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:
YangJun Xu and GongDa Liang and AoXiang Zhang

Supervisor:
Mingkui Tan

Student ID：
201530371633 and 201530612101 and 201530613580

Grade:
Undergraduate

December 16, 2017

Recommender System Based on Matrix Decomposition

Abstract—The recommender system is a research issue that is getting hotter and hotter now, Matrix decomposition is a more effective and fast way to implement the recommender system. High dimensional user-item scoring matrix is decomposed into two low-dimensional user factor matrices and item factor matrices. Use the product of these two low-dimensional matrices to fit the scoring matrix to predict the user's rating of the item, in order to recommend users with higher ratings of the item.

I. INTRODUCTION

The application of the recommender system is more and more widely used, and the recommended accuracy is the key to the recommender system. We will try to build a recommender system using matrix decomposition

We get two algorithms to implement the recommender systems ALS and SGD. ALS is less capable of handling large amounts of data than SGD, so we chose the SGD algorithm to implement the recommender system.

II. METHODS AND THEORY

In this experiment, we will use Stochastic Gradient Descent to matrix decomposition algorithm.

1. Matrix Factorization

Decompose high dimensional rating matrix R ∈ $\mathbb{R}^{m\times n}$ into two low dimensional user factor and item feature matrices.P ∈ $\mathbb{R}^{m\times k}$ and $Q \in \mathbb{R}^{k\times n}$.We will use the product of two low-rank feature matrices to predict the rating which users don't score. We chose the squared error loss as the objective function

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

2. Stochastic Gradient Descent

We use stochastic gradient descent to get the two low-rank feature matrices. Randomly select an observed sample $r_{u,i}$ from observed set Ω, calculate the gradient to the objective function and update the feature matrices P and Q.

SGD is to minimize the objective function

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

$$\text{Prediction error: } E_{u,i} = r_{u,i} - {p_u}^T q_i$$

$$\text{Gradient: } \frac{\partial \mathcal{L}}{\partial p_u} = E_{u,i}(-q_i) + \lambda_p p_u$$

```
def matrix_factorization(R, K, R_test, iterations = 15000, learning_rate = 0.04, lamda = 0.02, step = 200):
    P = np.random.rand(R.shape[0], K)
    Q = np.random.rand(K, R.shape[1])
    p = np.zeros((943))
    q = np.zeros((1682))
    l_validation = np.zeros((int(iterations / step)))
    u = 0
    i = 0
    for t in range(0, iterations):
        while True:
            u = rand.randint(0, 942)
            i = rand.randint(0, 1681)
            if R[u][i] > 0:
                break

        eui = R[u][i] - np.dot(P[u, :], Q[:, i])

        P[u] = P[u] + learning_rate * (2 * eui * Q[:, i] - lamda * P[u])
        Q[:, i] = Q[:, i] + learning_rate * (2 * eui * P[u] - lamda * Q[:, i])

        if t % step == 0:
            print(t)
            for u in range(0, 943):
                p[u] = lamda * np.linalg.norm(P[u])**2

            for i in range(0, 1682):
                q[i] = lamda * np.linalg.norm(Q[:, i])**2

            for u in range(0, 943):
                for i in range(0, 1682):
                    if R_test[u][i] > 0:
                        l_validation[int(t / step)] += pow(R_test[u][i] - np.dot(P[u].T, Q[:, i]), 2) + p[u] + q[i]
```

Figure 1

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

III. EXPERIMENT

1. Datasets

In this experiment, we use MovieLens-100k dataset. This dataset were collected by the GroupLens Research Project at the University of Minnesota. The data was collected through the MovieLens web site (movielens.umn.edu) during the seven-month period from September 19th, 1997 through April 22nd, 1998. It consists of 100,000 ratings (1-5) from 943 users on 1682 movies, and each user has rated at least 20 movies. We use u1.base and u1.test which are seperated from dataset u.data with proportion of 80% and 20%.

2. Experimental Setup

In this experiment, we use SGD to decompose the rating matrix. We calculate the gradient by observed sample selected randomly.

3. Algorithm Implementation

We implementation the algorithm as Figure 1. In this code, we should calculate the loss of validation set, but it takes long time. So we calculate the loss each 200 iterations.

Figure 2 show the loss of validation set has converged

IV. CONCLUSION

Through this experiment, we use the matrix decomposition method to build a recommendation system, and use a method of stochastic gradient descent to update the two low-rank feature matrices.

Because of the use of a stochastic gradient descent method, it takes less time to update the matrix, but calculating loss requires a lot of time, so we calculate a loss every 200 iterations.

REFERENCES

Albert Au Yeung Matrix Factorization: A Simple Tutorial and Implementation in Python
http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/

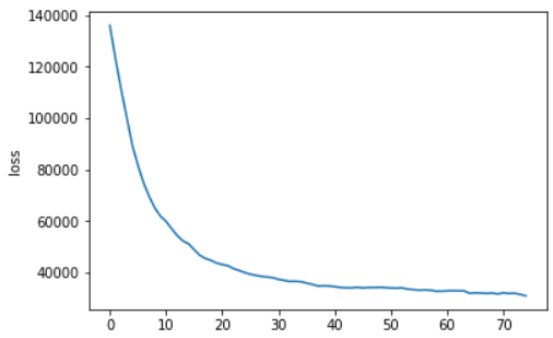

Figure 2