EE 219 Project 3 - Report

February 27

Isha Verma	Heenal Doshi	Pranav Thulasiram Bhat
ishaverma@cs.ucla.edu	heenald@cs.ucla.edu	pranavtbhat@cs.ucla.edu
404761131	004758927	704741684

1 Introduction

In this project we built a recommender system for the Movie Lens dataset. Given a bunch of users, movies and the an incomplete list of ratings given by the users to the movies, we had to generate movie suggestions or recommendations for the users.

The main technique used was collaborative filtering. Collaborative filtering is a method that automatically predicts the interests of a user, based on the preference information for a large number of users. The basic idea is, that if two users have provided similar ratings to a few movies, they are likely to have more common interests.

The given input was a matrix, with the $UserID$ along the row axis and the $MovieID$ along the column axis. The cells contained the rating (1-5) for a given $UserID, MovieID$ pair. The matrix is relatively sparse, i.e. there a large number of cells for which no rating information is present.

The problem is therefore to learn from the input matrix and predict the values for cells containing zeros. The technique used, matrix factorization using alternating least squares, decomposes the input matrix into the product of two matrices.

$$R_{mn} = U_{mk}.V_{kn}$$

To account for missing values, we use a weight matrix W which contains 1 for cells containing a rating,

and 0 otherwise. The least square factorization used is :

$$min \sum_{known i,j} (R_{i,j} - (U.V)_{i,j})^2$$

2 Question 1

This question required us to factorize the input matrix for varying values of k (the number of latent features), and calculate the mean squared error in each case.

Since we decided to use Python for this assignment, we had to code our own matrix factorization function. The key idea here was to alternatively solve for the matrices U and V using the other as an optimization constraint. For initializing the matrix, we used the *lstsq* method from Python's *linalg* library. This allowed us to obtain a reasonably good solution in fewer iterations.

Our implementation basically minimized the error:

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} (R_{i,j} - (U.V)_{i,j})^2$$

Number of Latent Features	Mean Squared Error
10	56743.0229
50	24768.134
100	12074.423

Table 1: The mean squared error for matrix factorizations with different numbers of latent features

3 Question 2

In this part we had to perform 10-fold cross validation to test the model designed earlier. We performed this using the following stages:

- Compute the list of non-zero indices
- Generate a permutation of the non-zero indices using vector indexing.
- Divide the input into 10 training-testing set combinations and perform the matrix factorization.
- Calculate the testing error within each fold, and overall for the entire input.

In this question, we have reported the mean absolute error for each of the folds.

Number of Latent Features	Maximum CV error	Minimum CV error	Average CV error
10	0.78277	0.76292	0.77317
50	0.89995	0.87954	0.8896
100	0.90931	0.880823	0.89462

Table 2: The mean squared error for matrix factorizations with different numbers of latent features

4 Question 3

For this question we classified the data based on the threshold values where threshold value ranging from 1 to 5 in increments of 0.5. The predicted value lying above the threshold denotes that user liked the movie and if it lies below the threshold it denotes the user didnt like it.

The recall and precision were calculated using the following equations:

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

If the rating value predicted is greater than the threshold and greater than the actual testing rating value then it is regarded as true positive tp otherwise as false positive fp. Also, if the prediction is less than the threshold then it is regarded as false negative fn.

Threshold	Precision	Recall
1	1.0	0.99323932
2	0.95562474	0.97049638
3	0.90457909	0.82935251
4	0.80946861	0.44746664
5	0.5846902	0.08479658

Table 3: Precision and Recall for k = 10

Threshold	Precision	Recall
1	1.0	0.99222922
2	0.95686126	0.95429428
3	0.90033792	0.78490328
4	0.76400814	0.47623499
5	0.47491335	0.1823625

Table 4: Precision and Recall for k = 50

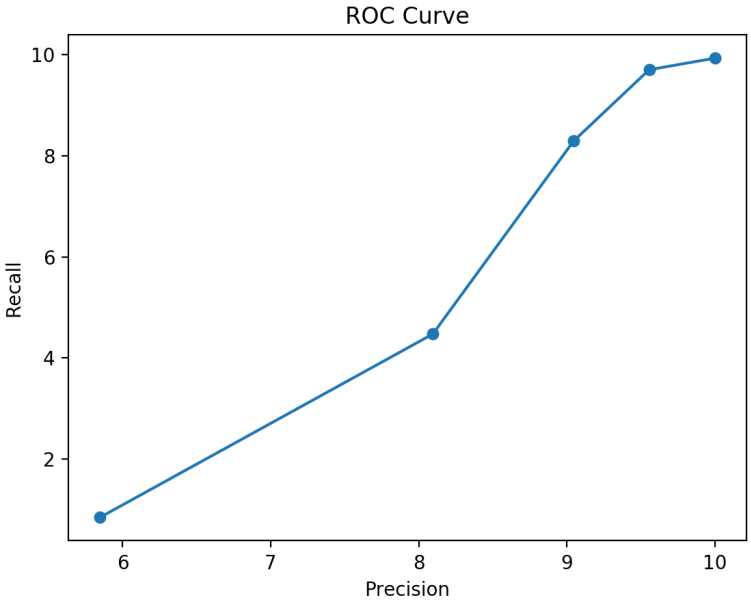

Figure 1: ROC curve for k = 10

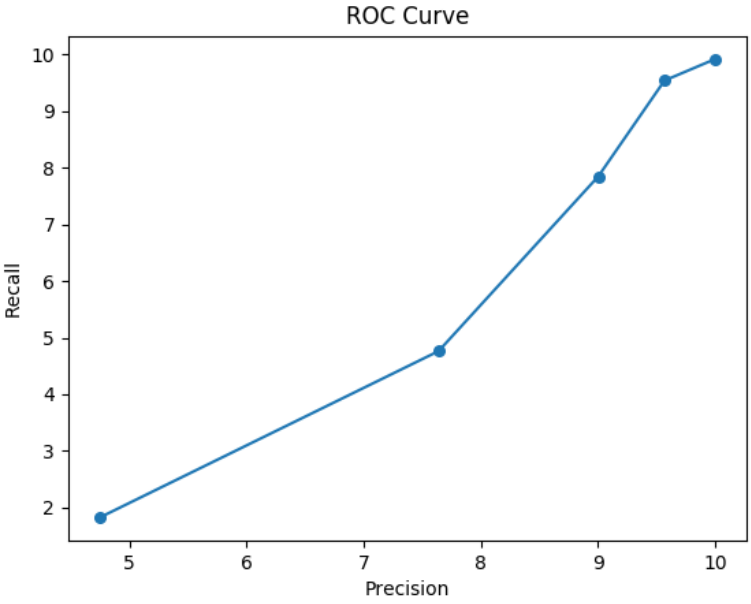

Figure 2: ROC curve for k = 50

Threshold	Precision	Recall
1	1.0	0.99316932
2	0.95716653	0.95664417
3	0.90134346	0.78555302
4	0.75616577	0.47885403
5	0.46446775	0.204831

Table 5: Precision and Recall for k = 100

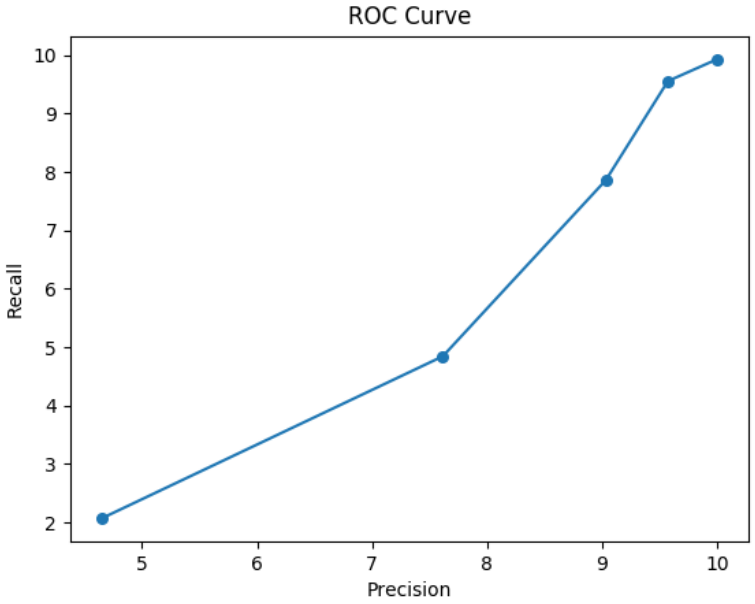

Figure 3: ROC curve for k = 100

5 Question 4

For this question in the first part we use rating matrix R as the weight matrix W and vice versa and calculated the squared error for different values of k = 10, 50, 100. The table below lists the squared error after factorizing for the different values of k.

We ran the matrix factorization with different values of = 0.01, 0.1, 1. We plot the ROC curve for all combinations of k and . The best precision vs recall is found to be with k = 100 and $\lambda = 1$.

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}(R_{i,j} - (U.V)_{i,j})^2 + \lambda(\sum_{i=1}^{m} \sum_{j=1}^{n} u_{ij}^2 + \sum_{i=1}^{m} \sum_{j=1}^{n} v_{ij}^2)$$

We ran the matrix factorization with different values of = 0.01, 0.1, 1. We plot the ROC curve for all combinations of k and . The best precision vs recall is found to be with k = 100 and = 1.

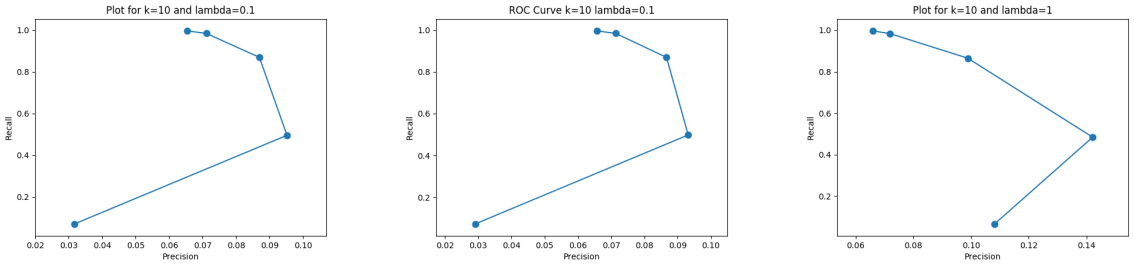

Figure 4: ROC curves for k = 10

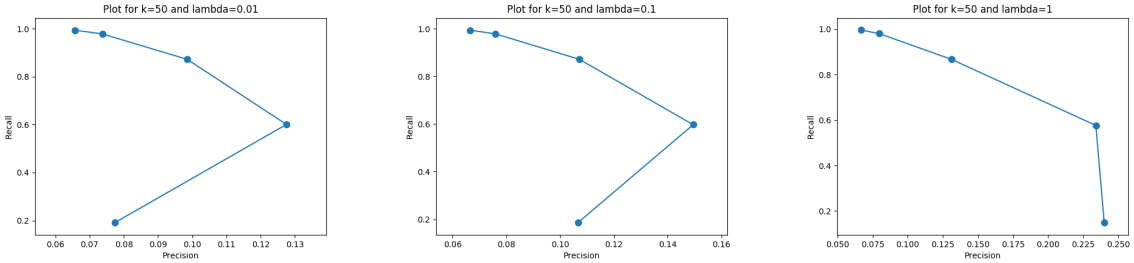

Figure 5: ROC curves for k = 50

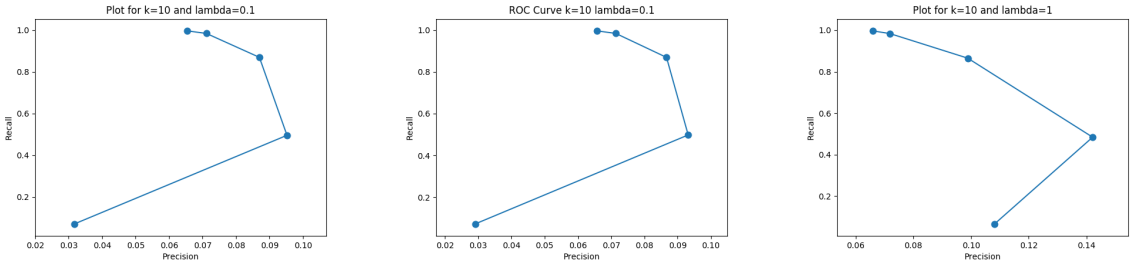

Figure 6: ROC curves for k = 100

K - Latent Features	λ	**Least Squared Error**
10	0.01	60078.0270
	0.1	60778.6993
	1	61242.1982
50	0.01	30723.5609
	0.1	31513.6674
	1	32681.2276
100	0.01	18175.8530
	0.1	18914.5025
	1	19865.1730

Table 5: Least Squared Error with regularization

Number of Latent Features	Least Squared Error
10	51.0337
50	269.6156
100	252.3031

Table 6: Least squared error for different values

6 Question 5

In this question the R matrix was created with binary values where 1 indicates rating is available and 0 when it is not. For the weight matrix we used the actual ratings the user had given. We predicted the matrix R and replaced the value by -1 if rating was not present. The ratings were sorted in descending order to obtain the top L movies for every user. These steps were done for every fold in cross validation. The value of L was varied from 1 to 5 and the precision for every value of L was calculated. The precision for L = 5 was obtained as 0.89595 when averaged out for all folds. The precision for all the values of L is in the table below.

L	Precision
1	0.88125
2	0.88928
3	0.88923
4	0.89435
5	0.89595

Table 7: L vs Precision

The hit rate and false alarm rate were then calculated. Hit rate was obtained by counting the number of predictions where the rating predicted was above the threshold of 3 and the false alarm rate was the fraction of ratings where the value was below the threshold. The plots show that hit rate increases as L is increased while the false alarm rate decreases. The plot of Hit rate vs False alarm rate is also shown below.

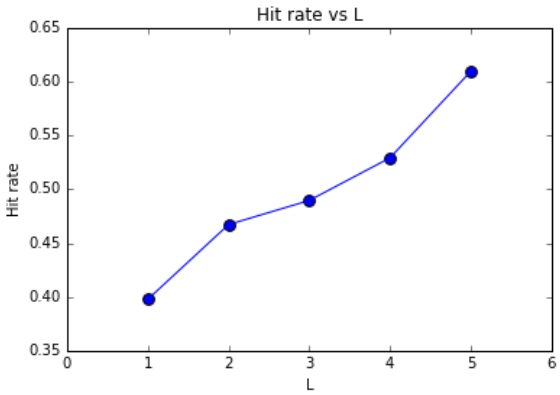

Figure 7: Hit rate vs L

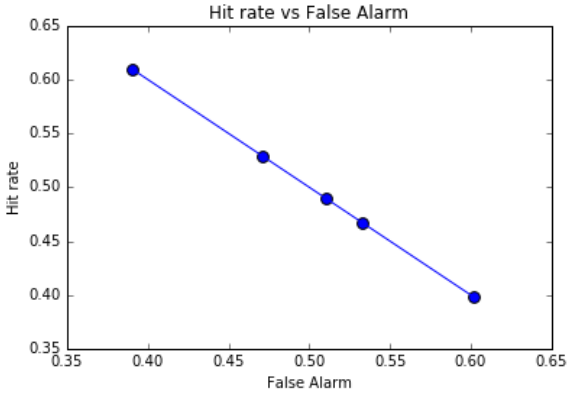

Figure 8: Hit rate vs False Alarm

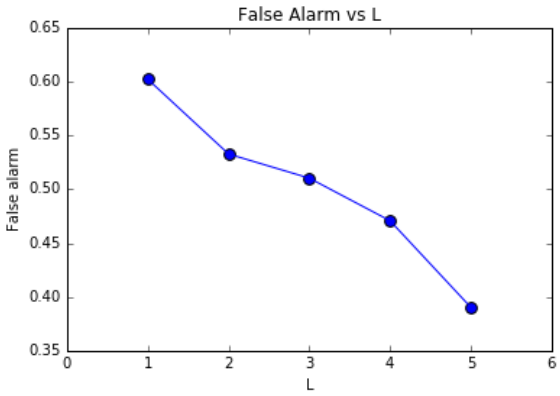

Figure 9: Hit rate vs False Alarm