华南理工大学

South China University of Technology

Recommender System Based on Matrix Decomposition

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Guoshi Lin Zhiyuan Lin and Zanjie Fang

Supervisor:
Mingkui Tan

Student ID:
201530612217 201530612323 and
201530611456

Grade:
2015

January 7, 2018

Recommender System Based on Matrix Decomposition

***Abstract*—A recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item.One approach to the design of recommender systems that has wide use is collaborative filtering.Collaborative filtering methods are based on collecting and analyzing a large amount of information on users behaviors, activities or preferences and predicting what users will like based on their similarity to other users.**

I. Introduction

RECOMMENDATION systems have become increasingly popular in recent years, and are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general.Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past.A particular type of collaborative filtering algorithm uses matrix factorization, a low-rank matrix approximation technique.

II. Methods and Theory

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:

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}} \tag{1}$$

In this way, each row of $\mathbf{P}$ would represent the strength of the associations between a user and the features. Similarly, each row of $\mathbf{Q}$ 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:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj} \tag{2}$$

Now, we have to find a way to obtain $\mathbf{P}$ and $\mathbf{Q}$. One way to approach this problem is the first intialize the two matrices with some values, calculate how 'different their product is to $\mathbf{M}$, 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:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 \tag{3}$$

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:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij} q_{kj} \tag{4}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij} p_{ik} \tag{5}$$

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

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj} \tag{6}$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik} \tag{7}$$

Here, α is a constant whose value determines the rate of approaching the minimum.

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:

$$E = \sum_{(u_i, d_j, r_{ij}) \in T} e_{ij} = \sum_{(u_i, d_j, r_{ij}) \in T} (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 \tag{8}$$

III. Experiments

A. Dataset

1.Utilizing MovieLens-100k dataset.

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

user id	item id	rating	timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

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.

4. You can also construct train set and validation set according to your own evaluation method.

B. Implementation

The process of the experiment is:

- 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 $\mathbf{R_{user,item}}$ against the raw data, and fill 0 for null values.
- 2.Initialize the user factor matrix $\mathbf{R_{user,K}}$ and the item (movie) factor matrix $\mathbf{R_{item,K}}$, where K is the number of potential features.
- 3.Determine the loss function and the hyperparameter learning rate η and the penalty factorλ.
- 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 $\mathbf{P}$ and an item factor matrix $\mathbf{Q}$, Draw a curve with $L_{validation}$ varying iterations.
- 6.The final score prediction matrix $\mathbf{\hat{R}_{user,item}}$ is obtained by multiplying the user factor matrix $\mathbf{R_{user,K}}$ and the transpose of the item factor matrix $\mathbf{R_{item,K}}$

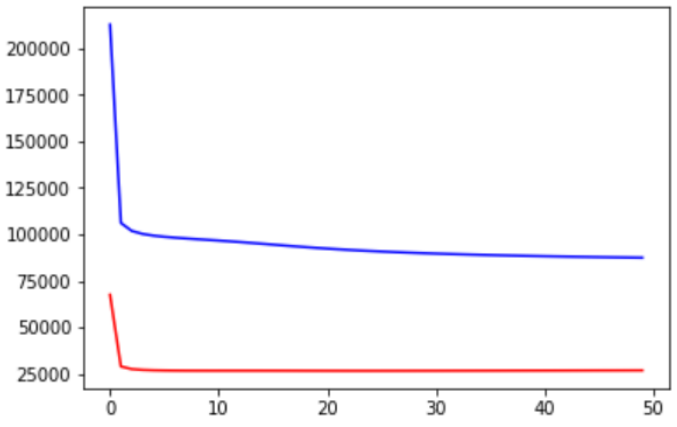
C. Initialization Method

TABLE I
PARAMETERS INITIALIZATION

Learning Rate	0.005
Iterations	50
Penalty Factor	0.1

D. Experiment result

The result of this experiment is represented by drawing the curve of $L_{validation}$ and the training loss(The red line is $L_{validation}$,the blue line is the curve of training loss):



IV. CONCLUSION

1.Recommender systems typically produce a list of recommendations in one of two ways through collaborative filtering or through content-based filtering .Collaborative filtering approaches build a model from a user's past behavior as well as similar decisions made by other users. This model is then used to predict items or ratings for items that the user may have an interest in.

2. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself.

3.But,Collaborative filtering approaches often suffer from three problems: cold start, scalability, and sparsity:

- Cold start: These systems often require a large amount of existing data on a user in order to make accurate recommendations.
- Scalability: In many of the environments in which these systems make recommendations, there are millions of users and products. Thus, a large amount of computation power is often necessary to calculate recommendations.
- Sparsity: The number of items sold on major e-commerce sites is extremely large. The most active users will only have rated a small subset of the overall database. Thus, even the most popular items have very few ratings.