EE219
Project 3 – Collaborative Filtering
Winter 2017

Prepared by-

Twinkle Gupta: 804740325
Shikhar Malhotra: 504741656
Omkar Patil: 904760474

"Collaborative filtering (CF) is a technique used by recommender systems. Collaborative filtering has two senses, a narrow one and a more general one. In the newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person. For example, a collaborative filtering recommendation system for television tastes could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average (non-specific) score for each item of interest, for example based on its number of votes.

In the more general sense, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. Collaborative filtering methods have been applied to many different kinds of data including: sensing and monitoring data, such as in mineral exploration, environmental sensing over large areas or multiple sensors; financial data, such as financial service institutions that integrate many financial sources; or in electronic commerce and web applications where the focus is on user data, etc." – *Wikipedia*

PART 1

The MovieLens dataset (http://grouplens.org/datasets/movielens/) has been provided which consists a total of 100K ratings. To load the entire matrix into MATLAB, a matrix R was created with rows referring to the user id and columns referring to the movie id. A single user could have rated more than 1 movie. The matrix R consisted of 943 rows and 1682 columns in total, referring to the number of users and movies respectively. The value of each cell of matrix R referred to the rating allotted to a movie by a particular user. If the user had not rated a certain movie, the cell

was labelled as NaN. The matrix R indeed had a large number of cells marked NaN, since all users didn't rate each and every movie.

Another matrix W is created with the same dimensions as R. The value W[i][j] was set to 1 if the the user *i* had rated movie *j*, otherwise it was rated NaN. This matrix was used to determine if a particular user had rated a certain movie or not.

Through this first part, the goal was to find U and V such that the squared error is minimized given by the following equation –

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

The function *wnmfrule*, was used from the Matrix Factorization Toolbox available in MATLAB. The given dimension values of K – 10, 50 and 100 were passed along with the matrix R to *wnmfrule.* The function then factorized the matrix R into two separate matrixes U and V returning a residual error along with it. The final problem is formulated as given by the equation -

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

The observed values for the factorization process with different values of K are as follows (for iterations – 150) –

K	Squared Error	Residual Error
10	5.9246e+04	2.4341e+02
50	2.7228e+04	1.6501e+02
100	1.3685e+04	1.1698e+02

Observation –

With the increase in K, the squared error keeps on decreasing. The residual error also keeps on decreasing. So, the best possible least squared error for 150 iterations was found at K = 100 i.e. 1.3685e+04.

PART 2

The 10-fold cross validation was performed using the *crossvalind* function of MATLAB. A random ordering of indices were generated in the range of 1 to 100000. As cross-validation is supposed to work, a part of the entire data was used as testing and the remaining bulk of data was

used for training the model. For 10-fold cross validations, $1/10^{th}$ of the data was being used for testing while the remaining $9/10^{th}$ was used for testing. To split the data, the random ordering of indices are used.

In this way, R and W was obtained. The R matrix was then passed with different values of K as input parameters to the *wnmfrule*. The output we get is the matrix factorization which is then compared with the testing data to obtain the absolute error for each fold. The following table summarizes the maximum, minimum and average error for different values of K.

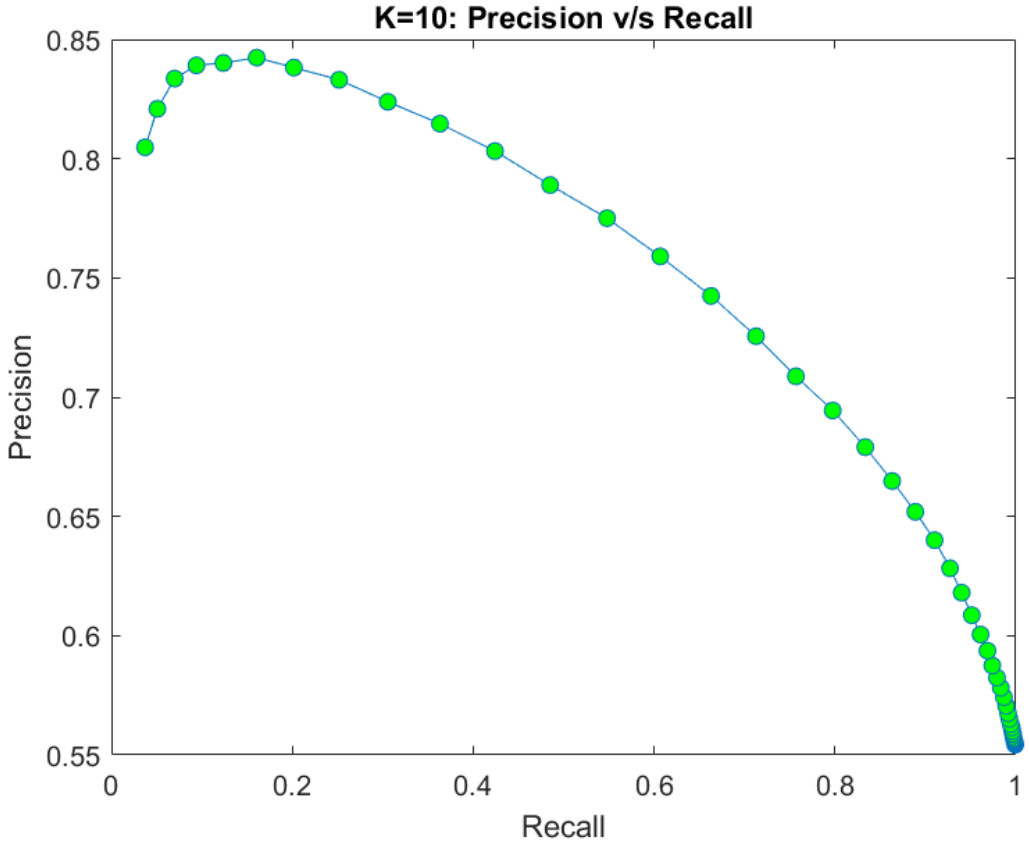
K	Maximum Error	Minimum Error	Average Error
10	1.242835e+04	7.824958e-01	1.250763e+03
50	9.603764e-01	9.137594e-01	9.324846e-01
100	9.578691e-01	9.144101e-01	9.420888e-01

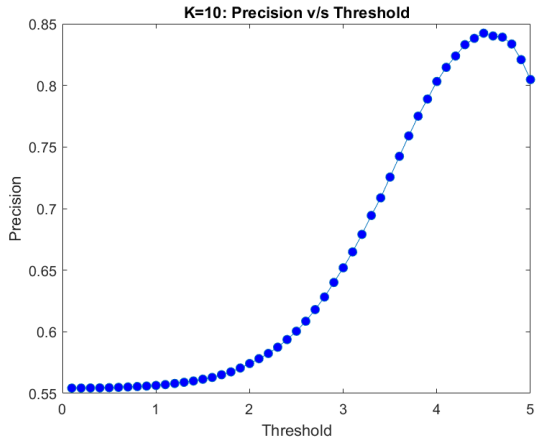
Observation –

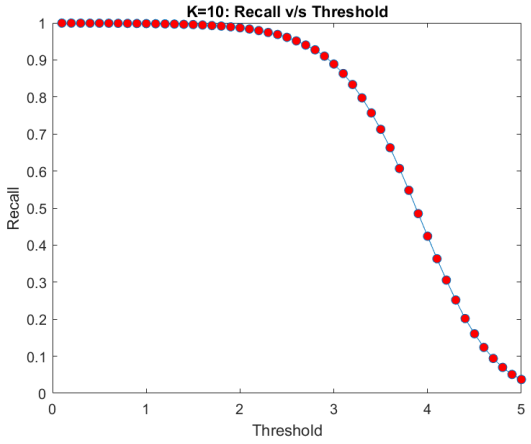
We can observe that the Minimum error keeps on increasing as the value of K is increased. On the other hand, the trend is opposite for Maximum error. There is a steady decrease in the values of this error as the value of K increases.

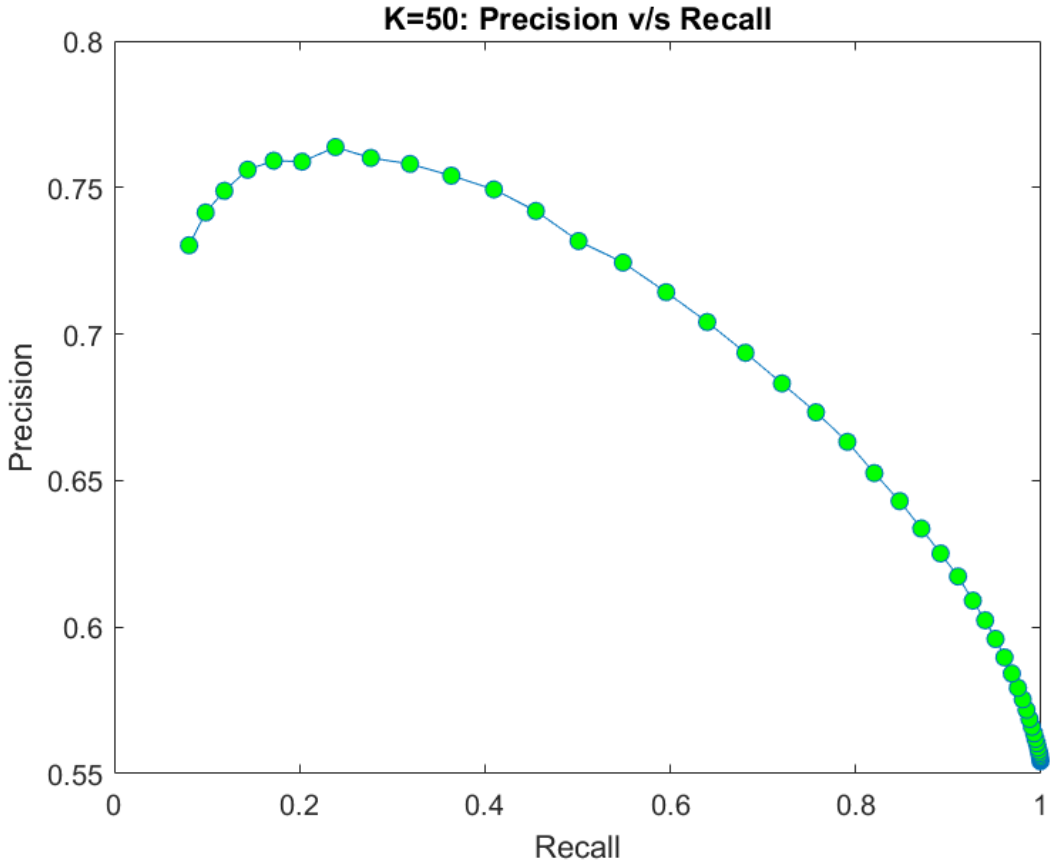
PART 3

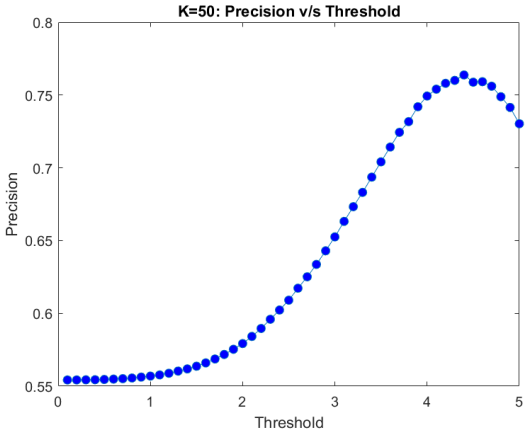
The objective for this part to find and plot the precision and recall of algorithm for different values of threshold and K (10, 50, and 100). Different values of threshold have been taken in the range of 0.1 to 5 for the prediction of ratings. For the actual ratings, a threshold of 3 is taken. For each value of K three different graphs have been plot – Precision v/s Recall, Precision v/s Threshold and Recall v/s Threshold.

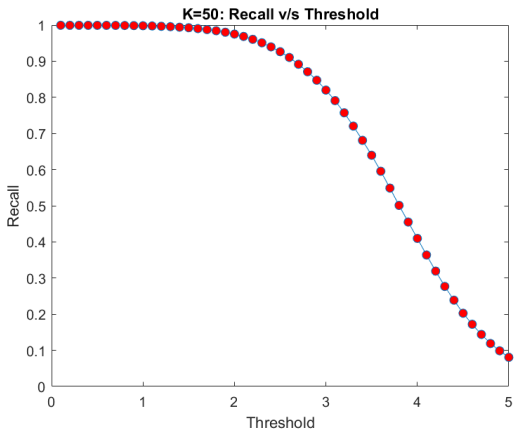

K=10: Precision v/s Recall
Precision
Recall
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0
0.2
0.4
0.6
0.8
1

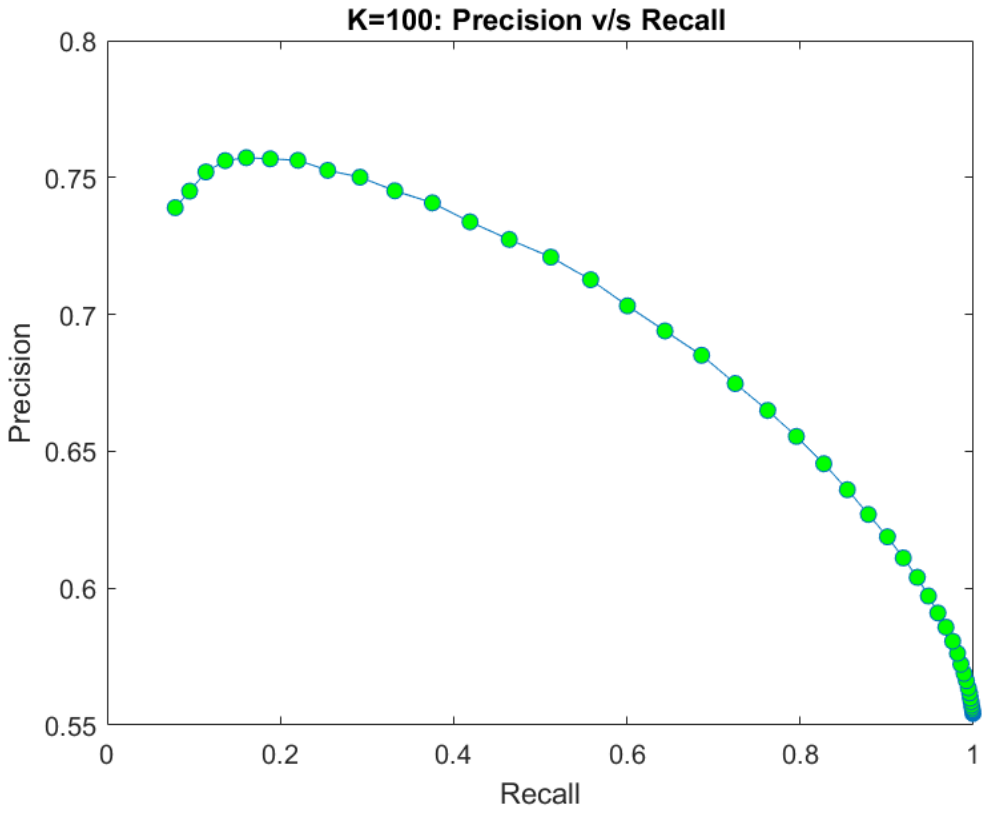

K=10: Precision v/s Threshold
Precision
Threshold
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0
1
2
3
4
5

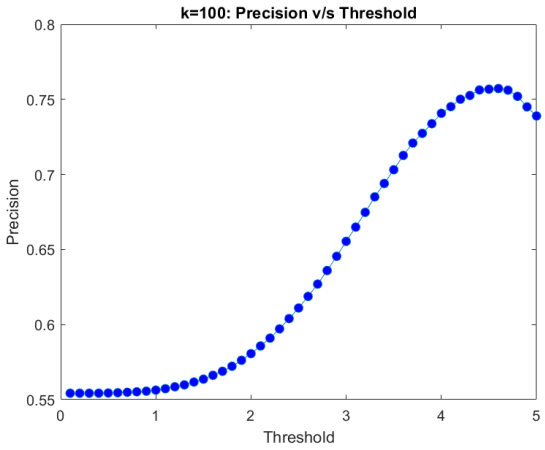

K=10: Recall v/s Threshold
Recall
Threshold
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5

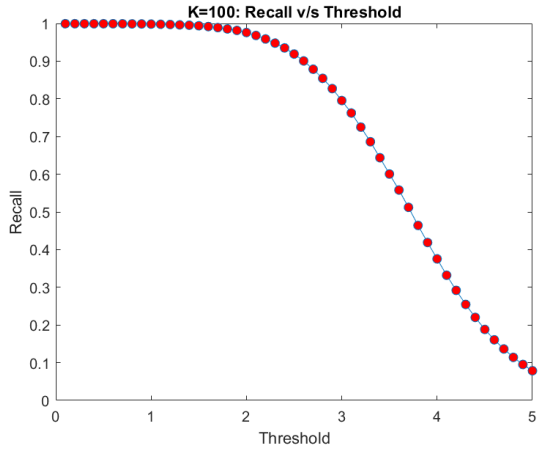

K=50: Precision v/s Recall
Precision
Recall
0.8
0.75
0.7
0.65
0.6
0.55
0
0.2
0.4
0.6
0.8
1


K=50: Precision v/s Threshold
Precision
Threshold
0.8
0.75
0.7
0.65
0.6
0.55
0
1
2
3
4
5


K=50: Recall v/s Threshold
Recall
Threshold
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5







Observations –

As the threshold value increases, there is an increase in the precision too. It can also be seen that with an increase in threshold the recall decreases. Most importantly, with an increase in recall, the precision decreases. This is true for all values of K.

All the graphs pose a similarity that as the value of K increases, the curve becomes steeper.

PART 4

The weight matrix W which was created in first part is changed in this part to assign the actual ratings to the matrix. The matrix R is modified with 1 referring to the valid entries and all entries with NaN are changed to 0. The least squared error is calculated again for the new R and W and it is observed that the value is less than the one found in Part 1.

The cost function is modified by adding a regularization term. This term forces all the entries to shrink. This is accompanied by extracting the predicted value of entries that have an actual rating. This term becomes more significant for the entries which have a higher rating. The entries that are more favorable will correspond to higher values in the precited matrix. The equation is as follows–

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

The above equation is solved by alternating least squared function where the matrixes U and V are iteratively updated to minimize the residual error. To do this, *reg_wnmfrule* function is used. The general formula for calculating factorized matrix is –

$$A = A.*(((W.*X)*Y')./((W.*(A*Y))*Y'))$$

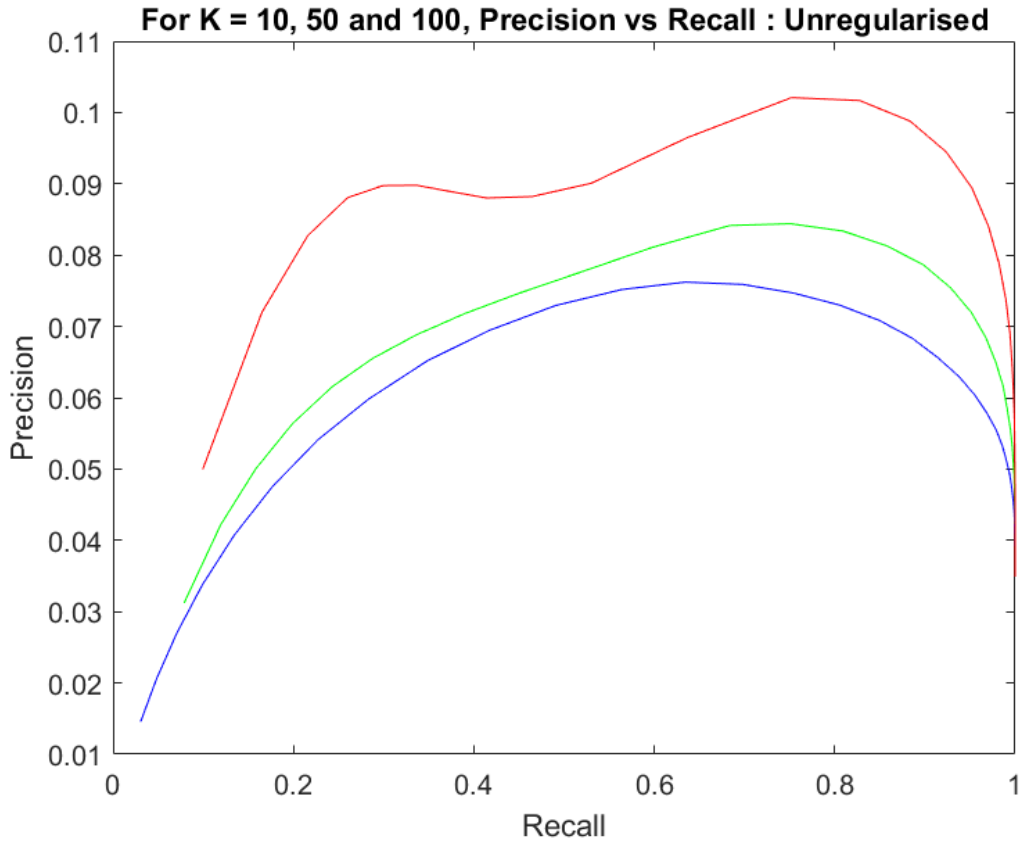
To get the values of U and V, the W and R matrix are passed along with the value of K. As the output, we get the factorized matrix U and V and he residual error.

For 150 iterations, the following output was recorded –

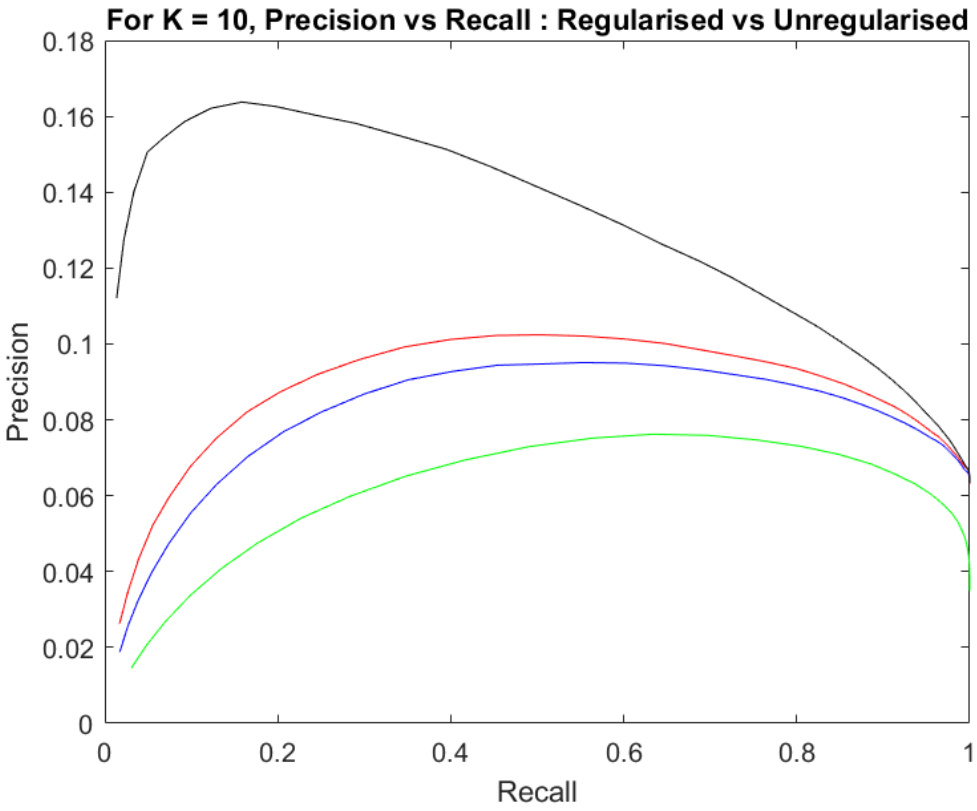
Normal R and W	
K	**Least Squared Error**
10	5.846268e+04
50	2.683503e+04
100	1.362224e+04
Swapped R and W	
K	**Squared Residual Error**
10	3.112043e+01
50	1.040943e+02
100	9.420349e+01

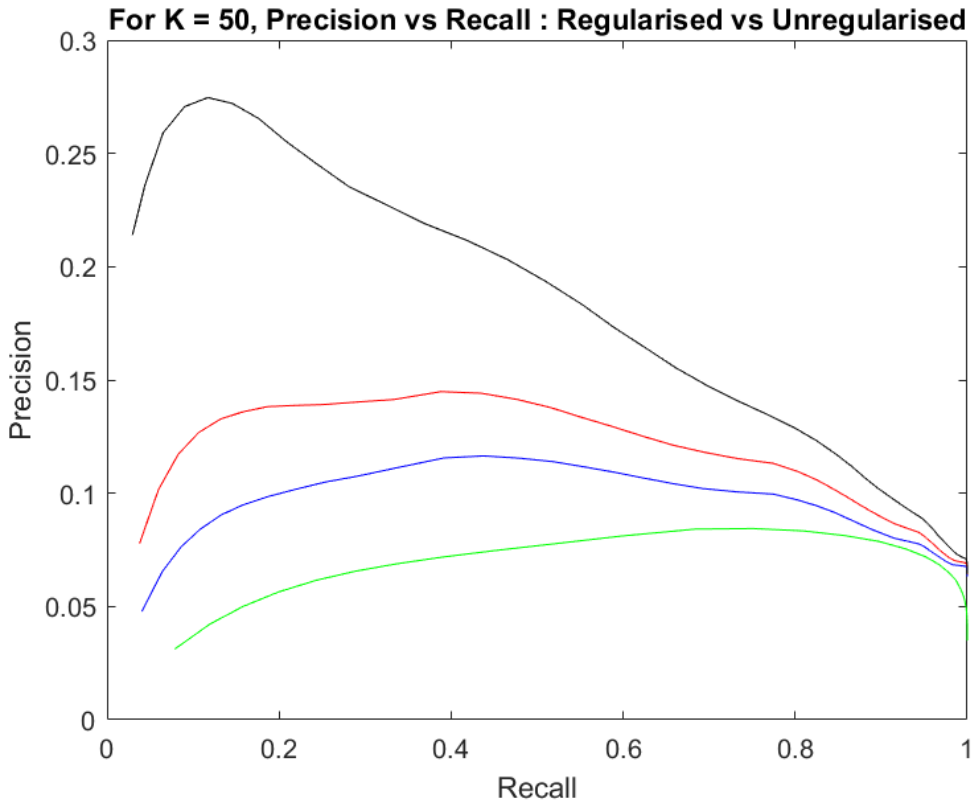
Regularized R and W		
Lambda	**K**	**Squared Residual Error (1.0e+04)**
0.01	10	5.8750
	50	2.6930
	100	1.3871
0.1	10	5.8778
	50	2.6949
	100	1.3885
1	10	5.9149
	50	2.8664
	100	1.5512

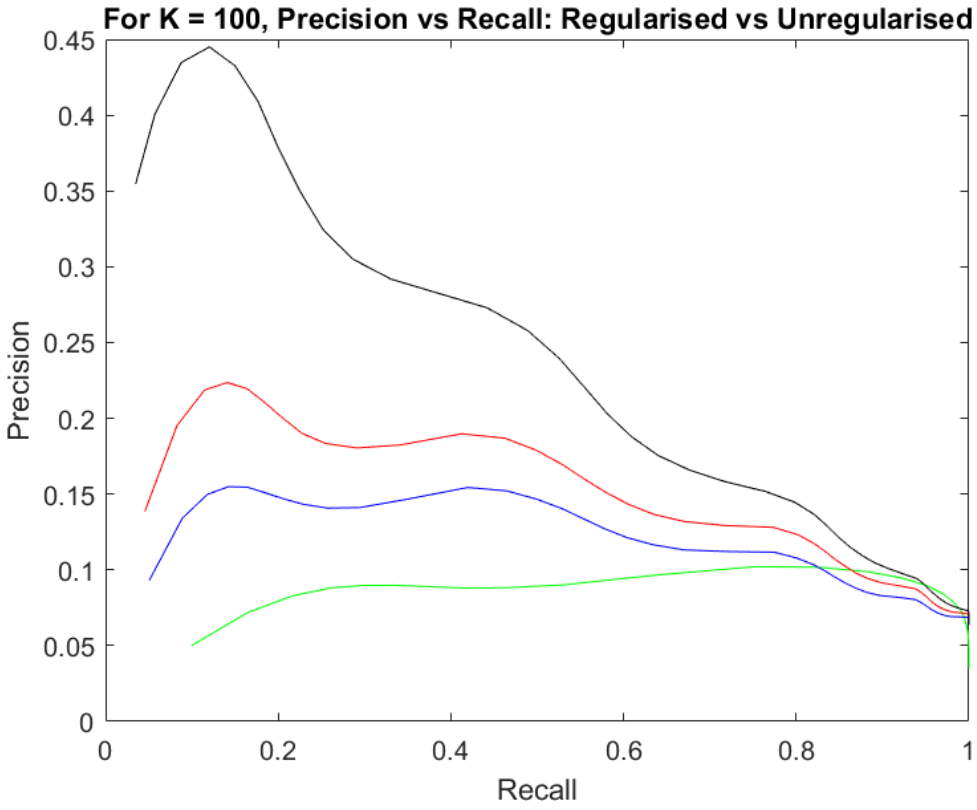
The following graphs depict the performance for unregularized R and W. The base models of normal and swapped R and W act as the benchmark for regularization. The following graph plots Precision v/s Recall for values of K varying from 10 (blue), 50 (green) and 100 (red).



The graphs for regularized R and W v/s unregularized with K varying as 10, 50 and 100 are as follows –


For K = 10, Precision vs Recall : Regularised vs Unregularised
Precision
Recall
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
0.2
0.4
0.6
0.8
1


For K = 50, Precision vs Recall : Regularised vs Unregularised
Precision
Recall
0
0.05
0.1
0.15
0.2
0.25
0.3
0
0.2
0.4
0.6
0.8
1


For K = 100, Precision vs Recall: Regularised vs Unregularised
Precision
Recall
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0
0.2
0.4
0.6
0.8
1

The green refers to the graph of unregularized R and W.
The black, red and blue refers to the unregularized curves for lambda – 0.01, 0.1, 1

Observation -

Regularized curves are more smoother than curves for unregularized R and W.

PART 5

The matrix R is again modified. This time for every valid entry a 1 was recorded and otherwise a NaN is stored. The weight matrix W consists of the actual ratings that the users had given to each movie. A 10-fold cross validation was performed and the new predicted matrix was formed consisting of predicted ratings for known data values. These predicted ratings for each user were sorted in a decreasing order and the corresponding movies were stored separately. The top L = 25 movies were found by increasing the value of L from 1 upto 25. For each L, a new matrix was created that stored the top L movies for each user.

The algorithm's hit rate and false alarm rate was calculated according to the formulas –

$$Hit\ Rate = \frac{True\ Postitve}{True\ Positive + False\ Negative}$$

$$False\ Alarm\ Rate = \frac{False\ Postitve}{False\ Positive + True\ Negative}$$

In this, true positive refers to the number of times that the actual and precited ratings were above the threshold. True negative refers to both these ratings being below the threshold. False positive refers to the number of times that the predicted rating is below the threshold and actual rating is above. False negative refers to the number of times the actual rating is below the threshold value when the precited rating is not.

All the actual ratings were compared against the ground threshold of 3. This means that any rating less than 3 refers to the movie being disliked. For any rating greater than this threshold refers to the movie being liked by that particular user. Based on the previous part, the threshold of 0.4 was taken as found by the experimental results.

For L = 5, the average precision of the algorithm was calculated according to the formula –

$$Precision = \frac{True\ Postitve}{True\ Positive + False\ Positive}$$

The results for K = 10, 50 and 100 are –

K	Average Precision
10	5.191941e-01
50	5.272534e-01
100	5.563097e-01

Observation -

The average precision keeps on increasing as the value of K increases. The highest precision is obtained at K=100 as 5.563e-01.

The mean hit rate and the mean false-alarm rate for different values of K are -

K = 10

L	Mean Hit Rate	Mean False-alarm Rate
1	4.962884e-01	5.037116e-01
5	9.172853e-01	9.066808e-01
10	9.840933e-01	9.745493e-01
15	9.957582e-01	9.851538e-01
20	9.384942e-01	9.330955e-01
25	8.239661e-01	8.239661e-01

K = 50

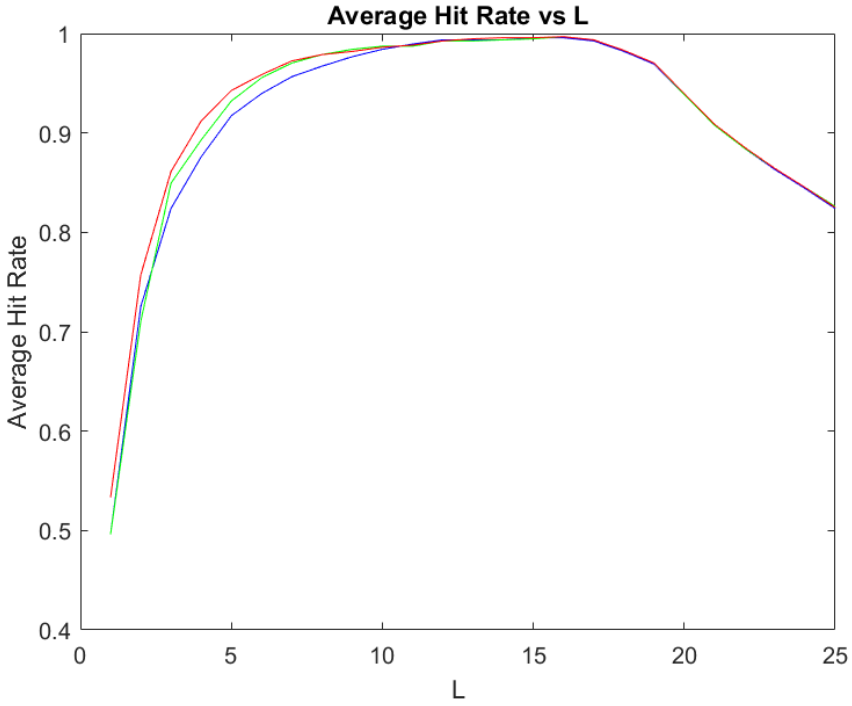
L	Mean Hit Rate	Mean False-alarm Rate
1	4.962884e-01	5.037116e-01
5	9.172853e-01	9.066808e-01
10	9.840933e-01	9.745493e-01
15	9.957582e-01	9.851538e-01
20	9.384942e-01	9.330955e-01
25	8.239661e-01	8.239661e-01

K = 100

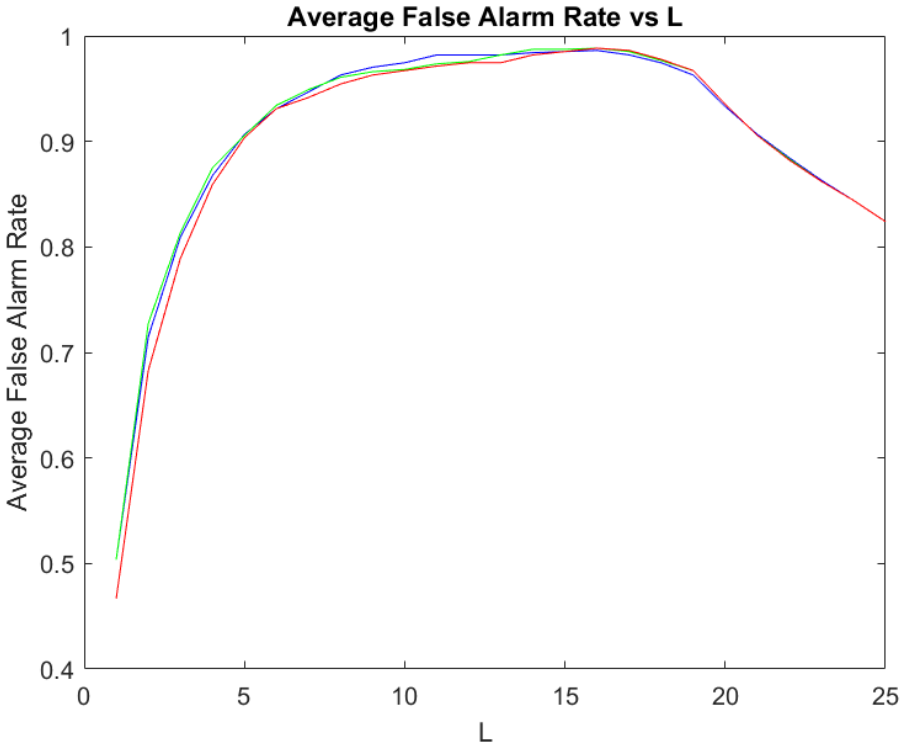
L	Mean Hit Rate	Mean False-alarm Rate
1	4.962884e-01	5.037116e-01
5	9.172853e-01	9.066808e-01
10	9.840933e-01	9.745493e-01

15	9.957582e-01	9.851538e-01
20	9.384942e-01	9.330955e-01
25	8.239661e-01	8.239661e-01

We plot the graph for Average hit rate v/s L for values of K = 10 (blue), 50 (green), 100 (red). As the value of L increases, the hit rate start converging to 1 which is the ideal solutions. As L increases, the scope of the movies which can be suggested to the user increases. The hit rate of 1 comes up if the number of movies suggested to the user surpasses the number of movies actually liked by the user.



Plotting the graph for Average false alarm rate v/s L for values of K = 10 (blue), 50 (green), 100 (red) reveals that the case is very similar to the previous graph. As L increases, there is a steady increase in the False alarm rate reaching 1. Again, the hit rate of 1 comes up if the number of movies suggested to the user surpasses the number of movies actually liked by the user.



Both of the above graphs, as the value of L increases the curves reach 1. Hence, the Hit rate vs False alarm rate curve also increases in a manner that it approaches 1.