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:
Yingting Huang, Fangyuan Ding
and Doudou Yang

Supervisor:
Qingyao Wu

Student ID：201720145136,
201720144979 and 201720145129

Grade:
Graduate

December 27, 2017

Recommender System Based on Matrix Decomposition

Abstract—This experiment let us program to achieve a recommender system based on matrix decomposition. During the solving process, we use two methods for optimization, they're ALS and SGD.

I. Introduction

According to this experiment, we have a command of the construction of recommender system, understand the principle of matrix decomposition and are familiar to the use of gradient descent. In this experiment, we construct a recommendation system under small-scale dataset,cultivate engineering ability.

II. Methods and Theory

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

Recommender systems has several advantages :

- Improve conversion rate: Help customers find a product she/he wants to buy.
- Cross-selling: Suggest additional products.
- Improve customer loyalty: Create a value-added relationship.

Make automatic predictions(filtering) about the interests of a user by collecting preferences or taste information from many other users(collaboration), which is called Collaborative Filtering.

There are two types of CF algorithms: Memory-based CF and Model-based CF. This experiment utilized the model-based CF called Matrix factorization.

Matrix Factorization

- Give a rating matrix R $\in R^{m*n}$, with sparse ratings from m users to n items.
- Assume rating matrix R can be factorized into the multiplication of two low-rank feature matrices P $\in R^{m*k}$ and Q $\in R^{k*n}$.

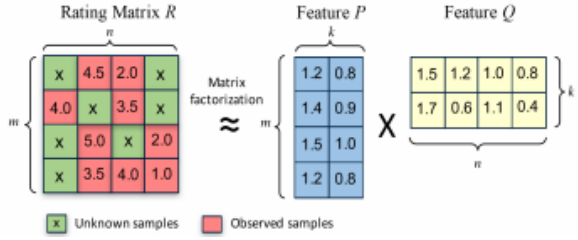


Figure 1 matrix factorization

- Determine an objective function.

Squared error loss:

$$\mathcal{L}(r_{u,i}, \hat{r}_{u,i}) = (r_{u,i} - \hat{r}_{u,i})^2$$

There are two methods for MF, and they are ALS and SGD. They are introduces as follows.

Algorithm 1 General Steps of ALS

1: **Require** rating matrix $\mathbf{R}$, feature matrices $\mathbf{P}$, $\mathbf{Q}$ and regularization parameter λ.
2: **Optimize** $\mathbf{P}$ while fixing $\mathbf{Q}$.
3: **Optimize** $\mathbf{Q}$ while fixing $\mathbf{P}$.
4: **Repeat** the above processes until **convergence**.

Figure 2 steps of ALS

Algorithm 3 General Steps of SGD

1: **Require** feature matrices $\mathbf{P}$, $\mathbf{Q}$, observed set Ω, regularization parameters λ_p, λ_q and learning rate α.
2: **Randomly** select an observed sample $r_{u,i}$ from observed set Ω.
3: Calculate the **gradient** w.r.t to the objective function.
4: **Update** the feature matrices $\mathbf{P}$ and $\mathbf{Q}$ with learning rate α and gradient.
5: **Repeat** the above processes until **convergence**.

Figure 3 steps of SGD

III. Experiment

- Experimental Step

The experiment code and drawing are both completed on jupyter.

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 $R_{nusers,nitems}$ against the raw data, and fill 0 for null values.
2. Initialize the user factor matrix $P_{nusers,K}$ and the item (movie) factor matrix $Q_{nitems,K}$, where K is the number of potential features.
3. Determine the loss function and the hyper parameter 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**.

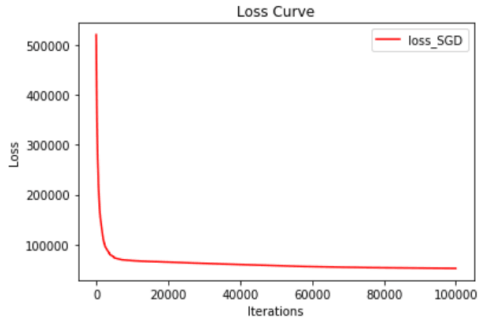
6. The final score prediction matrix $\hat{R}_{nusers,nitems}$is obtained by multiplying the user factor matrix $\hat{P}_{nusers,K}$and the transpose of the item factor matrix $Q_{nitems,K}$.

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 $R_{nusers,nitems}$ against the raw data, and fill 0 for null values.
2. Initialize the user factor matrix $P_{nusers,K}$and the item (movie) factor matrix$Q_{nitems,K}$, where K is the number of potential features.
3. Determine the loss function and hyperparameter learning rate η and the penalty factorλ.
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 $P_{nusers,K}$and $Q_{nitems,K}$;
   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 $\hat{R}_{nusers,nitems}$ is obtained by multiplying the user factor matrix $\hat{P}_{nusers,K}$ and the transpose of the item factor matrix $Q_{nitems,K}$.

- Experimental Result

Our final result, or $L_{validation}$ **curve with varying iterations** is depicted as follows:



IV. CONCLUSION

From the $L_{validation}$ **curve with varying iterations** we can clearly learn that the loss is gradually decreasing as iterations increases and the gradient of loss function tends to 0 at one certain iteration, keeping unchanged later. This statisfies what we expected and what is in fact, so our experiment is successful basically.