Project 3 Collaborative Filtering

Group member: Huang Xinxin 104589081
Lu Qiujing 704617222
Niu Longjia 304590762

Question 1

Description: Create a matrix named R containing user ratings with users on rows and movies on columns. We want to run a matrix factorization job to get matrices U and V such that $Rm\times n\approx Um\times kVk\times n$ in known data points. For this purpose, we calculate matrices U and V such that the squared error is minimized:

$$\min \sum_{known\ i,j} \left(r_{ij} - (UV)_{ij}\right)^2$$

This can be implemented by putting 0 s where the data is missing and creating a weight matrix to calculate the squared error. Assume that the weight matrix $Wm\times n$ contains 1 in entries where we have known data points and 0 in entries where the data is missing. Then if R contains 0 in missing entries, we can formulate the above problem as:

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

Choose k equal to 10, 50, and 100. What is the total least squared error in each case？

Solution:
We use matlab to finish this task. We implement the recommended Matrix Factorization Toolbox to calculate U and V, as well as the total least square error. After carefully test, we find that the results remain essentially unchanged, when the number of iteration larger than 1000. So we use the default 1000 times' iteration. The results are as follows:

Table 1-1 Total Least Square error

k	10	50	100
Total LS Error (1.0*e+4)	5.4388	1.9685	0.6168

The larger k means that more information about R is conserved. As we can see, the larger k is, the more precise the approximation is, which follows our cognition.

Question 2

Description: Train your system as in part 1 and find the average absolute error over testing data (prediction error $|Rpredicted - Ractual|$) What is the average error? (Average among 10 the tests for each system and for each test among all entries) What are the highest and lowest values of average error among the 10 tests? (Average over the testing entries in each test and find which of the 10 tests has highest or lowest average error) Test the recommendation system we designed by a "10-fold Cross-validation".

Solution:
Results:

Table 2-1 Test errors

	Avg Error	highest error	lowest error
k=10	294.6965	2923.1	0.8056
k=50	931.2582	8856.5	1.0738
k=100	1.2396	1.8	1.1084

Table 2-2 Training errors

	Avg Error	highest error	lowest error
k=10	0.5635	0.5652	0.5604
k=50	0.2942	0.2962	0.2915
k=100	0.1372	0.1390	0.1360

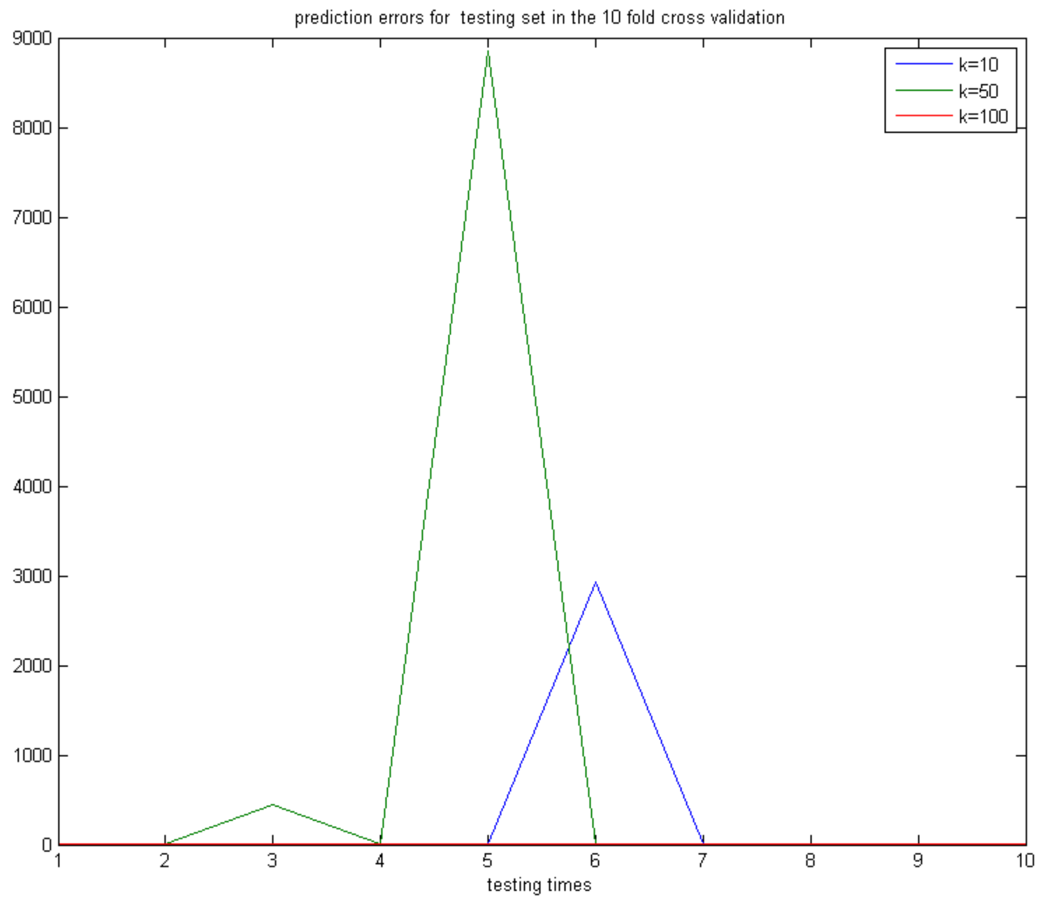

Fig 2-1 10 fold cross validation for testing set

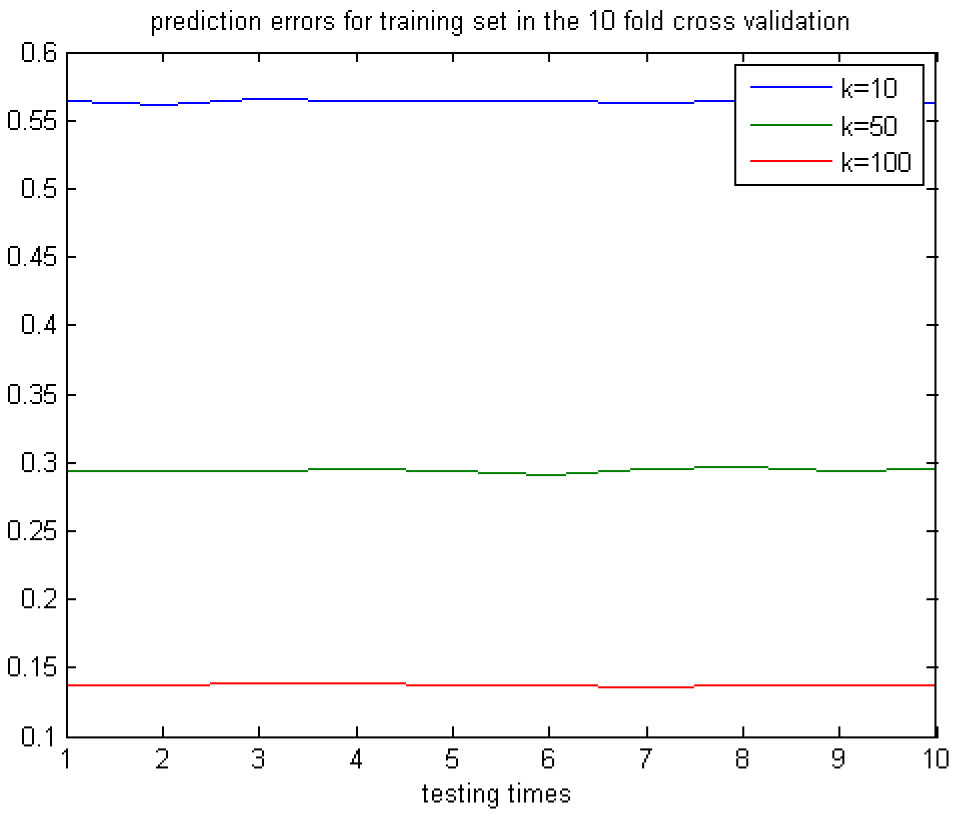

Fig 2-2 10 fold cross validation for training set

As shown above, the test error changes greatly for k=50, and k=50 has the highest 'highest test error'' although it has lower training errors. This means the factorization can ensure the training errors' minimization, but not the ability of prediction. K=100's performance is the steadiest. It has the lowest average error in the test set and in the training set, but its computation takes the longest time.

In comparison with the test and training result, we can conclude that the k=50 is the worst factorization since its testing errors are always high, which means the fail in finding the most important, representative features. So in this way, features are not always the more the better. Comparing k=10 and k=100, k=10 has higher testing error but it is acceptable if the access of computation is limited.

Question 3

Description: Now 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. Based on the estimated values in the R matrix in parts 1 and 2, we can use a threshold on the estimated entry to predict if the user will like the movie or not. Plot the precision over recall curve for different values of the threshold for the designed recommendation system.

Solution:

We first introduce the notation used in this part. P denotes the number of entities actual belongs to positive (like) class. N denotes the number of entities actual belongs to negative (unlike) class. TP, TN, FP and FN defined as follows.

Number of entities	Actual in P	Actual in N
Predicted in P	TP	FP
Predicted in N	FN	TN

At first, we explore the whole dataset, if the threshold is 4; the positive and negative entities are evenly distributed. P is roughly equal to N. So the positive and negative points are balanced in both training and testing set. In our situation, correctly finding the movies that people like is more important than finding the one people don't like. We don't pay much attention to how our model performance on negative class. So precision-recall curve will better reflect the performance of our model than ROC.
According to the definition of precision and recall,

$$precision = TP/(TP + FP)$$
$$recall = TP / (TP + FN)$$

In our experiment, since the threshold of real value remains 4, so the number of entities actual belongs to positive (like) class remains unchanged in one CV. When threshold equal to 1, TP+ FP is around 9900, where the total number of test set is 10000. That means almost all the entities are classified to the positive class, including the ones that are actual in positive class. In this condition, recall roughly equal to one and TP and FP roughly equal to P and N, which means precision approximate to P/P+N. So now precision can reflect the actual ration between two classes, which is around 1 in our experiment. With our threshold getting larger, the number of predicted entities belongs to positive class is getting fewer. That is to say, TP+FP are getting smaller. Both TP and FP are smaller. Recall is getting smaller. We cannot predict precision, since both the numerator and denominator are decreasing. As we can see from the following figure, the PR curve is concave.

In our experiment, we use different thresholds on estimated values to predict if they like a movie or not. For the real value, the threshold remains 4, which means if a user rates a movie larger or equal to 4, the user like the movie? We also test different k, with k = 10, 50, 100. We change threshold from 1 to 5 with step = 0.1 and use 10-crossfold validation to evaluate our system. Since there are many results, we will not display all the values in detail. We plot the Precision - recall (PR) curve and calculate AUC-PR, the area under the PR curve.

Table 3-1 Area under PR curve with different k

k	10	50	100
AUC-PR	0.7166	0.5839	0.5655

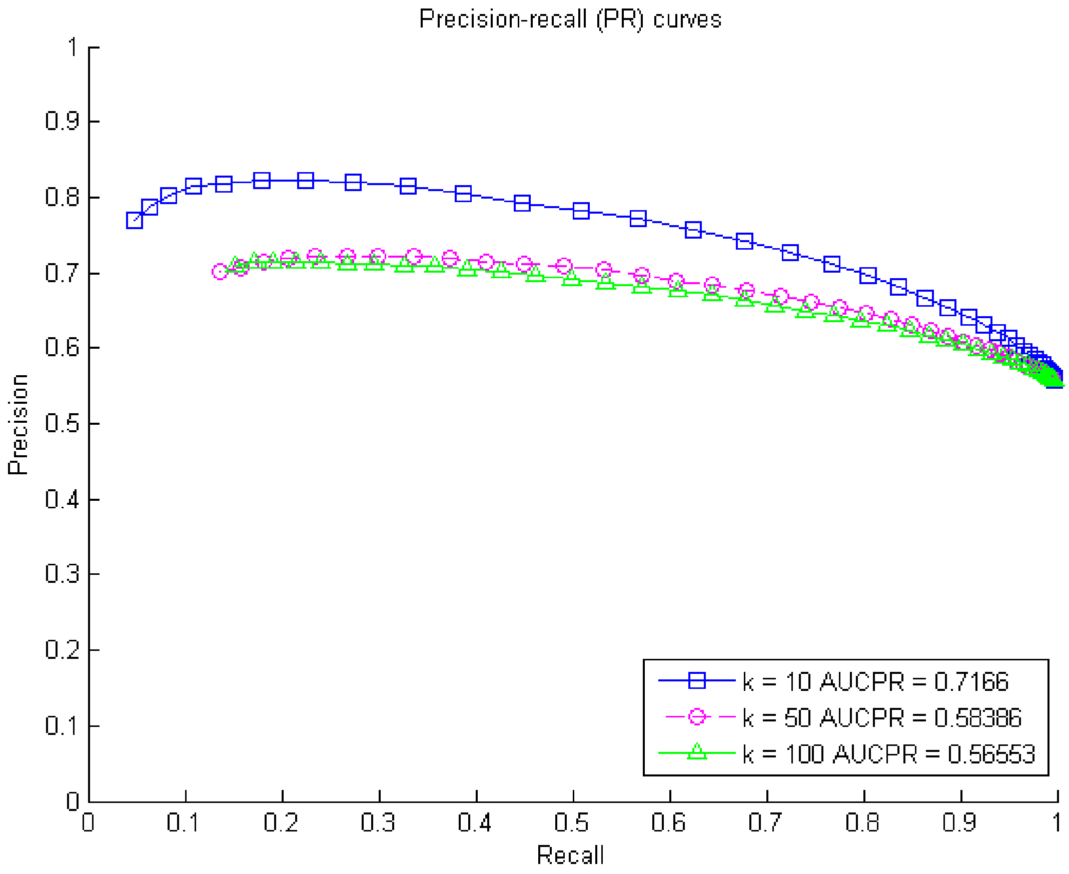

Fig 3-1 Precision-Recall (PR) Curve

According to the results above, we find that when k equals to 10, our model performs better than the other two cases. This is out of expectation. In our opinion, we can consider k as the number of 'features' we pick in classification. And when k = 10, we pick 10 most important features and ignore the others in our experiment, 10 features are enough to cover important relations in R. When k = 50 and 100, we may involve noise and get over fitted. So we will choose k = 10 in our recommendation system.

It is hard to decide the tradeoff between recall and precision just by the figure. We introduce F1-score, which is a useful statistical analysis of classification models, to evaluate our model. F1-score is calculated by

$$F1 = 2 * precision * recall/(precision + recall)$$

It is a weighted harmonic mean of the precision and recall. We will pick the threshold that gives the largest F1-score as our final threshold.

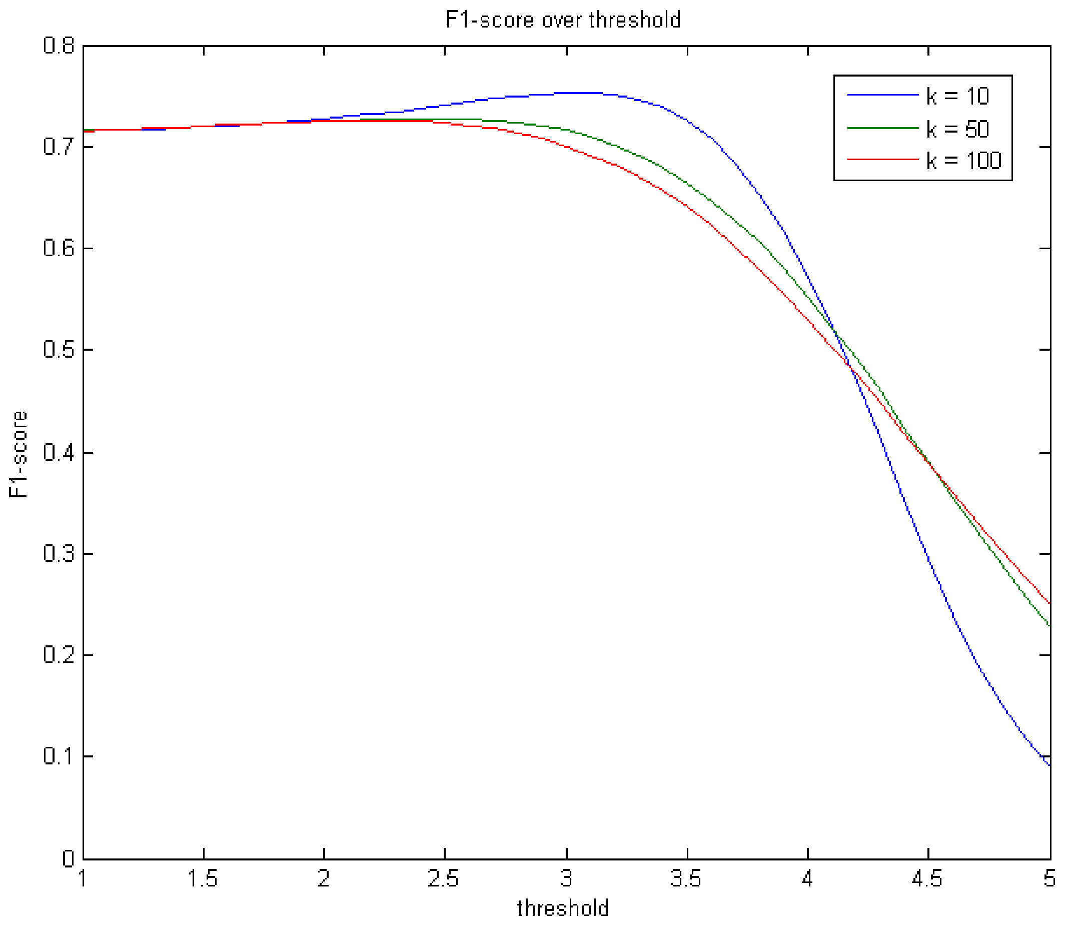

Fig 3-2. F1 score over threshold

We can see that different k corresponds to different largest f1-score, as well as different best threshold.

Table 3-2 Best F1- score and its corresponding threshold

k	Threshold	F1-score
10	3.0	0.7530
50	2.6	0.7275
100	2.2	0.7256

So if we build a recommendation system based on method in part 2), we can choose k = 10 and threshold = 3.0 for predicted values.

Question 4

Description and solution:

1. Try changing the weight matrix and use rating values as weights, instead of 1, for the known data points. Turn R into a 0-1 matrix where rij=1 for known ratings and rij=0 for unknown entries. Run matrix factorization again with the same values of k as in part 1.

k	error in question 1	error for the weighted algorithm
10	233.3303	1.3219
50	139.9123	3.0960
100	78.8815	4.9270

There is an obvious change in the total squared errors. This is due to the weighted objective function put more constraints on the higher-rated errors, so the records whose rates are higher are more likely to get accurate predictions. In this perspective, the squared error will fall down because of the smaller errors for high values. In addition, putting small weights on low rate relaxes the equal constraints on every data. Since the matrix is sparse, the relaxation makes it possible to get better approximation, exploring true hidden features.

2. According to the reference, the weighted matrix factorization can be computed by iterations. If we want to approximate X with weight W using matrix UV, it can be achieved by this updating computation.

$$\mathrm{U} \leftarrow \mathrm{U} \cdot \frac{(W \cdot X)V^T}{(W \cdot XV)V^T} \qquad \mathrm{V} \leftarrow \mathrm{V} \cdot \frac{(W \cdot X)U^T}{(W \cdot XU)U^T}$$

Furthermore, if adding regularized term into the optimization steps, the formula can be changed into

$$\mathrm{U} \leftarrow \mathrm{U} \cdot \frac{(W \cdot X)V^T}{(W \cdot XV)V^T + \lambda U} \qquad \mathrm{V} \leftarrow \mathrm{V} \cdot \frac{(W \cdot X)U^T}{(W \cdot XU)U^T + \lambda V}$$

The Precision- recall curves are as follows

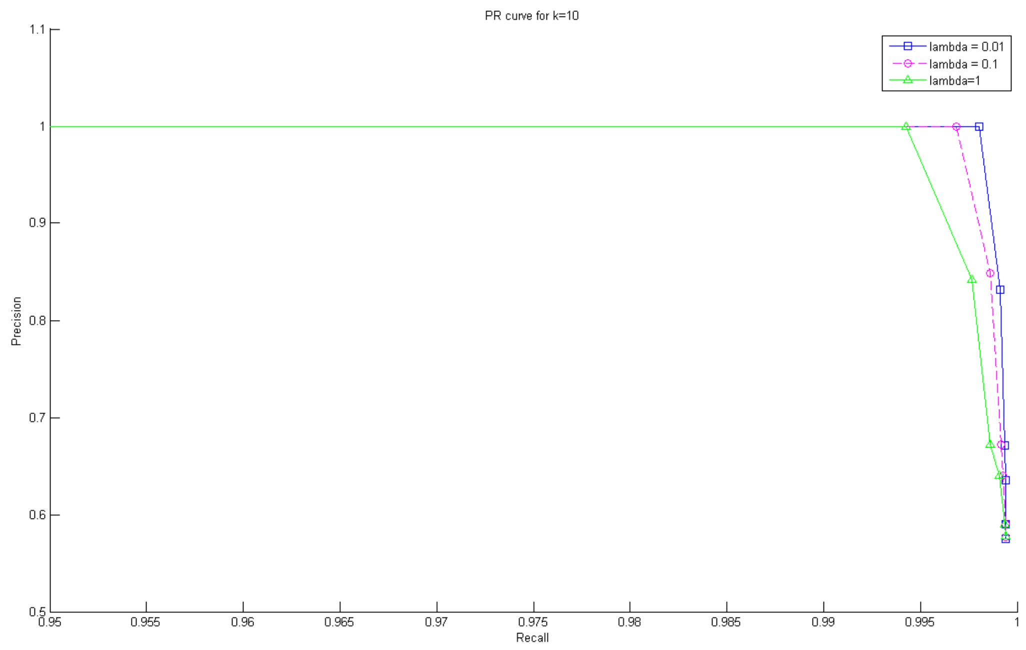

Fig. 4-1 PR curve with k = 10

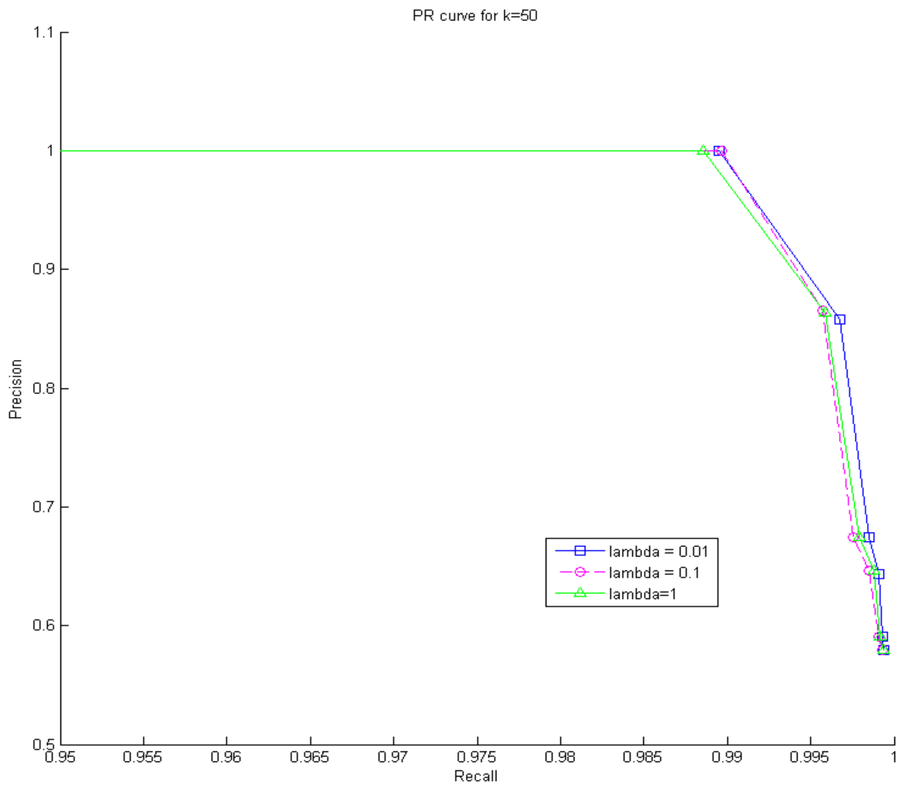

Fig. 4-2 PR curve with k = 50

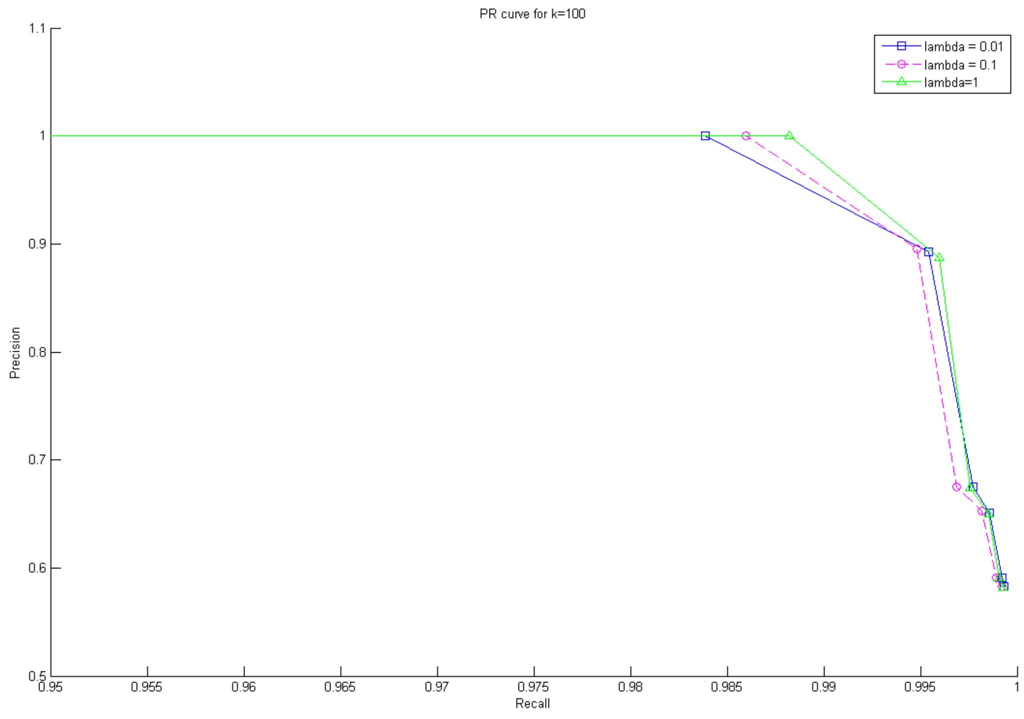

Fig. 4-3 PR curve with k = 100

First of all, all three curves show that the weighted matrix factorization is a good way to approximate the matrix. Since the precision is very high, we zoom into (0.95, 1, 0.5, and 1.1) to analyze the specific performance. For k=10 and 50, as shown in the picture, the increase of lambda causes early drop of the curve. According to the formula, we can observe that adding the regularized term will tend to shrink the matrix but the weighted error term tends to keep the predicted value of high rates correct. The contradictory force is controlled by the coefficient of regularized term. If we increase the shrinking force, our predicted results tend to be lower. So when threshold decreases, we can see an early drop in the precision.

Question 5 Movie Recommendation

Description: Convert R to a 0-1 matrix where the entries are 1 for available data points and 0 for missing data points. Use ratings as weights and run your algorithm to find the top *L* movies recommended to each user.

By using the cross-validation method, find out the **precision**, **hit rate** and **false-alarm rate**. Start from L=1 and while increasing L measure hit rate and false-alarm rate for different values of L. **Plot** these values as points in a two-dimensional space with hit rate on the y axis and false-alarm rate on the x axis.

Solution:

Confusion Matrix	Predicted No Top L	Predicted Yes Top L

Actual No Top L	TN	FP
Actual Yes Top L	FN	TP

The parameters are calculated as below:

Precision = TP/ (TP+FP)

Hit Rate = TP/ (TP+FN)

False-Alarm Rate = FP/ (TN+FP)

The recommendation step works as follows:

1. First we apply the regularized version of ALS algorithm training on the Rating matrix which has already been converted to {0, 1}, with the Weight matrix being original Rating matrix. And we get matrix U and V.
2. Reconstruct the Ratings matrix using Weight.*(U*V). Then, for each user, we sort the row vector and get the top L movies ratings and their index in the original Rating matrix and Reconstructed Rating matrix. If duplicate values exist, then we include them all.
3. For every set of training and testing matrix, we iterate through L and store the precision, hit-rate and false-alarm rate for each of them
4. Calculate averages and plot graphs.

The following tables and graph are tested given ALS iteration times=500, ALS k value=10, 10-fold cross validation.

Precision for different L:

Table 5-1 Precision under different L (Number of Top Rated Movies)

L	1	2	3	4	5	6	7	8	9	10
Precision (%)	**96.469**	**96.109**	**95.949**	**96.131**	**96.308**	**96.422**	**96.601**	**96.742**	**96.881**	**96.966**

Average precision: 96.4578%

Table 5-1 shows that our system achieves great precision rate of an average 96.4578%, which indicates good performance on prediction.

Hit-rate for different L:

Table 5-2 Hit-Rate under different L (Number of Top Rated Movies)

L	1	2	3	4	5	6	7	8	9	10
Hit-rate (%)	**33.330**	**42.975**	**49.938**	**55.200**	**59.258**	**62.628**	**65.418**	**67.811**	**69.928**	**71.796**

Table 5-2 shows that it gradually increases as L becomes larger. The reason is that when we encounter duplicate values in actual test Rating matrix, we include them all as the movies that the user actually likes. Unlike actual Rating matrix, the reconstructed Rating matrix contains decimal rating values rather than integral ones. Thus the actual-like movies set is larger than the predicted-like movies set, which will result in high precision rate but lower hit rate. However, as L becomes larger, the system will be less influenced by duplicate values in actual test Rating matrix.

False-alarm-rate for different L:

Table 5-3 False-Alarm-Rate under different L (Number of Top Rated Movies)

L	1	2	3	4	5	6	7	8	9	10
false-alarm-rate (%)	**0.445**	**0.756**	**1.072**	**1.378**	**1.524**	**1.749**	**1.926**	**2.105**	**2.267**	**2.455**

Table 5-3 shows that the false-alarm-rate goes up as L increases. This is reasonable because trying to predict more top movies will definitely amplify the errors of our model and bring out more prediction errors. However, our prediction model keeps the false-alarm-rate below 2.5%, which indicates great performance.

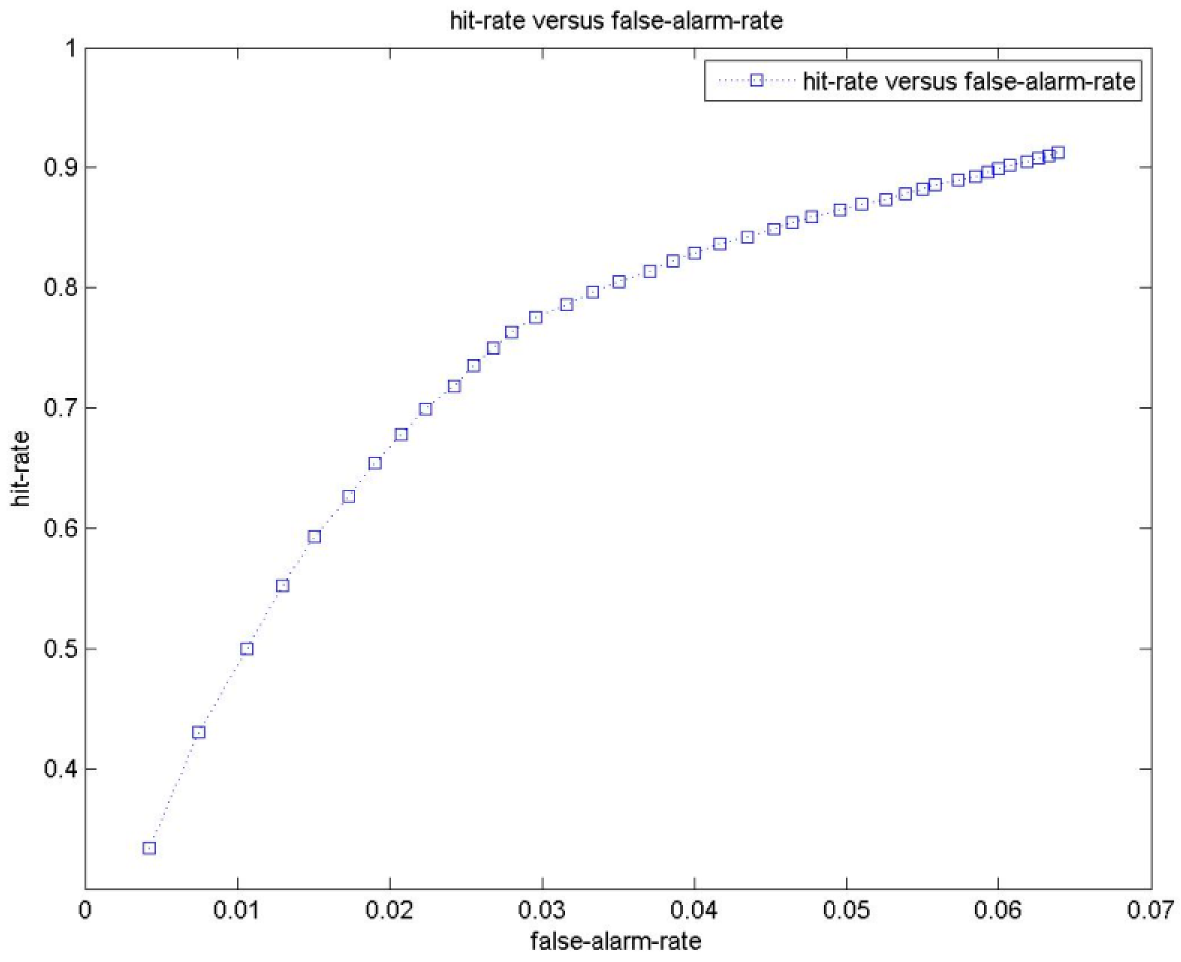

Figure 5-1 Hit-Rate Versus False-Alarm-Rate Under different L (Number of Top Rated Movies)

Figure 5-1 shows hit-rate versus false-alarm-rate for different values of L. The hit-rate and false-alarm-rate are highly correlated. As we increase the false-alarm-rate, we expect to see higher fraction of suggested movies not actually liked by the user, at the same time, more suggested movies actually liked by the user will also be introduced by our system, which will result in a hit-rate increase. Also, the above curve has a steep increase at the beginning but slows down its growth later on, which is due to the slowdown of hit-rate increase by including more top movies in our system.

Reference:

Ho N D, Van Dooren P, Blondel V. Weighted nonnegative matrix factorization and face feature extraction[J]. submitted to Image and Vision Computing, 2007.