The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Jing Liang, Shoubin Li and
Mengdan Zheng

Supervisor:
Mingkui Tan

Student ID： 201530741368,
201530612040 and 20153061379
5

Grade:
Undergraduate

December 27, 2017

Recommender System based on Matrix Decomposition

Abstract—A experiment of recommender system using matrix decomposition. Then compare the loss of using ALS and SGD with graphing and drawing the losses with jupyter notebook.

I. Introduction

While user-based or item-based collaborative filtering methods are simple and intuitive, matrix factorization techniques are usually more effective because they allow us to discover the latent features underlying the interactions between users and items. In this experiment, we will explore the construction of recommender system, understand the matrix decomposition and construct a system under small-scale dataset, cultivate the engineer ability.

II. Methods and Theory

Our task, then, is to find two matrices P and Q such that their product approximates R:

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

For ALS, the loss function is:

$$L(P, Q) = \sum_{i,j} (r_{ij} - p_i^T q_j)^2 + \lambda(|p_i|^2 + |q_j|^2)$$

Our target is to minimize it:
Thus, for each row of P and Q^T, we need to let the gradient be zero to get an update of P_i and Q_i.
Fix the Q and update P:

$$p_i = (Q^T Q + \lambda I)^{-1} Q^T r_i$$

Fix the P and update Q:

$$q_j = (P^T P + \lambda I)^{-1} P^T r_j$$

For SGD, our loss function is:

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

Then we add all e_{ij}^2 and we will get total errors.
Thus, we compute the gradient and update the P and Q.

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

III. Experiment

The selection of hyper parameters:
ALS:

epoch	20
λ	0.02

SGD:

epoch	25000
λ	0.02
α	0.001

Experiment Steps:
Using alternate least squares optimization(ALS):

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 against the raw data, and fill 0 for null values.
2. Initialize the user factor matrix and the item (movie) factor matrix, where is the number of potential features.
3. Determine the loss function and the hyperparameter learning rate and the penalty factor.
4. Use alternate least squares optimization method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:
   4.1 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 matrices.
   4.2 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
   4.3 Calculate the on the validation set, comparing with the of the previous iteration to determine if it has converged.
5. Repeat step 4. several times, get a satisfactory user factor matrix and an item factor matrix, **Draw a curve with varying iterations**.
6. The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix .

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 against the raw data, and fill 0 for null values.
2. Initialize the user factor matrix and the item (movie) factor matrix, where is the number of potential

features.

3. Determine the loss function and hyperparameter learning rate and the penalty factor.
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 and;
   4.4 Calculate the on the validation set, comparing with the of the previous iteration to determine if it has converged.
5. Repeat step 4. several times, get a satisfactory user factor matrix and an item factor matrix, **Draw a curve with varying iterations**.
6. The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix

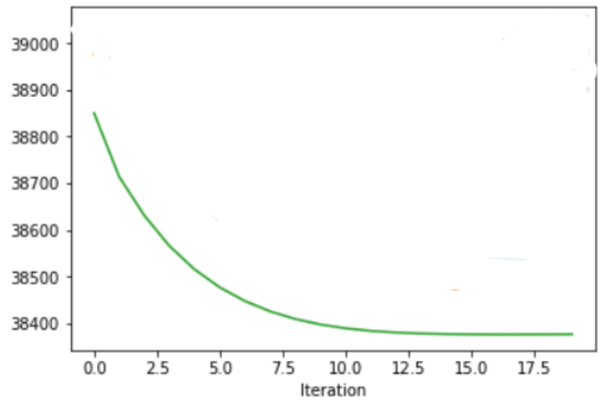

Fig1. Result of ALS

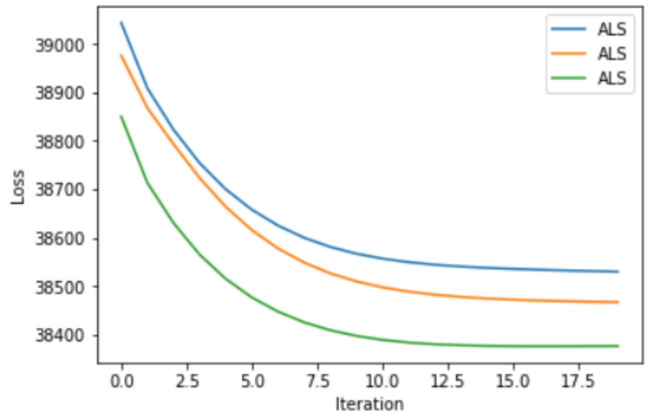

Fig2. Exception in figure

For ALS, it seems to have faster convergence and higher loss(the loss computation is similar to the SGD). What's more, when I try to optimize the hyper parameters, I find the phenomenon as Fig.2.

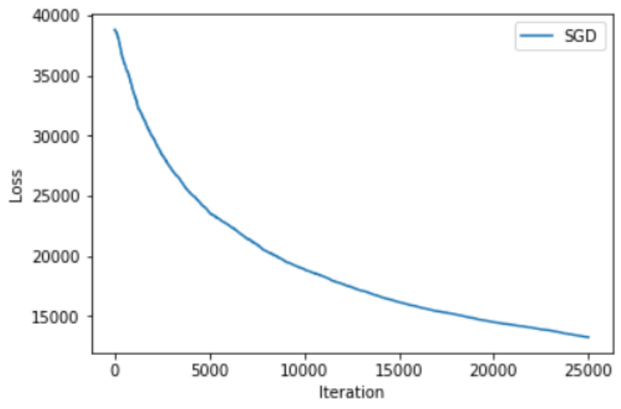

Fig 3. Result of SGD

The SGD runs quite slow (serval hours needed) and needs more iterations, however, its loss seems to be much smaller than the ALS.

IV. CONCLUSION

Matrix decomposition is a way to explore data and predict the reference of the users of items.
The main method is ALS and SGD. ALS is a way to make the gradient be zero then totally update each row of P and Q. However, the SGD, is purely calculate the gradient and make the classical gradient descent work.

We discuss the reason why we have problem about the updating in ALS:

I think it is the way in coding. In our code we update the items per row with matrix and use numpy methods, however, the sample code used in SGD updates each single elements instead of the whole row. Though the process seems the same, but the result looks differently.