239AS - Special Topics in Signals and Systems
Project 3 - Collaborative Filtering

Mansee Jadhav - 204567818
Mauli Shah - 004567942
Ronak Sumbaly - 604591897

March 4, 2016

Introduction

A collaborative filtering is a recommender system that refers to methods of predicting a user's opinion on an entity using other users' opinion. We predict the users with similar behavior as the target user. A rating matrix with $user_{id}$ as the rows and $movie_{id}$ as column is created along with a weight matrix W_{ij} by putting 0s for missing entries and 1 for known data points. We infer the unrated items by adopting the following construction: $R_{mn} = U_{mk} \cdot V_{kn}$ where the rows of U are the vectors that correspond to users and the columns of V characterize a particular item. The least square factorization used is :

$$min \sum_{known\ i,j} (r_{ij} - (UV)_{ij})^2 \tag{1}$$

$$\tag{2}$$

Question 1 - Least Square Factorization

We have defined a **nmf** function which takes input R matrix, latent features and maximum iterations for factorization. Matrix Factorization method generates two matrices $R_{mn} = U_{mk} \cdot V_{kn}$, such that each cell in R is generated by dot product of latent vector describing user and a latent vector describing item.

Intuitively, this product measures how similar these vectors are. During training you want to find "good" vectors, such that the approximation error is minimized. We considered entries from the prediction matrix that correspond to actual prediction the user actually made.

So on modification of eq.(1) we get :

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij}(r_{ij} - (UV)_{ij})^2 \tag{3}$$

We tested for various values of latent features K= 10, 50, 100 to optimize the sizes of U and V. Following were the results obtained:

K - Latent Features	Least Squared Error
10	60369.2282529
50	29754.4397803
100	17839.4962846

Table 1: Least Squared Error for different latent features

Question 2 - 10-fold Cross-Validation on Recommendation System

Now we test our recommendation system using **"10 - fold Cross validation"**. For doing this we randomly split our data into 90 percent training and 10 percent testing across 10 folds (unique test records taken for each fold). For predicting the values of our testing data, we set values in weight matrix to 0.

We use the partitioned training data R_{train} to factorize into U and V. We then validate the prediction values obtained from prediction matrix (dot product of U and V) against the actual rating in testing dataset across each fold.

Efficiency of our recommendation system is measured in terms of average absolute error over testing data $prediction\ error = |R_{predicted} - R_{actual}|$ Following are the results obtained for each fold.

Fold	Cross Validation Prediction Error
1	0.8259
2	0.8196
3	0.8314
4	0.8180
5	0.8260
6	0.8237
7	0.8184
8	0.8302
9	0.8220
10	0.8262

Table 2: 10 Fold Cross Validation Prediction Error

Cross Validation Average Error = **0.8241**
Highest Cross Validation Error = **0.8314**
Lowest Cross Validation Error = **0.8180**

Question 3 - Recommendation Systems with Threshold Limits

Based on the prediction done in above questions, we now classify data into two sets that is, if user likes a movie or if he does not like a movie based on the threshold values. If the predicted value is threshold that means the user likes the movie and vice versa.

We choose threshold values in range 1 to 5, with increments of 0.5. We use following quantities to define **Recall** and **Precision** of our classified data:

$$Precision = \frac{tp}{tp + fp} \tag{4}$$

$$Recall = \frac{tp}{tp + fn} \tag{5}$$

An element is classified as true positive *tp* if predicted rating value is greater than the threshold and also if it is greater than the actual testing rating value, else the element is classified as false positive *fp*. If the predicted rating value is less than threshold than the element is classified as false negative *fn*.

Threshold	Precision	Recall
1	1.0	0.9949
2	0.9574	0.9666
3	0.9058	0.8082
4	0.7830	0.4765
5	0.5208	0.1743

Table 3: Precision vs. Recall values based on Threshold

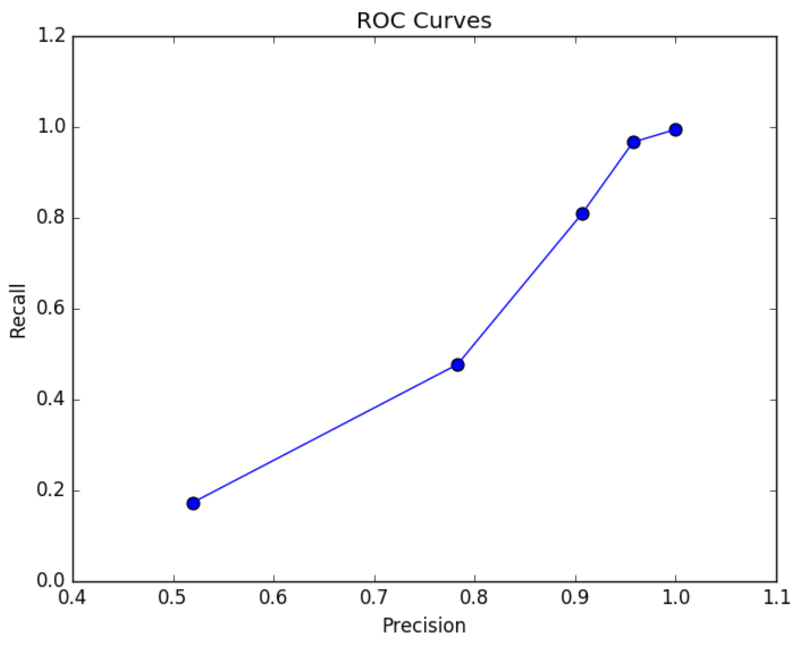

Figure 1: Precision vs Recall - ROC Curve for different threshold value

Question 4

Now in Part A of Ques. 4 we apply the same cost function as done in Ques. 1, but this time we reverse the roles of R and W matrices in the factorization step. We again applied our defined **nmf** function for different values of latent features K= 10, 50, 100 and measured the squared error. Below are the results obtained.

K - Latent Features	Least Squared Error
10	88.8867
50	194.3656
100	143.9701

Table 4: Least Squared Error for different latent features

We now modify the cost function to add a regularization term lambda λ. We choose values of lambda to be 0.01, 0.1 and 1.0. The precision and recall was calculated for all lambda values with respect to the latent features. The results and graphs are provided below. It is seen as the value of k and lambda increases the graph keeps on improving and the best curve is obtained for $k = 100$ and $\lambda = 1$

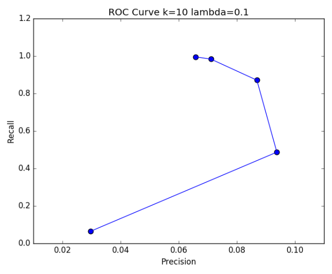

Figure 2: ROC Curve: k = 10 Lambda = 0.1

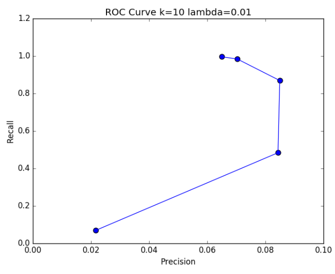

Figure 3: ROC Curve: k = 10 Lambda = 0.01

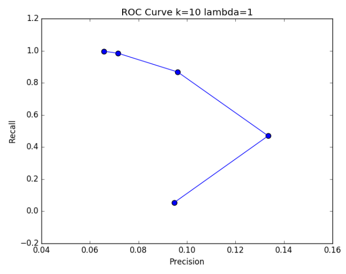

Figure 4: ROC Curve: k = 10 Lambda = 1

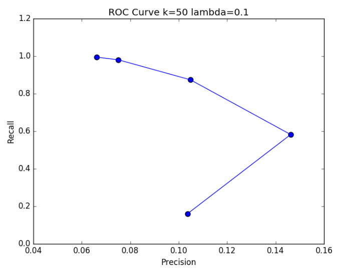

Figure 5: ROC Curve: k = 50 Lambda = 0.1

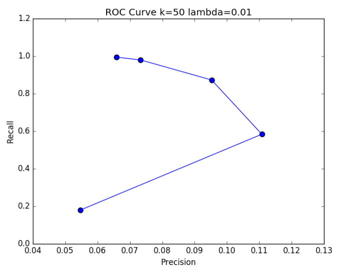

Figure 6: ROC Curve: k = 50 Lambda = 0.01

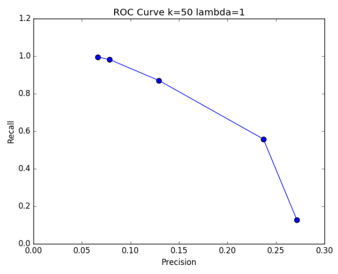

Figure 7: ROC Curve: k = 50 Lambda = 1

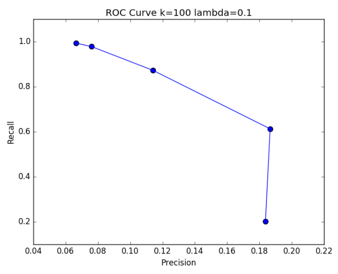

Figure 8: ROC Curve: k = 100 Lambda = 0.1

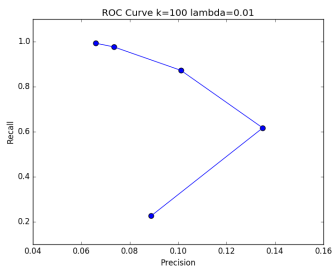

Figure 9: ROC Curve: k = 100 Lambda = 0.01

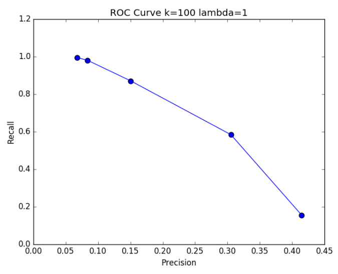

Figure 10: ROC Curve: k = 100 Lambda = 1

K - Latent Features	λ	**Least Squared Error**
10	0.01	60290.740
	0.1	60071.362
	1.0	60713.338
50	0.01	30630.763
	0.1	30563.810
	1.0	31791.308
100	0.01	17636.026
	0.1	17941.716
	1.0	19874.717

Table 5: Least Square error for different lambda and their respective Latent features

Question 5

For this part, first we find predicted R matrix by supplying R as a 0-1 matrix where 1 is when a rating is available and 0 otherwise. In this R predicted matrix we replace the ratings which are not present with -1 so that it is not considered in the further calculations. This is done keeping in mind that the predicted rating can be around 0 if the user has rated the movie very low. The next step is to sort the ratings for every user in the descending order to get the **top L movies for every user**.

The next step is to execute the above for every fold and finding the precision. The average precision obtained for all the folds for **L=5 is 0.8957**. While calculating the precision we ignore the movies whose ratings are not present in the actual R matrix i.e. by simply ignoring the ratings that we supplied -1 as the value. Next step is to calculate the Hit rate and False alarm rate.

Hit Rate is the movies recommended by the system which are liked by the user. For this, we calculate the number of movies in L for each user that has a value above threshold. This gives us the hit rate. On the other hand, the movies recommended by the system which are not liked by the user are counted as a false alarm. Thus any rating present in L falling below the threshold which indicates the user did not like it falls under this category. We got different values of Hit Rate and False Alarm Rate by increasing the L from 1 onwards. While calculating the values under L, we reuse the same R predicted matrix for that fold as refactorization is not required. Each L value is averaged over the folds to obtain an average Hit and False rates.

The obtained values were plotted as - **Hit-Rate on the Y axis and Miss-Rate on the X axis**.

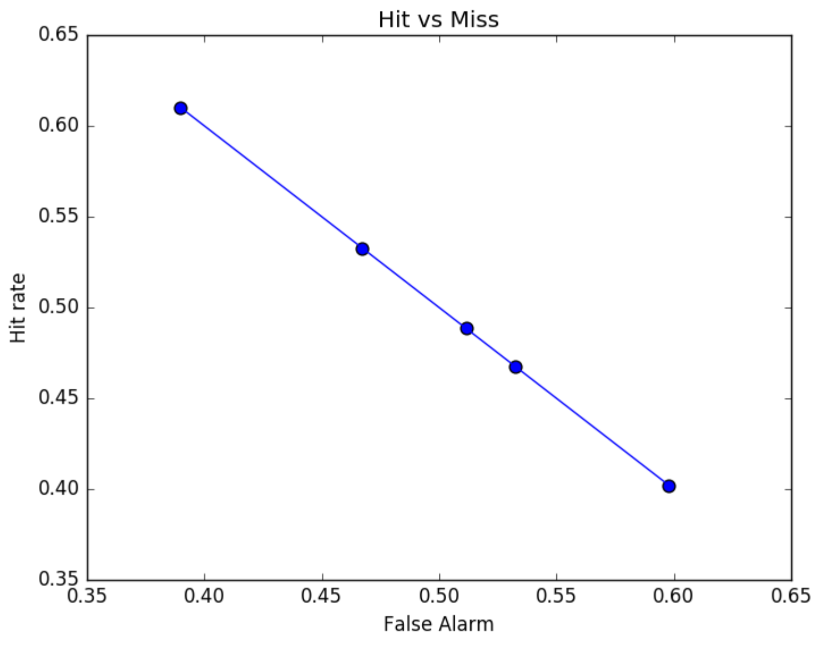

Figure 11: Precision vs Recall for different threshold values

Precision obtained for each value of L

L	Precision
1	0.8866
2	0.8913
3	0.8908
4	0.8936
5	0.8957

Table 6: Precision obtained for different values of L