The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
鲍淞琳，肖纾予 and 赵一鸣

Supervisor:
Mingkui Tan

Student ID：
201530611029，201530613160
and 201530661741

Grade:
Undergraduate

December 28, 2017

Recommender System Based on Matrix Decomposition

Abstract—Nowadays, recommender system is a popular application. So **t**his experiment applies stochastic gradient descent (SGD) and alternate least squares optimization(ALS) on recommend system based on Matrix Decomposition. And we draw the loss with the number of iterations in validation dataset.

I. Introduction

A recommender system is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item. Recommender systems have become increasingly popular in recent years, and are utilized in a variety of areas including movies, music, news, books, research articles and search queries in general. There are also recommender systems for experts, collaborators, restaurants, garments and Twitter pages.

Recommendations can be generated by a wide range of algorithms. 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. Of course, matrix factorization is simply a mathematical tool for playing around with matrices, and is therefore applicable in many scenarios where one would like to find out something hidden under the data.

The motivation of this experiment is as follows:

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

The mathematics of matrix factorization

Having discussed the intuition behind matrix factorization, we can now go on to work on the mathematics. Firstly, we have a set U of users, and a set D of items. Let R of size |U| \times |D| be the matrix that contains all the ratings that the users have assigned to the items. Also, we assume that we would like to discover K latent features. Our task, then, is to find two matrics matrices P (a |U| \times K matrix) and Q (a |D| \times K matrix) such that their product approximates R:

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

In this way, each row of P would represent the strength of the associations between a user and the features. Similarly, each row of Q would represent the strength of the associations between an item and the features. To get the prediction of a rating of an item d_j by u_i, we can calculate the dot product of the two vectors corresponding to u_i and d_j:

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

Now, we have to find a way to obtain P and Q. One way to approach this problem is the first intialize the two matrices with some values, calculate how `different' their product is to M, and then try to minimize this difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference.

The difference here, usually called the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair:

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

Here we consider the squared error because the estimated rating can be either higher or lower than the real rating.

To minimize the error, we have to know in which direction we have to modify the values of p_{ik} and q_{kj}. In other words, we need to know the gradient at the current values, and therefore we differentiate the above equation with respect to these two variables separately:

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

Having obtained the gradient, we can now formulate the update rules for both p_{ik} and q_{kj}:

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

Here, α is a constant whose value determines the rate of approaching the minimum. Usually we will choose a small value for α, say 0.0002. This is because if we make too large a step towards the minimum we may run into the risk of missing the minimum and end up oscillating around the minimum.

A question might have come to your mind by now: if we find two matrices $\mathbf{P}$ and $\mathbf{Q}$ such that $\mathbf{P} \times \mathbf{Q}$ approximates $\mathbf{R}$, isn't that our predictions of all the unseen ratings will all be zeros? In fact, we are not really trying to come up with $\mathbf{P}$ and $\mathbf{Q}$ such that we can reproduce $\mathbf{R}$ exactly. Instead, we will only try to minimise the errors of the observed user-item pairs. In other words, if we let T be a set of tuples, each of which is in the form of (u_i, d_j, r_{ij}), such that T contains all the observed user-item pairs together with the associated ratings, we are only trying to minimise every e_{ij} for $(u_i, d_j, r_{ij}) \in T$. (In other words, T is our set of training data.) As for the rest of the unknowns, we will be able to determine their values once the associations between the users, items and features have been learnt.

Using the above update rules, we can then iteratively perform the operation until the error converges to its minimum. 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. Datasets

1. *MovieLens-100k datasets*
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.*

B. Implementation

（1） *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 R_{n_users,n_items} against the raw data, and fill 0 for null values.

2. Initialize the user factor matrix $P_{n\_users,K}$ nd the item (movie) factor matrix $Q_{n\_item,K}$, where K is the number of potential features.

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

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 fixd 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 fixd 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 $L_{validation}$ 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 P and an item factor matrix 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 $Q_{n\_item,K}$.

（2）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 $P_{n\_users,K}$ nd the item (movie) factor matrix $Q_{n\_item,K}$, where K is the number of potential features.

3. Determine the loss function and the 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

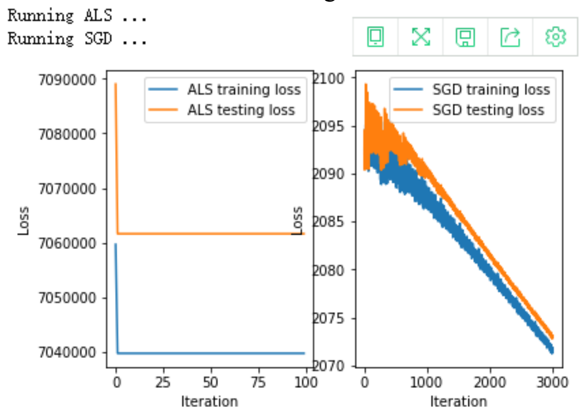
5. Repeat step 4. several times, get a satisfactory user factor matrix P and an item factor matrix Q, Draw a $L_{validation}$ curve with varying iterations.

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 $Q_{n\_item,K}$.

Parameters

the number of potential features of SGD	8
learning_rate of SGD	1e-2
regularization_coefficient of SGD	3e-2
the number of potential features of ALS	8

Figs



```
Running ALS ...
Running SGD ...
```

IV. CONCLUSION

In this experiment, we study the construction of recommended system and understand the principle of matrix decomposition. From this experiment, We also have a better understand of the method of SGD and ALS. And we realize that the influence of parameters on the experiment. So it is necessary to tune the parameters. What's more, by constructing this system under small-scale dataset, we improve engineering ability.