p_u) + \lambda_q q_i$$

4. Update the feature matrices **P** and **Q** with learning rate α and gradient.

$$p_u = p_u + \alpha\left(E_{u,i}q_i - \lambda_p p_u\right)$$
$$q_i = q_i + \alpha\left(E_{u,i}p_u - \lambda_q q_i\right)$$

5. Repeat the above processes until convergence.

Time complexity per iteration of ALS:$O(|\Omega|k^2 + (m+n)k^3)$
Note:
$|\Omega|$ denotes the number of observed samples.
k denotes the rank.
m and *n* denote the number of users and items.
So ALS is not scalable to large-scale datasets.

Time complexity per iteration of SGD: $O(|\Omega|k)$
So SGD is scalable to large-scale datasets.

Expect for the time complexity, SGD and ALS also differ from:

- ALS is easier to parallelize than SGD.
- ALS converges faster than the SGD.
- SGD has less storage complexity than ALS. (ALS needs to store the rating matrix R)
- SGD has less computational complexity than ALS. (ALS needs to compute the matrix-vector multiplication)

III. EXPERIMENT

Dataset:

The dataset of this experiment utilizing is MovieLens-100k dataset, which consist 10,000 ratings from 943 users out of 1,682 movies. And at least each user give 20 movies ratings. The dataset has its own structure like the table below.

User id	Item id	Rating	Timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116

First, I read the dataset 'u1.base' as my training set and 'u1.test' as validation set. According to the training data, I can initial the scoring matrix $\mathbf{R}_{nusers,nitems}$. Every row of **R** refers to a user and column refers to a movie. If we the find the user gives its rating to the movie, the value will be the rating. If not, the initial value will be 0.

After initializing the scoring matrix $\mathbf{R}_{nusers,nitems}$, I initialize the user factor matrix $\mathbf{P}_{nusers,K}$and the item(movie) factor matrix $\mathbf{Q}_{nitem,K}$, given K=5.

Determine that the Squared loss as loss function, and learning rate 0.002, and the regularization parameter 0.02. The next step is doing the decomposition. I use alternate least squares optimization method to do that:

```
for step in range(steps):
    for i in range(len(R)):

        for j in range(len(R[i])):
            if R[i][j] > 0:
```

$$e_{i,j} = R[i][j] - numpy.dot(P[i,:], Q[:,j])$$

1. With fixed item factor matrix, find the loss partial derivative of each row (column) of the user factor matrices, ask the partial derivative to be zero and update the user factor matric.

```
for k in range(K):
```

$$\mathbf{P}[i][k] = \mathbf{P}[i][k] + \alpha\left(2e_{i,j}\mathbf{Q}[k][j] - \beta\mathbf{P}[i][k]\right)$$

2. With fixed user factor matrix, find the loss partial derivative of each row (column) of the item factor matrices, ask the partial derivative to be zero and update the item factor matric.

```
for k in range(K):
```

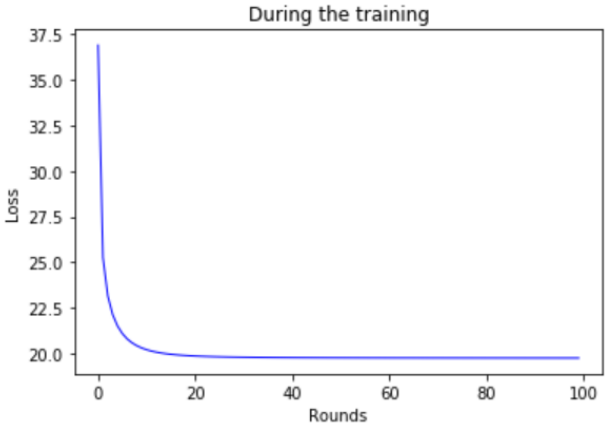
$$\mathbf{Q}[k][j] = \mathbf{Q}[k][j] + \alpha\left(2e_{i,j}\mathbf{P}[i][k] - \beta\mathbf{Q}[k][j]\right)$$

3. Calculate the loss on the validation set.

```
for each in TestData:
    userid, movieid, rating=each[0], each[1], each[2]
    predict_rating=R[int(userid)-1][int(movieid)-1]
    loss+=pow(predict_rating-rating, 2)
```

After multi times of the 3 steps, we can get the final user factor matrix **P** and the item factor matrix **Q**. And after the dot of **P** and $\mathbf{Q}^T$, we can get the final score prediction matrix **R**.

Result:
K=6, learning rate=0.002, regularization parameter=0.02



IV. CONCLUSION

In this experiment, we implement the matrix decomposition based on the alternate least squares optimization method.

After the implementation, I have an outline of recommender system in mind. By converging the loss value, I really get sense of achievement. And during the experiment, I found that if the learning rate is a bit higher, the loss value might not converge. And if K is a bit lower, it will lead to the similar result. So it's really hard to find a balance between speed and quality of model.

But finally, this experiment reinforces my understanding of recommender system indeed.