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:
Jianing Ren, Ziqi Ruan and
Zhengying Xing

Supervisor:
Mingkui Tan

Student ID：
201530612651, 201530612668
and 201530613238

Grade:
Undergraduate

December 28, 2017

Recommender System Based on Matrix Decomposition

Abstract—We implement a recommender system based on matrix decomposition. In this report, we record experiments we carried out to implement and access the performance of the recommender system.

I. INTRODUCTION

Recommender System applies statistical and knowledge discovery techniques to the problem of making product recommendations.

Our recommender system aims at estimate the user's rating for a movie through a certain number (about 80,000) of ratings. We want to explore the construction of recommended system, understand the principle of matrix decomposition, and to be familiar to the use of gradient descent through this experiment.

In the next section, we will discuss the methods and theory of recommender system in detail. After that, we conduct an experiment to verify the effectiveness of recommender system. In the last section, we summarize our paper.

II. METHODS AND THEORY

In this experiment we are going to use matrix decomposition to make prediction on the dataset. In the dataset we have users and their ratings on different movies, and we want to decompose the matrix into two matrixes P and Q, and summarize each user u with a k dimensional vector p_u, and each movie i with a k dimensional vector q_i. Then, to predict user u's rating for movie i, we simply predict $r_{ui} \approx p_u^T q_i$. Our goal is to estimate the complete rating matrix R $\approx P^T Q$. We can formulate this problem as an optimization problem in which we aim to minimize an objective function and find optimal X and Y. In particular, we aim to minimize the least squares error of the observed ratings (and regularize):

$$L = \sum_{u,i \in S} (r_{ui} - p_u^T q_i)^2 + \beta(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2)$$

If we fix the set of variables of variables X and treat them as constants, then the objective is a convex function of Y and vice versa. Our approach will therefore be to fix Y and optimize X, then fix X and optimize Y, and repeat until convergence. This approach is known as altering least squares. For the

Another approach to optimize the loss function is to use gradient descent. In each iteration we pick a sample from the training data, and we calculate the gradient at the current values, and therefore we differentiate the loss function with respect to these two variables separately.

The error of each element is calculated as:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^K p_{ik} q_{kj})^2$$

And we have the gradient of the two variables:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij} q_{kj}$$
$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij} p_{ik}$$

Having obtained the gradient, we can now formulate the update rules:

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

A common extension to this basic algorithm is to introduce regularization to avoid overfitting. So, the formula should be:

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

In the experiment, we used the SGD method, because it is faster. As we found that picking only one element in each iteration will make it hard to converge, in practice we used gradient descent, which means that in each iteration we compute the gradient of all samples and update the two matrixes.

III. EXPERIMENT

A. Dataset

In the experiment we use the dataset MovieLens-100k, which contains 100 thousand pieces of ratings from 943 users for 1682 movies. Each user scored at least 20 movies.

Moreover, we use the data in u2.base as training data. It contains 80% total data. The rest 20% data, which is in u2.test, is used as testing data.

B. Implementation

First, we initialize the original scoring matrix with the training data, and fill the null position with random value (less than 1). And we initialize the user factor matrix P and item (movie) factor matrix Q. We set K, the number of potential features, to 14.

Then, with learning rate 0.0002 and penalty coefficient 0.02, we begin a cyclic process until the loss go to a convergence:

1) compute the gradient descent of the loss function for P and Q for each sample which score is more than 0 in the scoring matrix.

2)update the matrix P and Q.

3)calculate the loss on testing dataset.

Finally, we draw the curve (Figure 1) of the relation between the loss on testing dataset and the number of iterations. And we get the final score prediction matrix by multiplying the matrix P and Q.

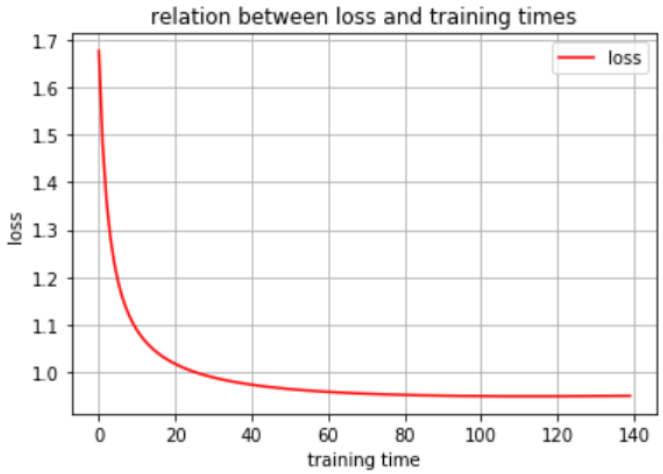

Figure 2

IV. CONCLUSION

We have implemented a recommender system and demonstrated that the matrix decomposition performs well on a recommender system problem. Through the process, we understand the theory of recommender system better. In the same time, we enhance our coding ability.