Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Course No. : CSE 4238
Course Name : Soft Computing Lab

Assignment No. : 01

Submitted By:

Name : Sadia Tasnim
ID No. : 17 01 04 037
Session : Fall - 2020
Section : A (A2)

Contents

Generation of Loss curves for the hyper parameters

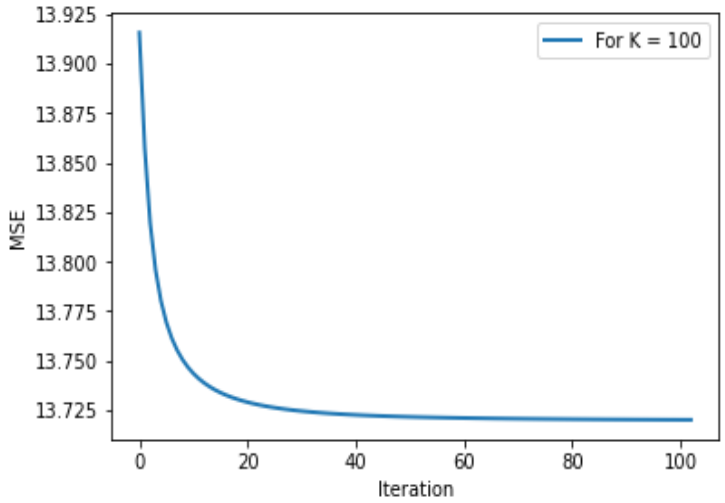
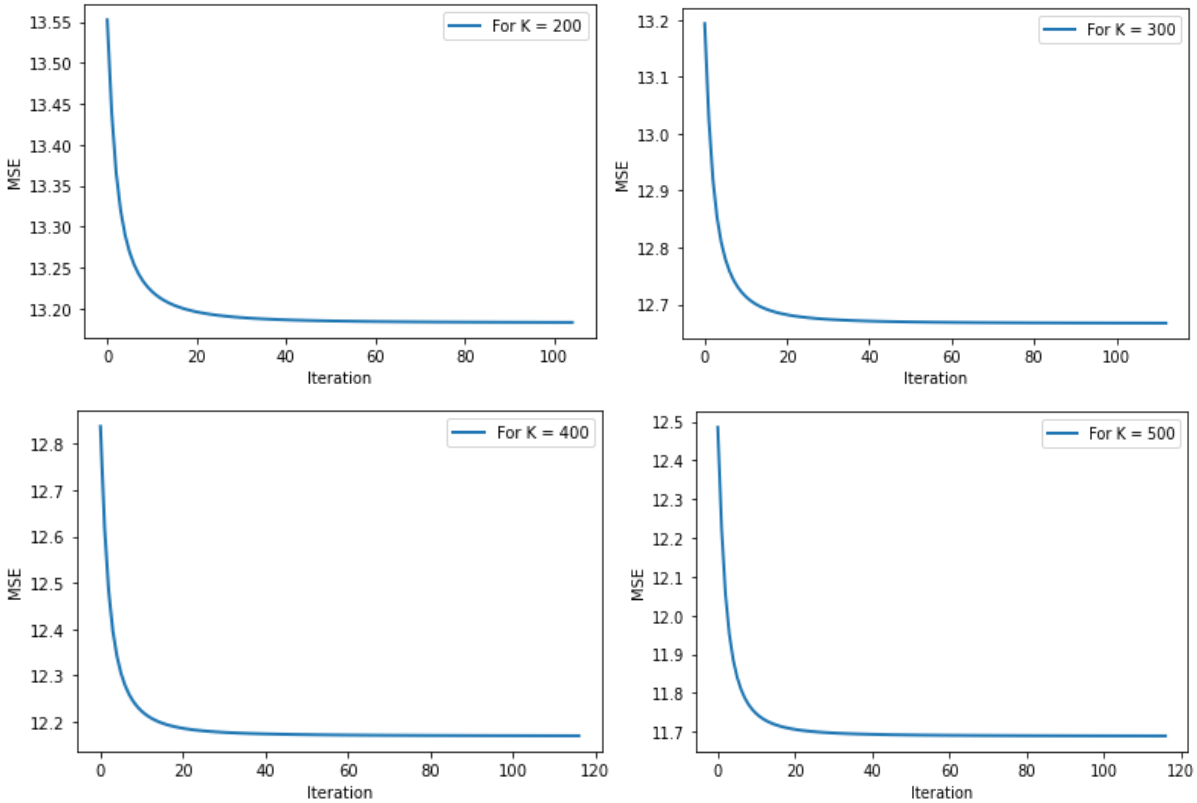
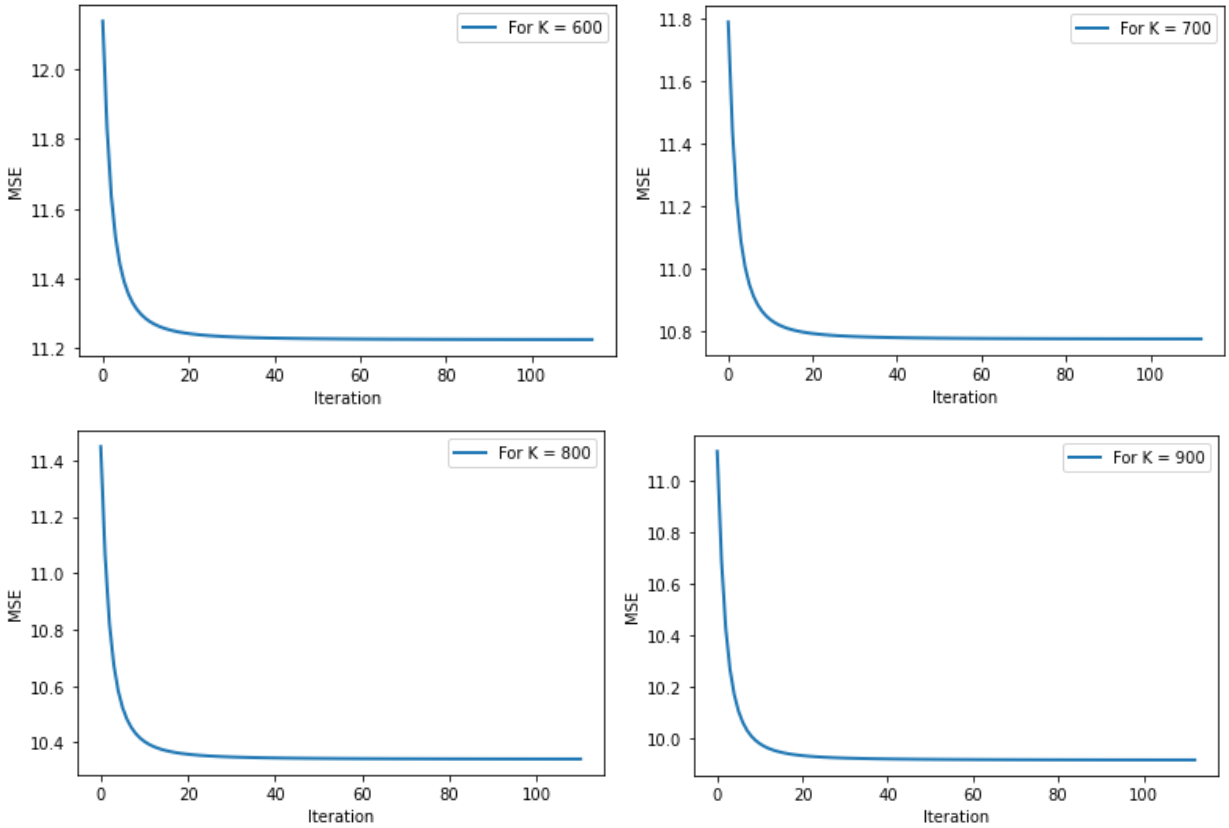


Figure 1 : Loss curve for K = 100 and iteration = 100

Here for K = 100 calculated mean squared error was 13.725. At the beginning, error was 13.925 but with the improved U and V loss reduced and after 100 iterations convergence was too slow .After comparing loss L, between consecutive iterations convergence was less than 0.0001. So K was updated into 200.





The value for K was updated 10 times (100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000)

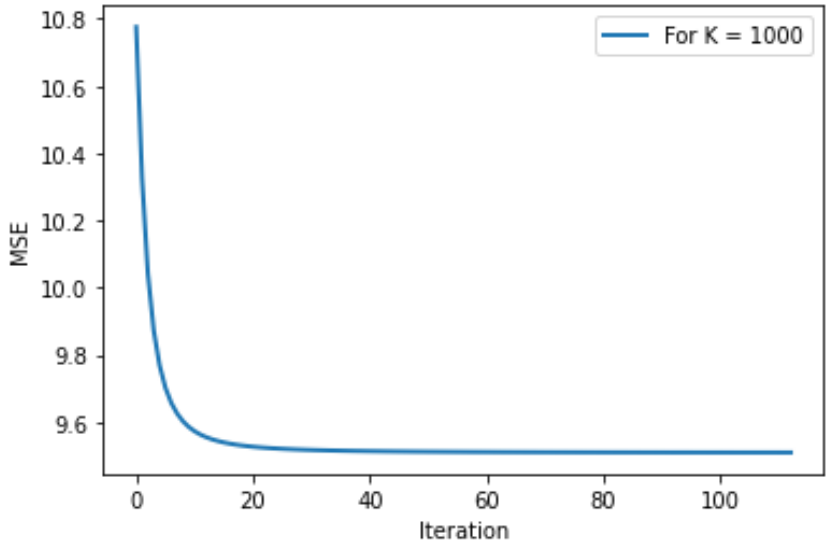


Figure 2 : Loss curve for K = 1000 and iteration = 110

For K = 1000 mean squared error reduced into 9.6. After 110 iterations convergence was less than 0.0001 and was not notable.

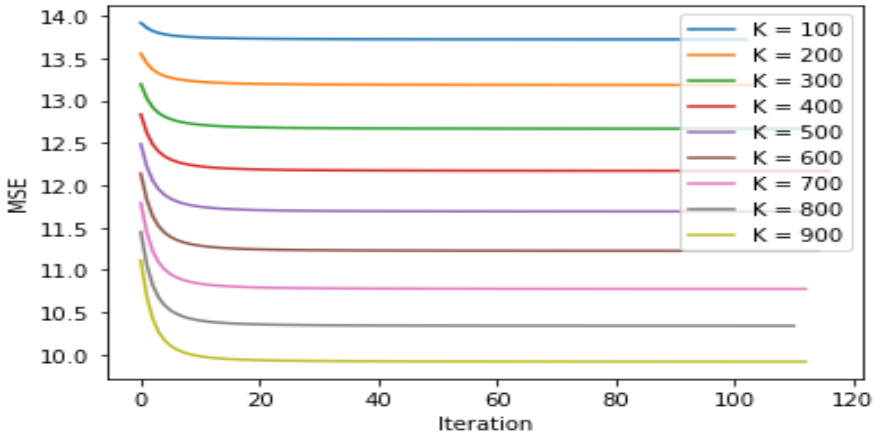


Figure 3 : Comparing loss curve for K =100 to 1000

As K is increasing, value of L is decreasing. So it can be said that for a large value of K loss can be reduced.

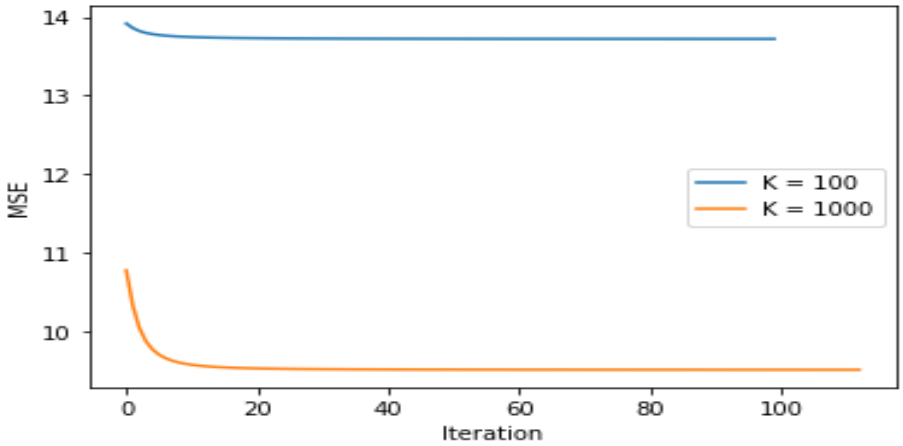


Figure 4 : Comparing loss curve for K = 100 and K = 1000

By comparing two loss curve, for K = 100, Loss was 13.725 but for K = 1000, Loss was 9.6. For K = 100, After 100 iterations loss convergence was less than 0.00001. But as the value of K increased number of iteration also increased .For K = 1000, iteration continued till 110 for reaching convergence 0.00001.

The optimal hyper parameters with proper explanation

As said before with a large value of K loss can be reduced.

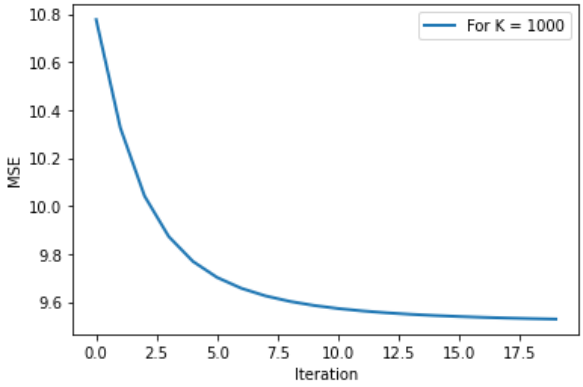


Figure 5: Loss curve for K=1000, Iteration = 20

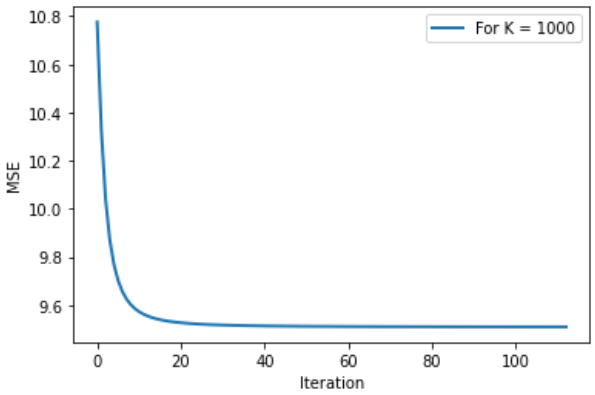


Figure 6: Loss curve for K = 1000, Iteration = 110

K = 1000 and iteration number=110 cannot be considered as an optimal hyper parameter. Loss is 9.6 for this value. It can reduce more with larger value of K. But as K's value is increasing, computational cost was also increasing. K = 100 took around 18 minutes to compute but K = 1000 took 2 hours 30 minutes.

But by reducing iteration number computation time can be reduced. K = 1000 with 20 iterations took only 20 minutes to compute and the loss was 9.6. After 20 iteration convergence was very low.

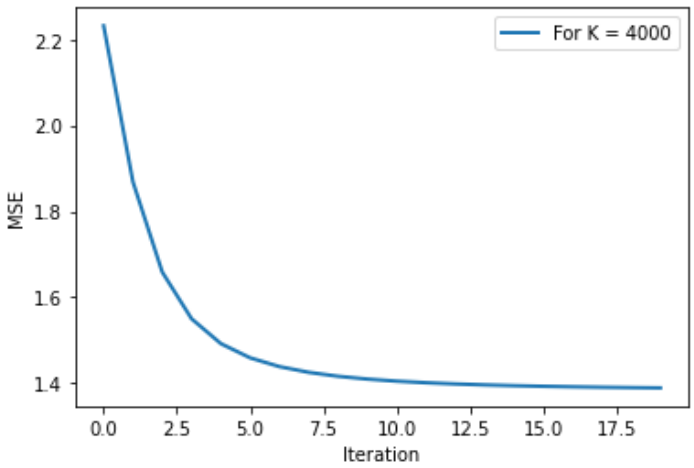


Figure 7: Loss curve for K = 4000, Iteration = 20

K = 4000 with iteration number = 20 can be considered as an optimal hyper parameter. If we decrease the number of iteration into 20, change in loss was not notable. As from the loss curve we can see that the line was flattened after 20 iterations. Here the loss was 1.4 and computation time was 1 hour 20 minutes. Larger value for K with less iterations can be considered as an optimal hyper parameter value as it took less time to compute and also gives a minimum loss.

Cosine similarity between all pairs of users

Cosine similarity is measured to know the similarity between two points by complementing to cosine distance. It shows the similarity between two points. As the distance increases similarity decreases and vice versa. Matrix factorization is used in recommendation system. For movie suggestion users or movies can be clustered into groups according to the similarities.

	pair	similarity
0	([0.50852215, -8.244865, -2.1389105, -12.59222...	-0.036034
1	([0.50852215, -8.244865, -2.1389105, -12.59222...	-0.024205
2	([0.50852215, -8.244865, -2.1389105, -12.59222...	0.036196
3	([0.50852215, -8.244865, -2.1389105, -12.59222...	0.035086
4	([0.50852215, -8.244865, -2.1389105, -12.59222...	0.007697

	pair	similarity
10122745	([1.9837978, 8.961695, -0.15829039, -3.4256666...	-0.015252
10122746	([1.9837978, 8.961695, -0.15829039, -3.4256666...	-0.039389
10122747	([4.7360296, -8.561624, 0.9574435, -2.6274548,...	-0.001276
10122748	([4.7360296, -8.561624, 0.9574435, -2.6274548,...	0.031909
10122749	([-3.7575936, -6.393804, 2.8837128, -1.4495814...	-0.014330

Figure 8 : Cosine similarities for users (U)

At first for N users, total N*(N-1) pairs were made to compute cosine similarity. Then by using ‘spatial.distance’ cosine distance was measured and after that cosine similarity was calculated. Larger value of similarity indicates more similarity between two pairs.
For computation cost issue similarity between 4500 users were calculated.

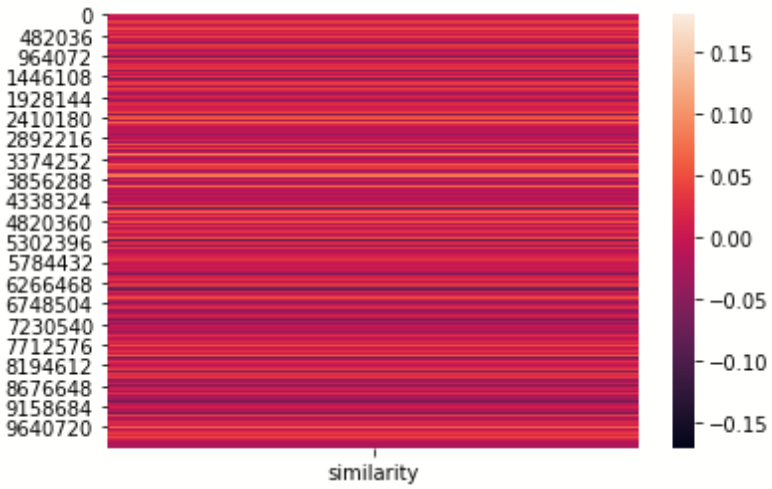


Figure 9 : Heat map for U

Darker color in heat map means more similarity for a pair and lighter color means less similarity.

Cosine similarity between all pairs of movies

All the movies can be clustered into different groups according to the cosine similarity. For example, if a move is both comedy and action then it has some similarities with other action movies and can be clustered into one group.

	pair	similarity
0	([0.045074146, 0.039826065, 0.004812716, 0.026...	0.719928
1	([0.045074146, 0.039826065, 0.004812716, 0.026...	0.704116
2	([0.045074146, 0.039826065, 0.004812716, 0.026...	0.706271
3	([0.045074146, 0.039826065, 0.004812716, 0.026...	0.728366
4	([0.045074146, 0.039826065, 0.004812716, 0.026...	0.708168

	pair	similarity
10122745	([-0.0034558326, 0.021391653, 0.0267021, 0.011...	0.723414
10122746	([-0.0034558326, 0.021391653, 0.0267021, 0.011...	0.700795
10122747	([0.012811938, 0.027002161, 0.03344594, 0.0552...	0.727188
10122748	([0.012811938, 0.027002161, 0.03344594, 0.0552...	0.714991
10122749	([0.07942145, 0.0052551418, 0.03735216, 0.0105...	0.706201

Figure 10 : Cosine Similarity for V

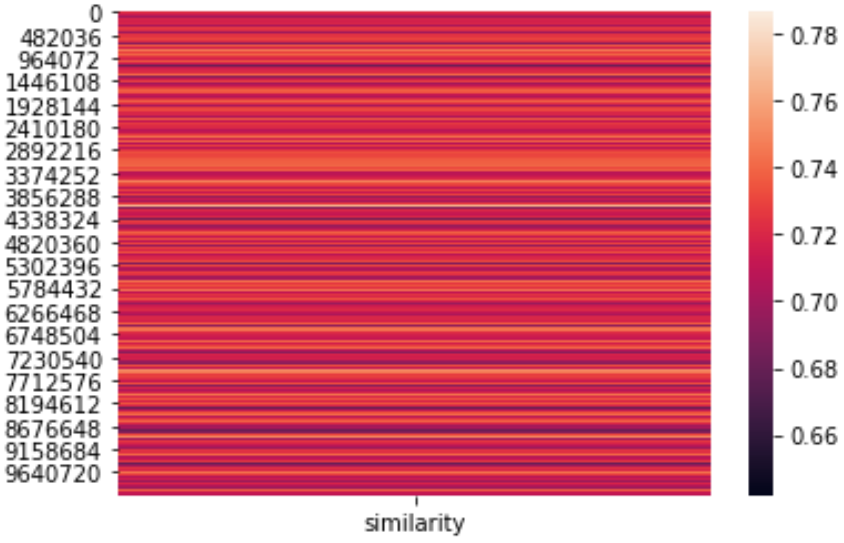
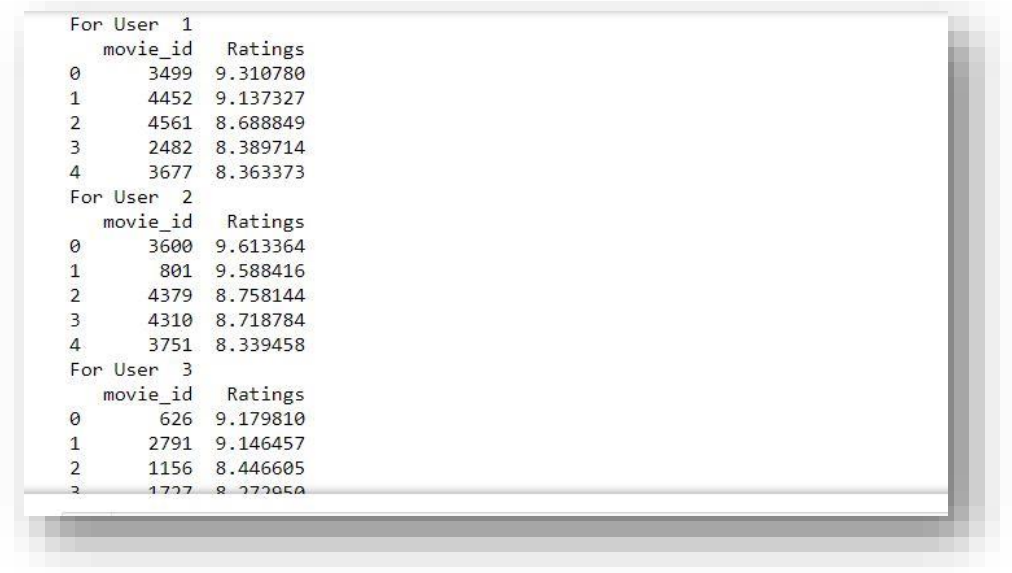


Figure 11: Heat map for V

Suggestion of five movies (V) that user (U) did not review yet

At first every ‘nan’ value of movies (V) for a particular user (U) was stored. Then from the predicted dataset, those ‘nan’ values were collected and sorted into ascending order. According to the instruction first 5 movies with their rating were shown.



```
For User  1
   movie_id   Ratings
0      3499  9.310780
1      4452  9.137327
2      4561  8.688849
3      2482  8.389714
4      3677  8.363373
For User  2
   movie_id   Ratings
0      3600  9.613364
1       801  9.588416
2      4379  8.758144
3      4310  8.718784
4      3751  8.339458
For User  3
   movie_id   Ratings
0       626  9.179810
1      2791  9.146457
2      1156  8.446605
3      1727  8.272950
```

Figure 12 : Five movie suggestion for users

Result analysis

	K = 100	K = 500	K = 1000	K=1000	K=4000
Iteration	100	100	110	20	20
Loss	13.725	11.7	9.6	9.6	1.4
Computation Time	18 minutes	1 hour	2 hour 30 minutes	20 minutes	1 hour 20 minutes

Table 1: Result analysis

Here we can see for K =1000 with 110 iterations Loss was same 9.6 but as we decreased the number of iteration, computation time reduced a lot. For K = 4000 and no. of iteration = 20, loss was reduced into 1.4

Figure 13 : Result analysis

Applications

Generally Matrix Factorization is used for recommendation systems. Some applications of these is given below

- E-commerce sites can use it for recommending items to the users
- Social Media like Facebook, Instagram can recommend various types of content to the user
- Netflix, Amazon etc. can use it for recommending. Though Amazon have their own algorithm for it.
- With the help of cosine similarity items and users can be clustered into different groups.