EE 239AS
Special Topics in Signals and Systems
Project 3
Collaborative Filtering
Winter 2016

Liqiang YU, Kaiming WANG and Jun FENG
904592975, 504592374, 304588434

03-04-2016

Contents

1 Introduction

In this project, we tried to build a movie recommendation system based on the collaborative filtering algorithm. This method based on the fact that there existed some other users who has the similar behaviors so we can use them to make the prediction for a specific target user. First some data preprocessing steps were taken to create the rating matrix. Then we implemented different matrix factorization algorithms to retrieve two factor matrices and get the prediction matrix. The prediction result were measured with 10-fold cross validation and the trade off curve between precision and recall. Finally, we evaluated the effect of our recommendation system by changing the number of movies we want to recommend.

The report is organized as follows: In section 2, we introduce the dataset we use briefly data preprocessing steps.In section 3, we discussed how to use weighted non-negative matrix factoration to predict missing data and methods to evaluate the results. In section 4, we discussed the weighted non-negative matrix factorization with regularization parameters and how to implement it with the alternating least squares algorithm. And we repeat the evaluation methods to compare results with the previous parts. In section 5, we evaluated the recommendation system with the precision when recommending top 5 movies. Moreover, by changing the number of movies to recommend, we plot the curve between hit rate and false alarm rate.

2 Data Preprocessing

In this project, we use MovieLens data sets, which were collected by the GroupLens Research Project at the University of Minnesota. This data set consists of 100,000 ratings from 1 to 5 from 943 users on 1682 movies. So we use the Import Data tool of MATLAB to transfer raw data file into a 100,000*4 matrix and four columns are userId, itemId, rating and timestamp, respectively. Use the first three columns, we can achieve a 943*1682 matrix R, $R(i, j)$ represent rating of user i on item j.

3 Weighted Non-Negative Matrix Factorization

Since we only have 100,000 ratings in the data sets, there are many missing ratings in matrix R, which is fulfilled by NaN values. In order to predict these values, we can employ non-negative matrix factorization to get matrices U, V such that $R_{m\times n} = U_{m\times k}V_{k\times n}$. It is necessary to calculate the least square error and minimize it.

This can be implemented by putting 0s where the data is missing and creating a weight matrix to calculate the squared error. Assume that the weight matrix $W_{m\times n}$ contains 1 in entries where we have known data points and 0 in

Table 1: The Least Square Error with Different K and Factorization Iteration

k	10	50	100
Iteration=50	65422.3843	38416.6458	26216.834
Iteration=100	60709.7768	30553.0711	17190.6811
Iteration=200	57633.8853	25012.4859	11590.326
Iteration=500	55607.3742	21087.0429	7732.6324
Iteration=1000	54398.9678	19426.6643	6169.5815
Iteration=2000	52557.7469	17968.6925	5228.23

entries where the data is missing. At last, we can formulate the above problem as:

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} (r_{ij} - (UV)_{ij})^2$$

Luckily, we do not need to implement this factorization by hand. Instead, we can use *wnmfrule* function in the Matrix Factorization Toolbox in MATLAB. By choosing the k equal to $10, 50, 100$, the total least squared error is shown in table 4. Furthermore, we found that under different iterations, we may have different performance. We may find the total least square error become smaller when k and iteration rise.

3.1 10-fold Cross-validation

As before, we will use cross-validation in our recommendation system design. We will divide 100,000 records into 10 folds exclusively. Each time, we use 9 folds as trainset and remaining 1 fold as testset. However, we will calculate average absolute error over testing data among all entries this time, not previous total least squared error as in section 1. At this time, we choose k to be 100, and set factorization to be 50, 100, 200, 500, 1000 and 2000 to get Average absolute error over testing data for each entry of all 10 tests, Highest average absolute error over testing data for each entry and Lowest average absolute error over testing data for each entry. The result is shown by table2, so we can draw the conclusion that in this part, we should choose a suitable iteration to get the best absolute error. In order to illustrate this phenomenon, we try to calculate absolute error within low iterations and the result is shown in table3. It seems that according to k=100, we should not use too high iterations for matrix factorization.

3.2 Precision Over Recall

According to testing data, we can assume that if a user has rated a movie 3 or lower we conclude they didn't like the movie, and if a user has rated a movie 4 or higher they have liked it. However, when it comes to predicted data, it is

Table 2: Absolute Error over Testing Data under Different Iteration

	Average	Highest	Lowest
Iteration=50	0.8547	0.86669	0.84256
Iteration=100	0.90612	0.92049	0.89578
Iteration=200	0.97256	0.99214	0.95611
Iteration=500	247.925	2469.2667	1.0425
Iteration=1000	115.8072	515.373	1.1092
Iteration=2000	65.2333	312.8415	1.1706

Table 3: Absolute Error over Testing Data under Low Iteration

	Average	Highest	Lowest
Iteration=10	0.80078	0.81206	0.79389
Iteration=20	0.80965	0.82013	0.79624
Iteration=30	0.82235	0.83746	0.81402
Iteration=40	0.84093	0.8543	0.82794
Iteration=50	0.85362	0.87073	0.84427
Iteration=60	0.86266	0.87285	0.84627

our job to set the threshold to decide whether users like or dislike items.

Out of all predicted entries in which user likes the item, the percentage of the user actually like the item is precision. While out of all entries in which user actually likes the item, the percentage entries which we have predicted successfully is recall.

From the previous sections, we knew that both k and iterations hve impact on our prediction performance, so in figure1 and figure3, we show this relations. It seems that since we only have a small amount of data, we had better use small k and fewer iterations.

4 Weighted Non-Negative Matrix Factorization with Regularization

In the previous section, we talked about how to make recommendation based on the weighted non-negative matrix factorization. In this part, we replaced rating matrix with weight matrix and vice versa. However without any regularization parameter, the prediction matrix would all be 1. Therefore, we add some regularization terms to the cost function. The new version cost function

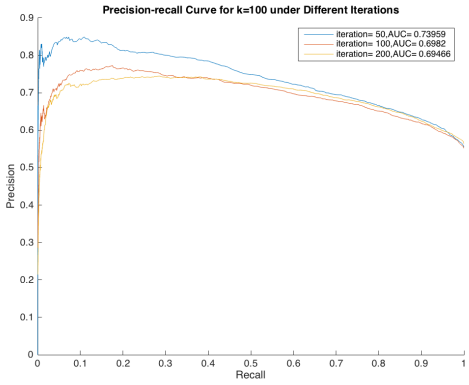

Figure 1: Precision-recall Curve for k=100 under Different Iterations

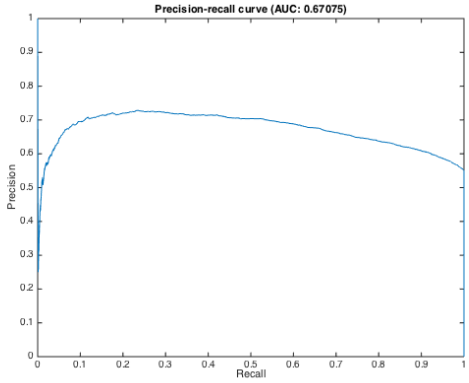

Figure 2: Precision versus Recall

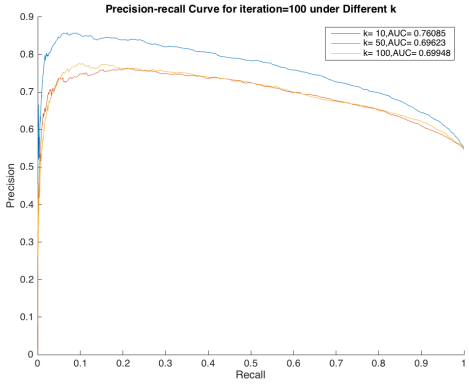

Figure 3: Precision-recall Curve for iteration=100 under Different k

Table 4: The total square error with different K

k	10	50	100
total square error	19.2809	23.4241	41.6349

is as follow:

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij}(r_{ij} - (UV)_{ij})^2 + \lambda \left(\sum_{i=1}^{m} \sum_{j=1}^{k} u_{ij}^2 + \sum_{i=1}^{k} \sum_{j=1}^{n} v_{ij}^2 \right)$$

4.1 The Reason of Regularization

In the first part of weighted non-negative matrix factorization with regularization, we consider the problem that we use the rating matrix as the weight and set R into a 0-1 matrix and no regularization is applied. Formation of this problem is as below:

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij}(r_{ij} - (UV)_{ij})^2 \tag{1}$$

The algorithm to solve it is almost exactly like the one we used in the previous problem. And the only difference here is we exchange the placement of R and W. Thus, we are actually reconstructing a 0-1 matrix. And we calculate the total squared error to evaluate the performance of this algorithm under 3 different setting of k=10,50 and 100. The first thing we notice here is the total squared error is much smaller than before. This is mainly because the reconstructed matrix in this two problems are different and we suppose the 0-1 matrix is much easier to rebuilt. Another strange attribute of this result is that when k is larger the total squared error is also larger, which is exactly opposed to the result of previous one. The solution of this is that the optimal of this optimization problem is a matrix with all 1 entries and actually when k is small there are less entries in each matrix and the constraints between them are rather loose. Thus,it is easier to get an result approximate to the optimal one, which comes along with a better result of total squared error.
However, this does not mean the prediction is better. Since what we indeed want to achieve is a matrix whose element value is positively correlated to the rating. And that cannot be done by introducing the regularization terms.

4.2 Regularized Version of Alternating Least Squares

In the alternating least squares algorithm that is implemented in many recommendation system, we need to construct a binary matrix P

$$P = \begin{cases} 1, R > 0 \\ 0, R = 0 \end{cases}$$

Then we want to factorize P into X and Y such that $P \approx XY^T$. The recommendations are largest values in XY^T. Since optimizing X and Y simultaneously is non-convex, alternating least squares idea was used, for the reason that if X or Y is fixed, it's just a system of linear equations, which is convex and easy to solve. The solving processes are as follows:

1. Initialize Y with random values
2. Solve for X
3. Fix X, solve for Y
4. Repeat above processes until it converges

Let's define the regularization weights $c_{ui} = r_{ui}$, where the subscripts are for user u and movie i, and define C_u as the diagonal matrix of c_u. Then the update equation for x is

$$x_u = (Y^T C_u Y + \lambda I)^{-1} Y^T C_u p_u$$

4.3 Evaluation of the Results

We used the same evaluation methods to test the result of the regularized ALS. The precision VS recall curve with k=10,50 and 100 is shown in figure 4, figure 5 and figure 6,respectively. And in each figure we plot three curves which corresponding to λ=0.01,0.1 and 1. Finally, we calculate the area under curve(AUC) of them and since the curves in the same figure is closed to each other, we calculate the average AUC of them. Finally the AUC for these 3 figures are 0.6257,0.6423,0.6333,respectively.

From this figures, we see the tradeoff between precision and recall. In each graph, the precision reaches its maximum around recall equals to 0 and then gradually decreases while recall approaches 1.

5 Creating the Recommendation System

The precision of our recommendation system depended on the prediction matrix P and how many movies you want to recommend. When choosing top 5 movies, the precision in the 10-fold cross validation is shown in figure 7 and the average precision is 84.07%.

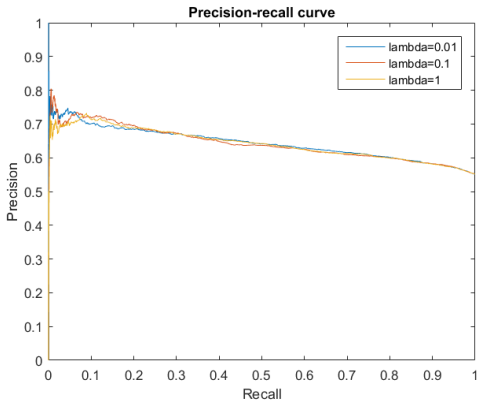

Figure 4: Precision recall curve with k=10

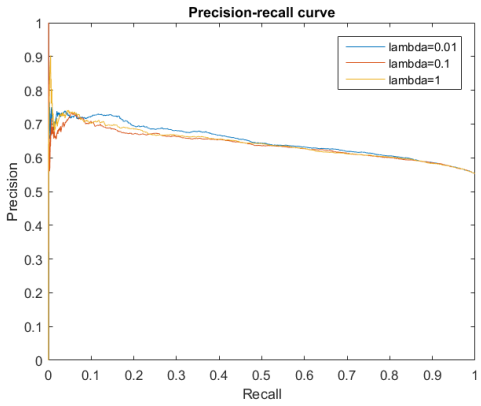

Figure 5: Precision recall curve with k=10

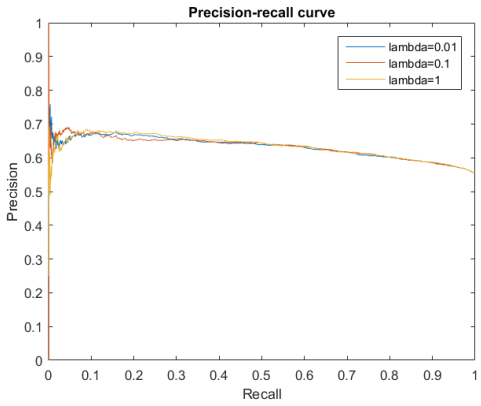

Figure 6: Precision recall curve with k=100

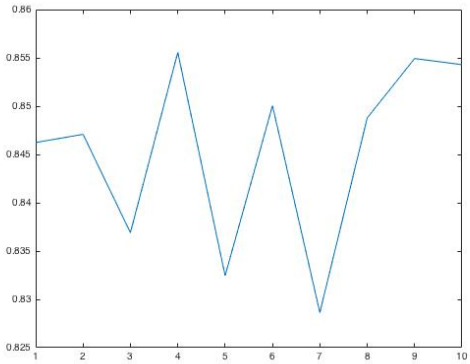

Figure 7: The precision over 10-fold cross validation

The hit rate and false alarm rate can change dramatically with different number of recommendations. The results is shown in figure 8. From the figure we can see, at the beginning the hit rate increased dramatically with the increasing of the recommendations. After some point, the false alarm rate increased more rapidly than the hit rate, which means the number of recommendation may not be larger than 20.

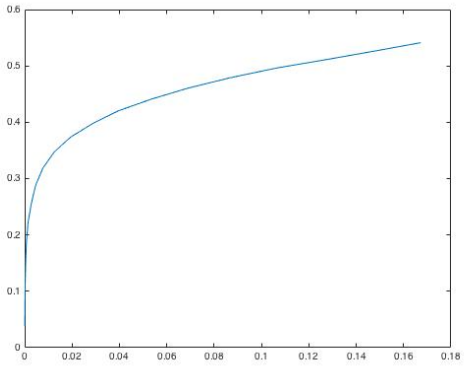

Figure 8: Hit rate VS false alarm rate