COMS 6998

Hw1

xw2501

Xiaoning Wan

Description of Functions

As there are not too many functions, I will not specially create a readme file, descriptions and notes about functions will all be inclued here.

- 'preprocess'
  Load 'ratings.csv' skipping the head line, and remove the fourth column, which contains timestep information that is not used in the algorithm. Then save the processed data, other functions will only load the processed data.
- 'matrix_fac'
  Main function. Define paramters, load data, execute the algotithm and save the data. In order to test the affect of both rank and λ, I used a double loop to go through all possible combination in a pre-defined array of value.
  - 'computeVW'
    This is the core part of matrix_fac. This function implements the gradient descending algorithm.
    $$\min_{V,W} \sum_{(i,j)\epsilon\Omega} \left(R_{i,j} - (VW^T)_{i,j}\right)^2 + \lambda\,(\|V\|_F^2 + \|W\|_F^2)$$
    Updating rule
    $$V \leftarrow V + \alpha[(R - VW^T)W - \lambda V]$$
    $$W \leftarrow W + \alpha[(R - VW^T)^T V - \lambda W]$$
    Yet, as the size of R and VW^T matrix are too large to be created, the updating is actually performed in an element wise way. As the matrix multiplication is a linear operation, updating the matrix element by element instead of updating it as a whole does not affect the result.
    For a comment $[userId, movieId, rating]$, the relevant update made is
    $$V(userId) \leftarrow V(userId) + \alpha[(R - V(userId)W(movieId)^T)W(movieId) - \lambda V(userId)]$$
    $$W(movieId) \leftarrow W(movieId) + \alpha[(R - V(userId)W(movieId)^T)^T V(userId) - \lambda W(movieId)]$$
    Keep updating until any one of the following conditions mathces:
    i. Exceeded max loop times
    ii. Updating made is smaller than threshold
    iii. Average rating error starts to increase
  - 'computeRMSE'
    As V and W are already computed, this function computes RMSE of test set based on obtained V and W.
  - 'computeMRR'
    Computes the MRR by sorting the prediction ratings in test set. Then find rank of movies with actual rating higher or equal to 3. And thus the MRR can be easily computed.
- 'visualization'
  Load the test results, and then visualize the results.

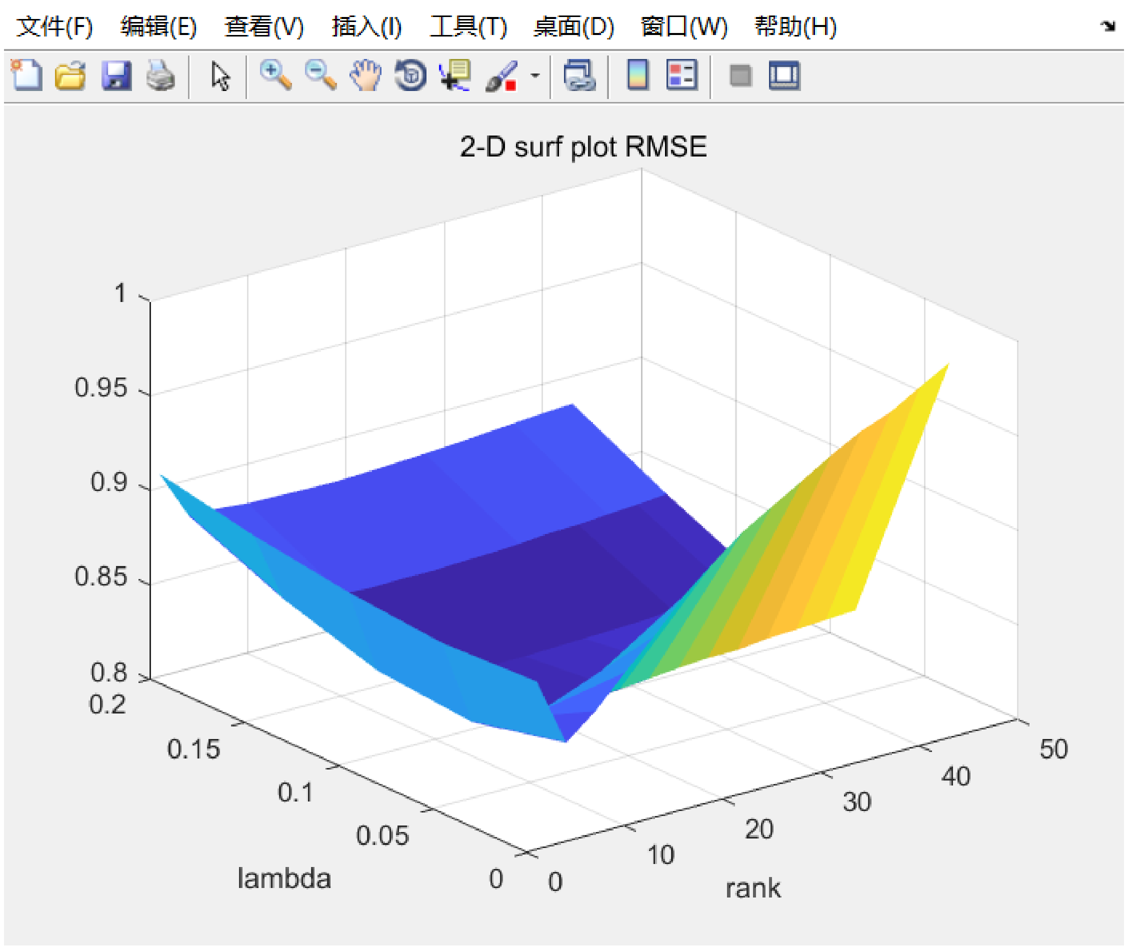
Notes,

Before the update of homework, I used the rank in all movies to compute MRR, this part of codes is kept in the 'computeMRR' and is commented. Some test results made before the updating are all computed in this way and consequently have a very small MRR.

Results and Figures

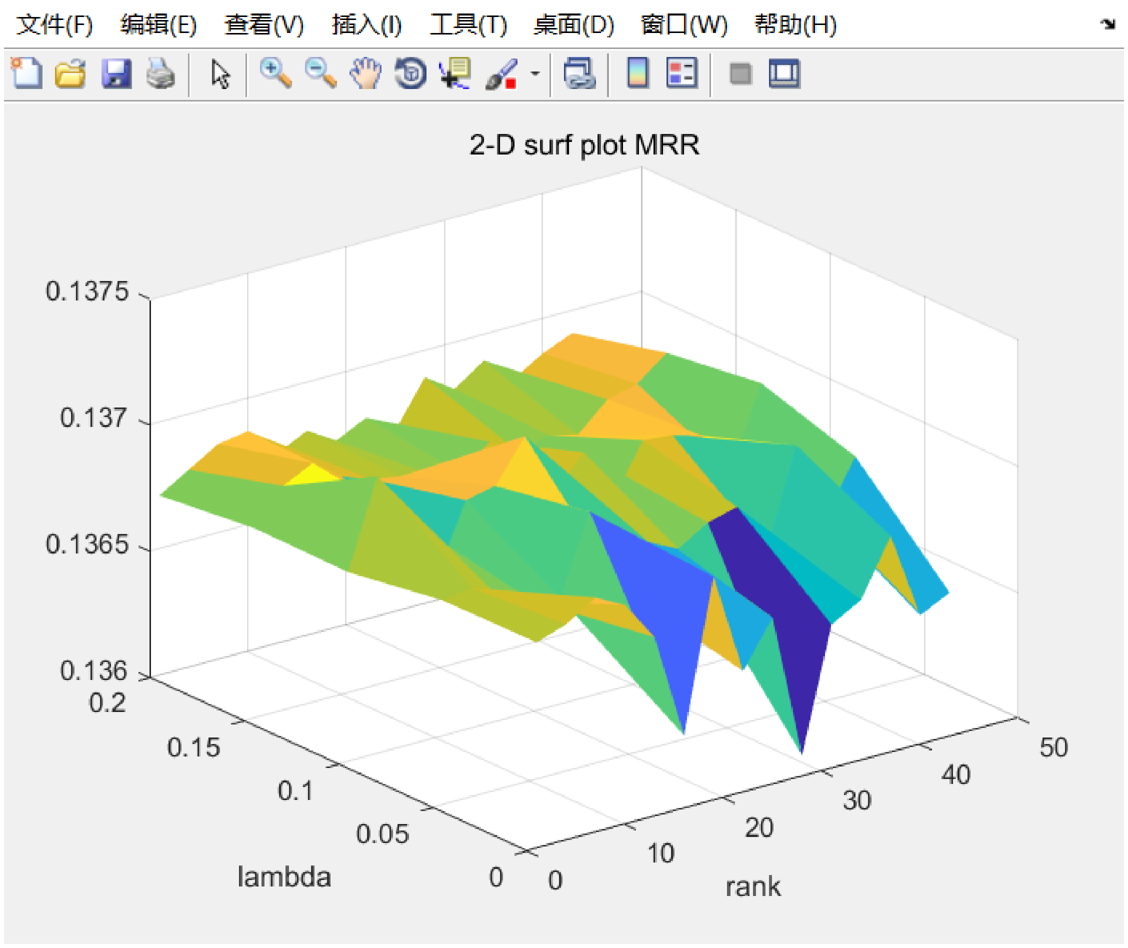
I wrote a function named 'visualization' to create the figures, you can run it to reproduce them, or you can just open a folder named results where I saved all the graphes.

A surface map over rank and λ.



In this 2d surface plot of RMSE over a choice of rank and λ, I found that to get optimal performance of RMSE, λ should be set to 0.1.

As for the rank, with rank increasing, RMSE descends first, and then increases.



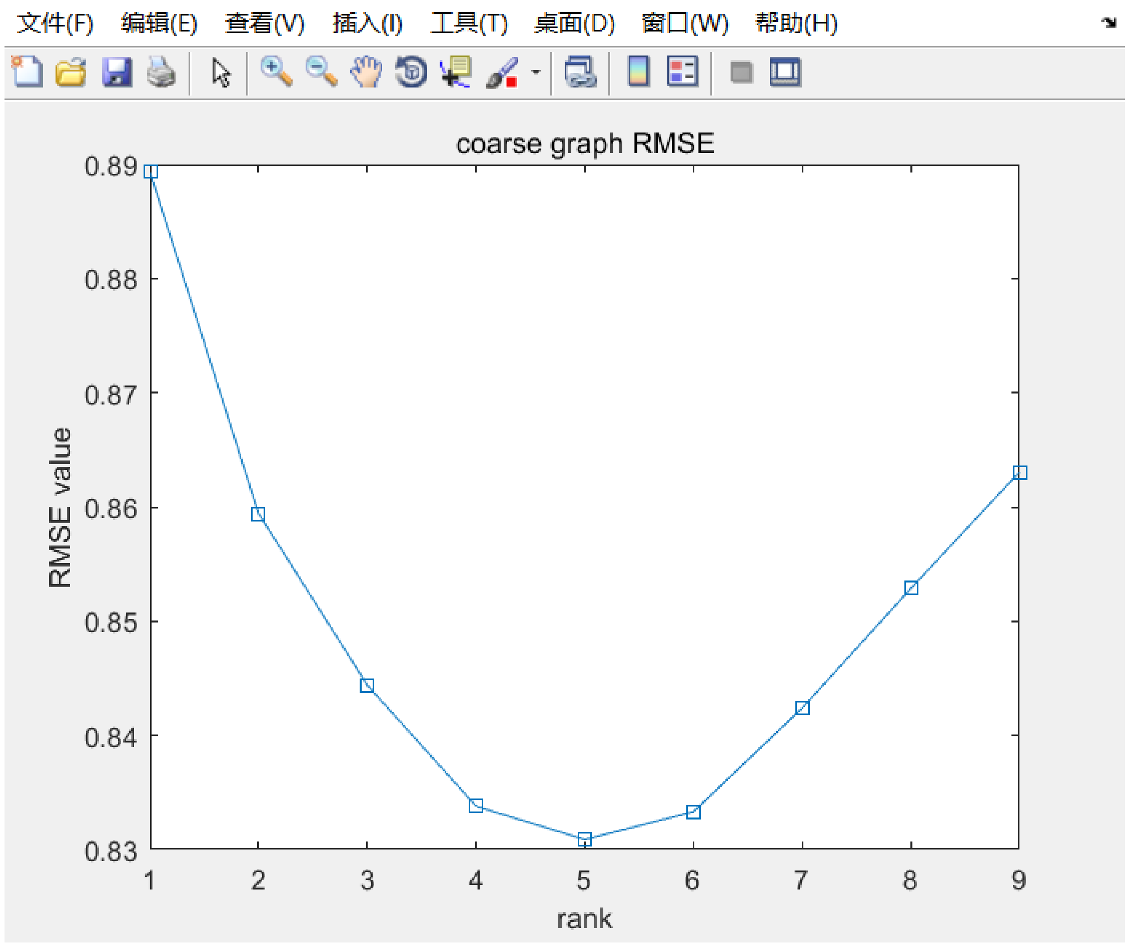
In this 2d surface plot of MRR over a choice of rank and λ, I found that MRR does not vary much as λ changes. Except for $\lambda = 0$, other choice of λ does not seem to affect the MRR very much.

All above, in following graph, λ will be set to 0.1, and I will mainly focus on testing for the optimal rank.

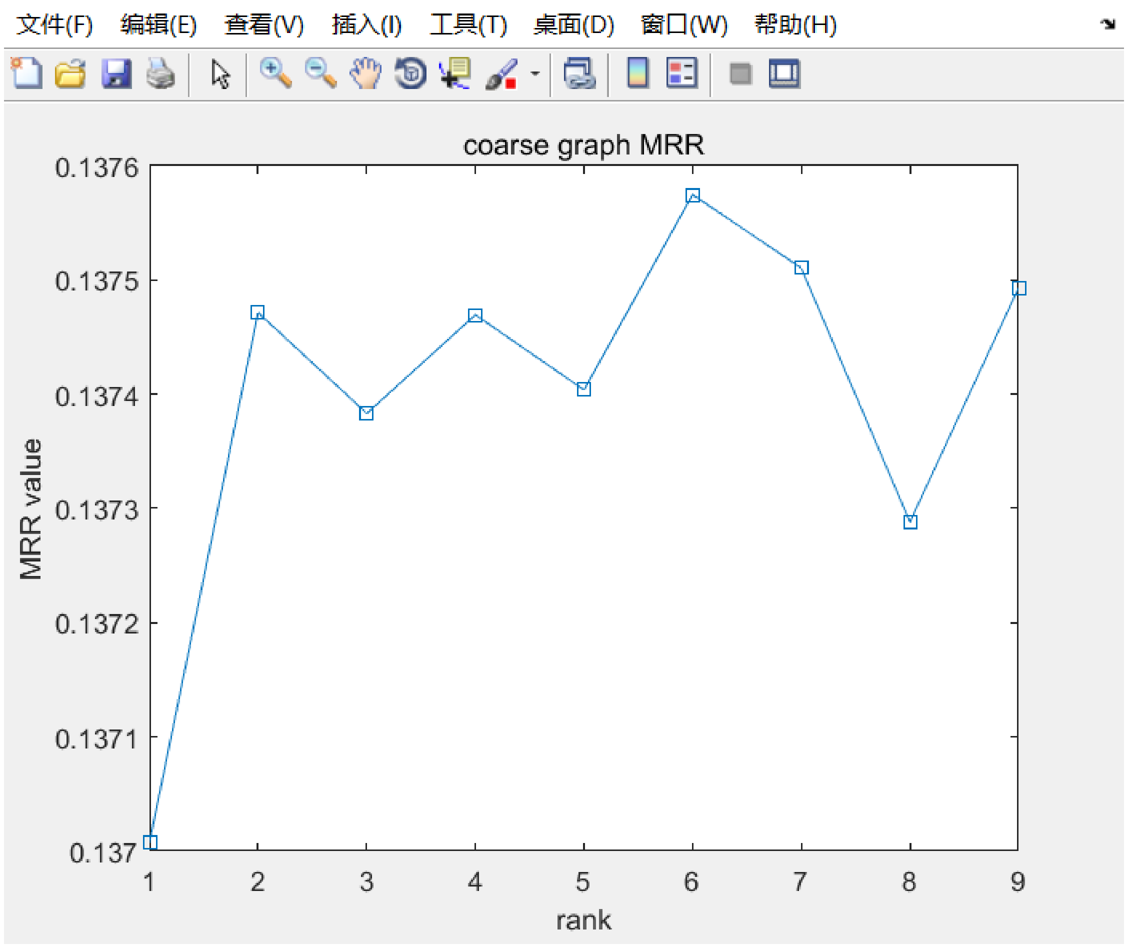
Coarse graph over ranks

To find the optimal rank for performance of RMSE and MRR, I first chose a set of rank with exponential growth to roughly find the optimal rank, or to say, find a range of ranks that the optimal is most likely to occur. The ranks used for test are 1,2,4,8,16,32,64,128,256.

Note that here the label on x-axis does not indicates its real rank, the real rank should be $r = 2^{i-1}$, where i is the label.



As rank grows, RMSE decreases first and then increases. And we can see that the minimum RMSE occurs at rank ranging from 8 to 32.

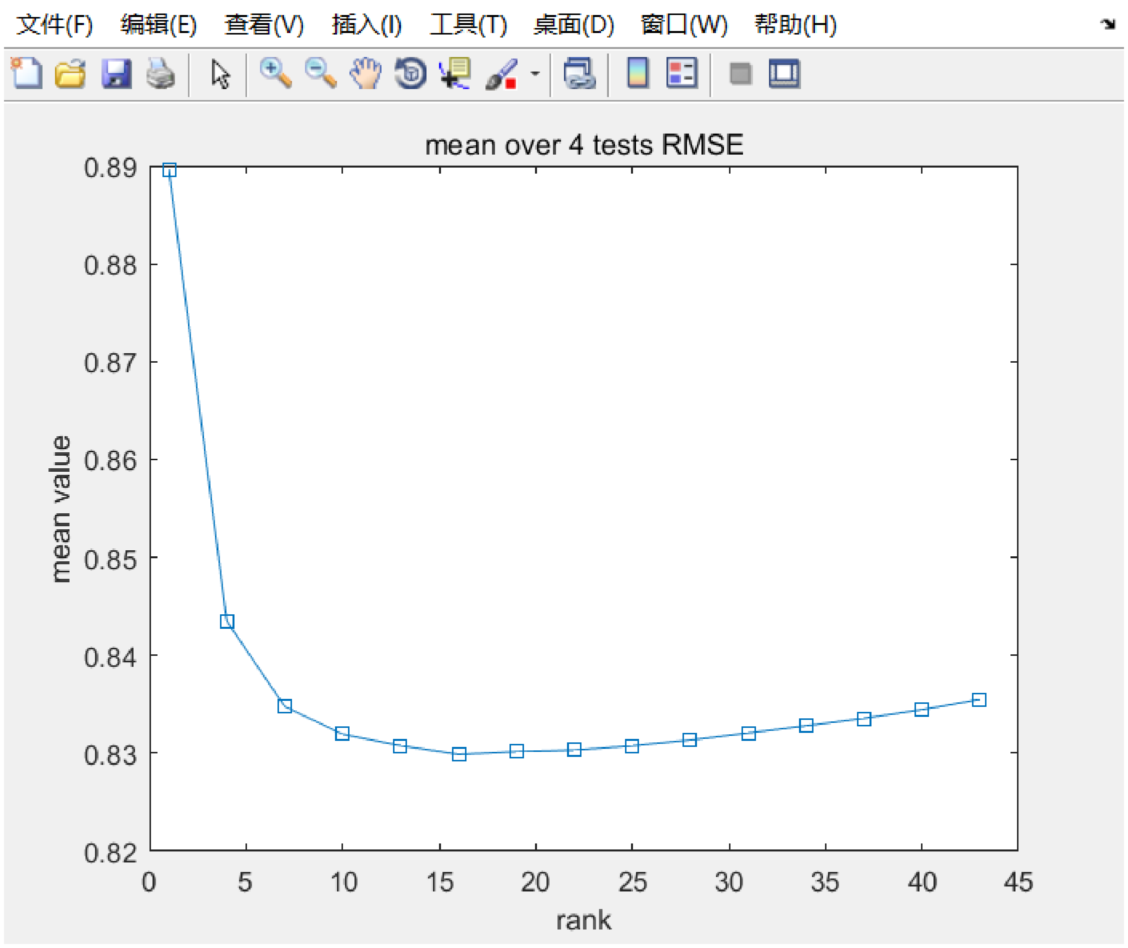


Though the curve of MRR is not as smooth as RMSE, still we can find that MRR increases first, and then becomes steady after rank equals to 2, and then it fluctuates within a range.

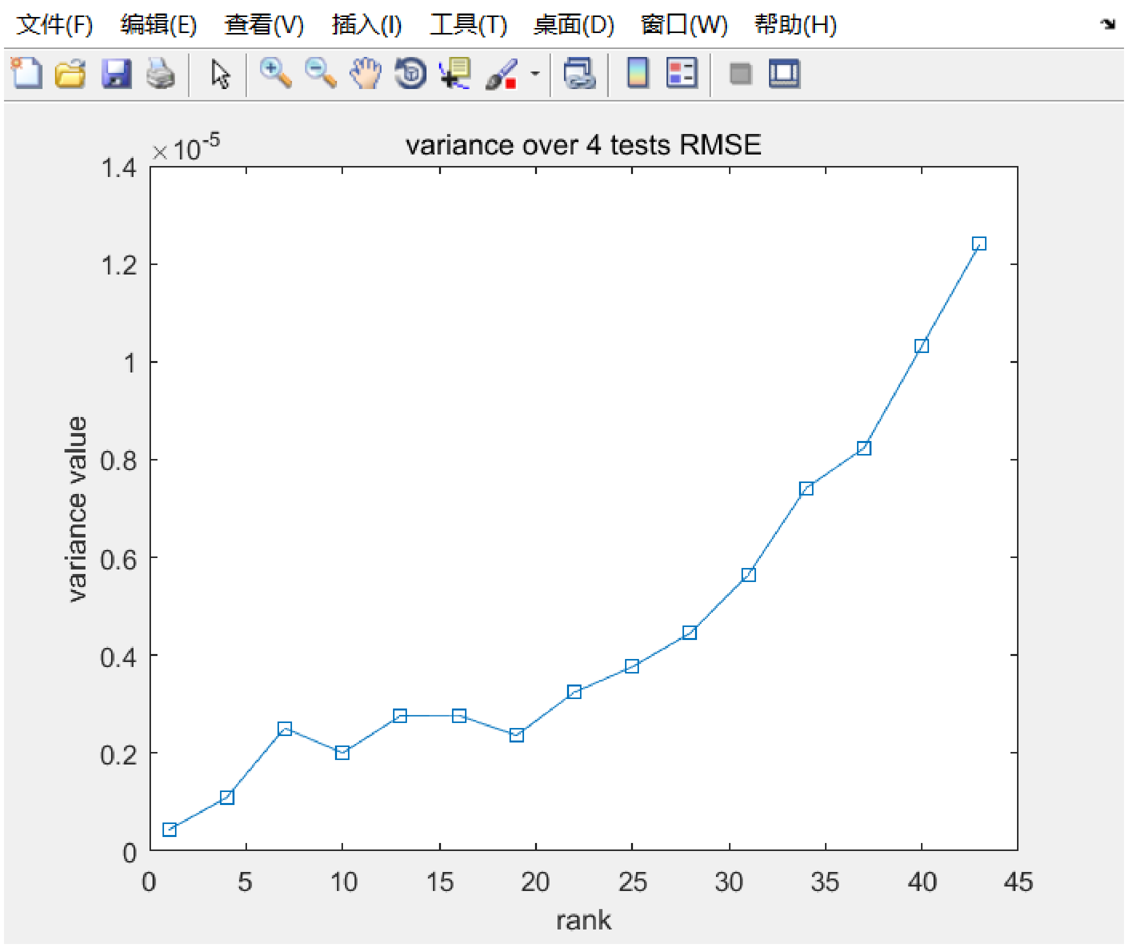
Based on these two graphes, next step is to find the rank which gives optimal (minimal) RMSE.

Fine graph over ranks

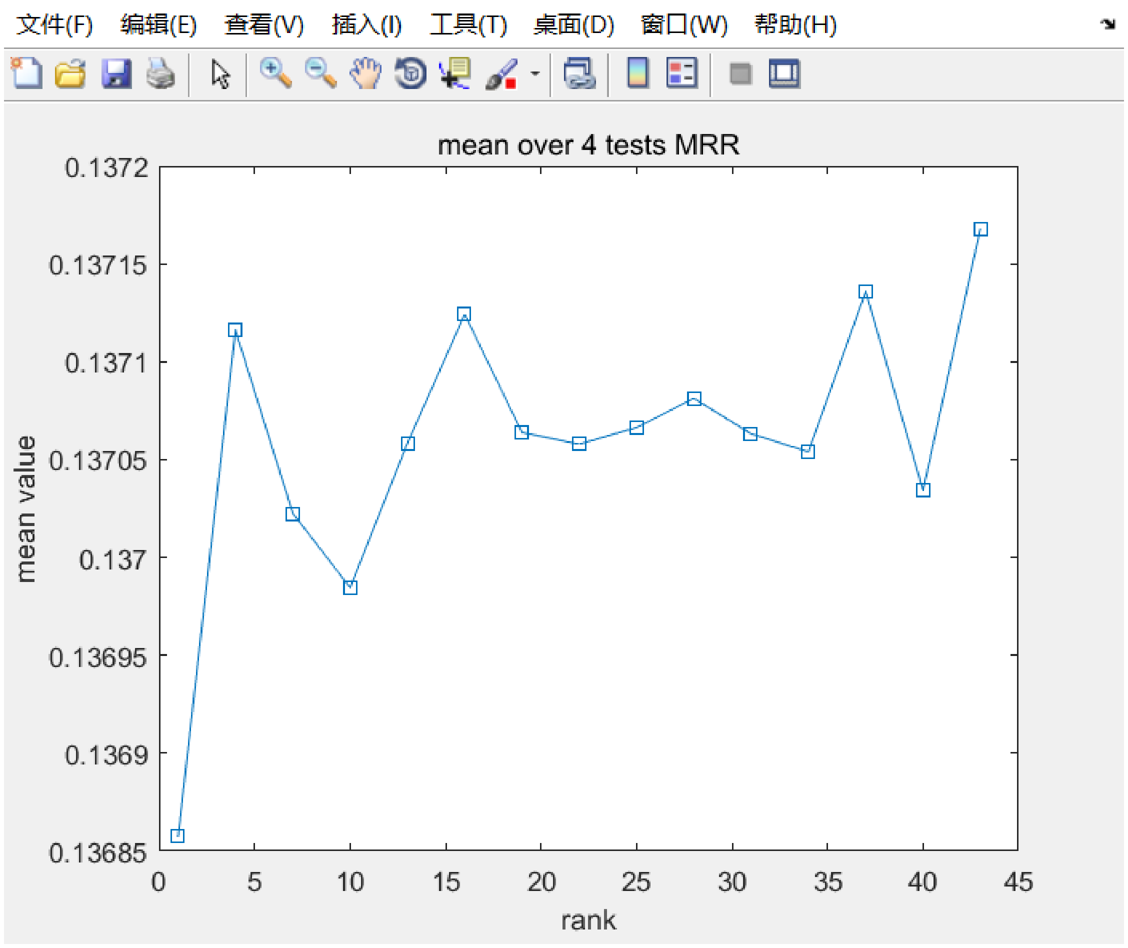
This is the result for rank from 1 to 43 with step size 3. And λ, as mentioned above, is set to 0.1. The graphes are based on 4 times of individual tests.



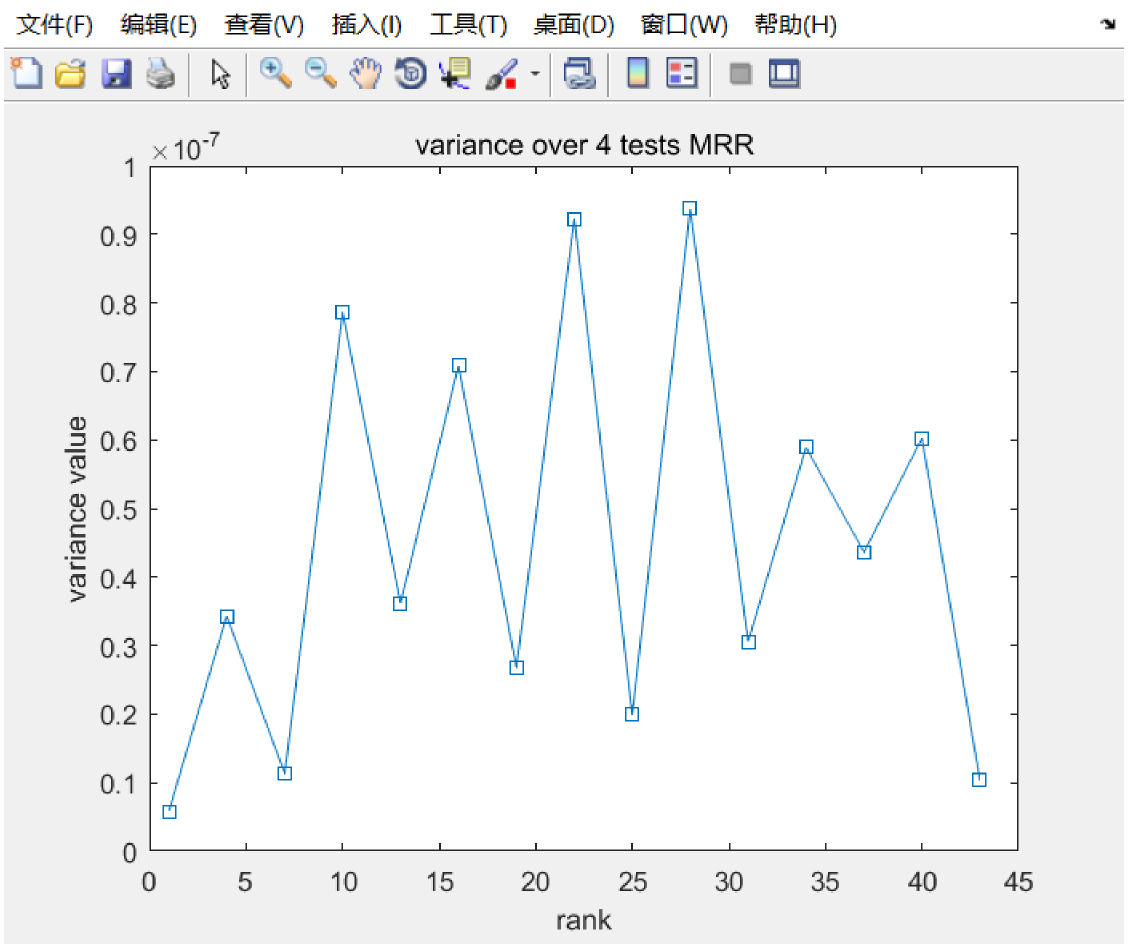
In this graph, we can see that when rank equals to 16, the RMSE is minimal with value 0.8299. And thus, to have best performance of RMSE, one should set the rank to 16 and λ to 0.1.



In this graph, the variance of RMSE gradually increases as rank grows. However, though variance is increasing, the value of variance is still very small. As in every time, the initialization process of splitting of test set and train set and intial value of matrix V and W are all randomly selected. We can ssy this algorithm is robust and has a good performance of convergence for RMSE.



Unlike RMSE, the curve fluctuates more. But still, we can see the trend of MRR curve is increase first and then become steady. However, the growth of MRR value is too tiny, and thus, we can say that for these ranks, they have very similar performance of MRR.



Just like the varaince of RMSE, the variance value of MRR is very small, even smaller. This proves that the algorithm is also robust and has good convergence performance on MRR.

All above, as RMSE decreases first and then increases, MRR increses first and then steady at a level. Plus, our target is to find a rank which gives minimal RMSE and maximal MRR, our choice of rank should be 16. Consequently, the final choice of parameters are $rank = 16, \lambda = 0.1$ and its performance is $RMSE = 0.8299, MRR = 0.1371$.