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:
Zhishang Zhou, Fankai Xu, Li Kang

Supervisor:
Mingkui Tan and Qingyao Wu

Student ID:
201720144931, 201721046012,
201721046005

Grade:
Graduate

December 29, 2017

Recommender System Based on Matrix Decomposition

***Abstract*—In this experiment report, I will show you how Matrix Decomposition works in Recommender System. 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. There are many ways to implement a recommender system, while in this time, we will construct the system based on matrix decomposition. In part 1, I will briefly talk about the whole procedure we implemented. In part2, I will show Recommender System and Matrix Decomposition in detail. Finally, in part 3, I will show you complete experimental setup and experimental results to comfirm you the power of Adaboost algorithm.**

I. Introduction

RECOMMENDER 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. There are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners (online dating), and Twitter pages.

Recommendations can be generated by a wide range of algorithms. 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.

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.

In the following sections, we will go through the important ideas and the mathematics of matrix factorization.

II. Methods and Theory

Recommender System: Historically, people have relied on recommendations and mentions from their peers or the advice of experts to support decisions and discover new material. They discuss the weeks blockbuster over the water cooler, they read reviews in the newspapers entertainment section, or they ask a librarian to suggest a book. They may trust their local theater manager or news stand to narrow down their choices, or turn on the TV and watch whatever happens to be playing.

These methods of recommending new things have their limits, particularly for information discovery. There may be an independent film or book that a person would enjoy, but no one in their circle of acquaintances has heard of it yet. There may be a new indie band in another city whose music will likely never cross the local critics radar. Computer-based systems provide the opportunity to expand the set of people from whom users can obtain recommendations. They also enable us to mine users history and stated preferences for patterns that neither they nor their acquaintances identify, potentially providing a more finely-tuned selection experience.

There has been a good deal of research over the last 20 years on how to automatically recommend things to people and a wide variety of methods have been proposed. Recently, the Recommender Systems Handbook was published, providing in-depth discussions of a variety of recommender methods and topics. This survey, however, is focused primarily on collaborative filtering, a class of methods that recommend items to users based on the preferences other users have expressed for those items.

In addition to academic interest, recommendation systems are seeing significant interest from industry. Amazon.com has been using collaborative filtering for a decade to recommend products to their customers, and Netflix valued improvements to the recommender technology underlying their movie rental service at $1M via the widelypublicized Netflix Prize.

The capacity of computers to provide recommendations was recognized fairly early in the history of computing. Grundy, a computerbased librarian, was an early step towards automatic recommender systems. It was fairly primitive, grouping users into stereotypes based on a short interview and using hard-coded information about various sterotypes book preferences to generate recommendations, but it represents an important early entry in the recommender systems space.

Collaborative Filtering Methods: In the early 1990s, collaborative filtering began to arise as a solution for dealing with overload in online information spaces. Tapestry was a manual collaborative filtering system: it allowed the user to query for items in an information domain, such as corporate e-mail, based on other users opinions or actions (give me all the messages forwarded by John). It required effort on the part of its users, but allowed them to harness the reactions of previous readers of a piece of correspondence to determine its relevance to them.

Automated collaborative filtering systems soon followed, automatically locating relevant opinions and aggregating them to provide recommendations. GroupLens used this technique to identify Usenet articles which are likely to be interesting to a particular user. Users only needed to provide ratings or perform other observable actions; the system combined these with the ratings or actions of other users to provide

personalized results. With these systems, users do not obtain any direct knowledge of other users opinions, nor do they need to know what other users or items are in the system in order to receive recommendations.

In a word, Collaborative filtering (CF) is a popular recommendation algorithm that bases its predictions and recommendations on the ratings or behavior of other users in the system. The fundamental assumption behind this method is that other users opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the active users preference. Intuitively, they assume that, if users agree about the quality or relevance of some items, then they will likely agree about other items if a group of users likes the same things as Mary, then Mary is likely to like the things they like which she hasnt yet seen.

Matrix Factorization: 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.

Matrix factorization is to, obviously, factorize a matrix, i.e. to find out two (or more) matrices such that when you multiply them you will get back the original matrix. The intuition behind using matrix factorization to construct Recommender System is that there should be some latent features that determine how a user rates an item. For example, two users would give high ratings to a certain movie if they both like the actors/actresses of the movie, or if the movie is an action movie, which is a genre preferred by both users. Hence, if we can discover these latent features, we should be able to predict a rating with respect to a certain user and a certain item, because the features associated with the user should match with the features associated with the item.

Formally, suppose we have a set U of users, and a set D of items. Let $\mathbf{R}$ 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 $\mathbf{P}$ (a $|U| \times K$ matrix) and $\mathbf{Q}$ (a $|D| \times K$ matrix) such that their product approximates $\mathbf{R}$:

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

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:

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

Now, we want to minimize the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair:

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

We calculate the gradient to find the direction where we have to modify the values of p_{ik} and q_{kj}:

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

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

Having obtained the gradient, we can use a learning rate α, say 0.0002, to step towards the minimum.

Further more, we can introduce regularization to avoid overfitting. This is done by adding a parameter β and modify the squared error as follows:

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^{K} (||P||^2 + ||Q||^2)$$

In other words, the new parameter β is used to control the magnitudes of the user-feature and item-feature vectors such that P and Q would give a good approximation of R without having to contain large numbers. In practice, β is set to some values in the range of 0.02. The new update rules for this squared error can be obtained by a procedure similar to the one described above. The new update rules are as follows.

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

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

III. Experiments

A. Dataset

In this work, we utilize $MovieLens - 100k$ dataset, which 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

TABLE I
MovieLens-100k

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

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. In this experiment, we select u1.base / u1.test as our training data / testing data.

B. Implementation

Using stochastic gradient descent method(SGD):

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. Repeat step 4. several times, get a satisfactory user factor matrix and an item factor matrix , Draw a curve with varying iterations. The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix .

1. Use u1.base / u1.test as our training data / testing data. Populate the original scoring matrix $R_{n_{users},n_{items}}$ against the raw data, and fill null values.
2. Initialize the user factor matrix $P_{n_{users},K}$ and the item (movie) factor matrix $Q_{n_{items},K}$, where K is the number of potential features. Here we set K equals 10.
3. Determine the loss function and hyperparameter learning rate η and the penalty factor λ. Here we set $\eta = 0.0002$ and $\lambda = 0.02$, the loss function is shown as Section 2 wrotes.
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 $Q_{n_{items},K}$;
   - 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. 1000 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 $Q_{n_{items},K}$.

C. Results

As Fig. 1 shows, the $L_{validation}$ curve with varying iterations decrease rapidky due to SGD method. When we run out 1000 iterations, the object value seems can be smaller with more iterations.

As Fig. 2 shows, 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 $Q_{n_{items},K}$.

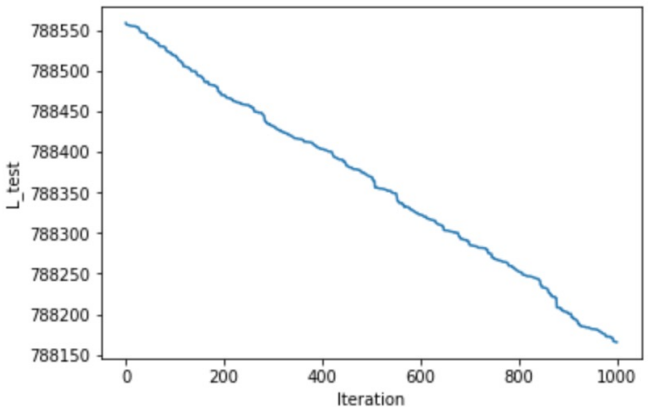

Fig. 1. $L_{validation}$ curve.

```
In [4]: hatR = numpy.dot(P, Q.T)

In [5]: hatR

Out[5]: array([[  3.94884439,   1.16673789,  -1.63331888, ...,  -0.57582477,
                  0.55763809,  -2.91401188],
               [ -2.66715982,  -1.5287109 ,  -0.99068579, ...,   5.59773491,
                 -2.86836905,   3.30592781],
               [  5.82514802,   5.16322775,   0.77880314, ..., -11.77511308,
                  3.43662043,  -4.88787886],
               ...,
               [  9.58552765,  -0.68187423,   2.20862905, ...,   3.36550079,
                  2.10581087,  -0.77189423],
               [  4.61076569,  -0.76806756,  -0.34113421, ...,   3.27444084,
                 -1.54904875,  -0.10175768],
               [ -0.72115502,   0.07676991,   1.67799372, ...,  -8.19947101,
                 -2.60792422,   1.62214911]])
```

Fig. 2. The final score prediction matrix $\hat{R}_{n_{users},n_{items}}$.

IV. Conclusion

In this experiment, we construct a small Recommender System based on Matrix Decomposition. We talk about the details of Recommender System and Collaborative Filtering Methods. The procedure and the mathematics of Matrix Factorization is also illustrated. We implemented a matrix factorization method and using SGD to achieve Collaborative Filtering Methods. The experiment results proved that we did a good job. After achieve this system, we not only get a well understand about the relative knowledge. but also learned a lot about collaboration. Finally, thanks all you effort in designing those good problems!