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:
邢浩、何嘉豪、黄迦密

Supervisor:
Mingkui Tan

Student ID：201530613221(邢 浩
)201530361276(何嘉豪)
201537611718(黄迦密)

Grade:
Undergraduate

December 9, 2017

Recommender System Based on Matrix Decomposition

Abstract—According to machine learning and matrix decomposition theory, this experiment is aimed to realize a recommender system.

I. Introduction

A. Motivation of Experiment

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.

B. 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.

C. Environment for Experiment

python3, at least including following python package: sklearn, numpy, matplotlib, jupyter.
An advice is installing anaconda3 directly, which already contains the python package mentioned above.

II. Methods and Theory

A. Basic Ideas
Matrix factorization is to factorize a matrix to find out two (or more) matrices such that when you multiply them you will get back the original matrix.
From an application point of view, matrix factorization can be used to discover latent features underlying the interactions between two different kinds of entities. And one obvious application is to predict ratings in collaborative filtering.
In a recommendation system such as Netflix or MovieLens, there is a group of users and a set of items (movies for the above two systems). Given that each users have rated some items in the system, we would like to predict how the users would rate the items that they have not yet rated, such that we can make recommendations to the users.
The intuition behind using matrix factorization is that there should be some latent features that determine how a user rates an item.
B. Mathematics of Matrix Decomposition
The matrix that contains all the ratings that the users have assigned to the items is

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$

so that

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

The squares loss is

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^{K} (||P||^2 + ||Q||^2)$$

The update rule is

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha(2 e_{ij} q_{kj} - \beta p_{ik})$$
$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2 e_{ij} p_{ik} - \beta q_{kj})$$

We can check the overall error as calculated using the following equation and determine when we should stop the process.

$$E = \sum_{(u_i, d_j, r_{ij}) \in T} e_{ij} = \sum_{(u_i, d_j, r_{ij}) \in T} (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

III. Experiment

A. Experiment Step

The experiment code and drawing are both completed on jupyter.

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

5. Repeat step 4. several times, get a satisfactory user factor matrix $\mathbb{P}$ and an item factor matrix $\mathbb{Q}$, **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 $Q_{n_{item},K}$.

B. Code

1. Loss function:

```
eij = R1[i][j] - numpy.dot(P[i,:],Q[:,j])
```

2. Update function:

```
P[i,:] = P[i,:] + alpha * (eij * Q[:,j] - beta * P[i,:])
Q[:,j] = Q[:,j] + alpha * (eij * P[i,:] - beta * Q[:,j])
```

3. Data set:

```
u1.base/u1.test
```

4. Parameters initialization:

```
K = 2
steps = 1000
alpha = 0.001
beta = 0.02
```

5. Computing loss:

Training loss:

```
for i in range(len(R1)):
    for j in range(len(R1[i])):
        if R1[i][j] > 0:
            error0 += pow(R1[i][j] - numpy.dot(P[i,:],Q[:,j]), 2)\
                +(beta/2) * numpy.dot(P[i,:],P[i,:])\
                +(beta/2) * numpy.dot(Q[:,j],Q[:,j])
```

Validation loss:

```
for i in range(len(R2)):
    for j in range(len(R2[i])):
        if R2[i][j] > 0:
            error1 += pow(R2[i][j] - numpy.dot(P[i,:],Q[:,j]), 2)\
                + (beta/2) * numpy.dot(P[i,:], P[i,:])\
                + (beta/2) * numpy.dot(Q[:,j], Q[:,j])
```

C. Result

The codes run successfully. However, our validation loss (total loss) is less than training loss(total loss) as shown in figure 1.1, which is confusing.

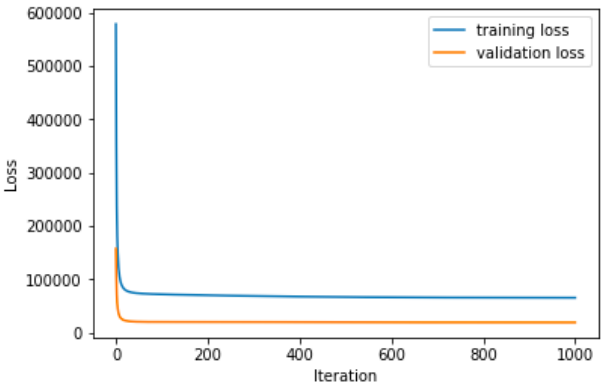

Figure 1.1

IV. CONCLUSION

Through this experiment, we experience making a recommender system on matrix decomposition and gain further understanding about it. What's more, we find that Machine learning could solve problems maybe more than we expected and is only waiting for us to discover.