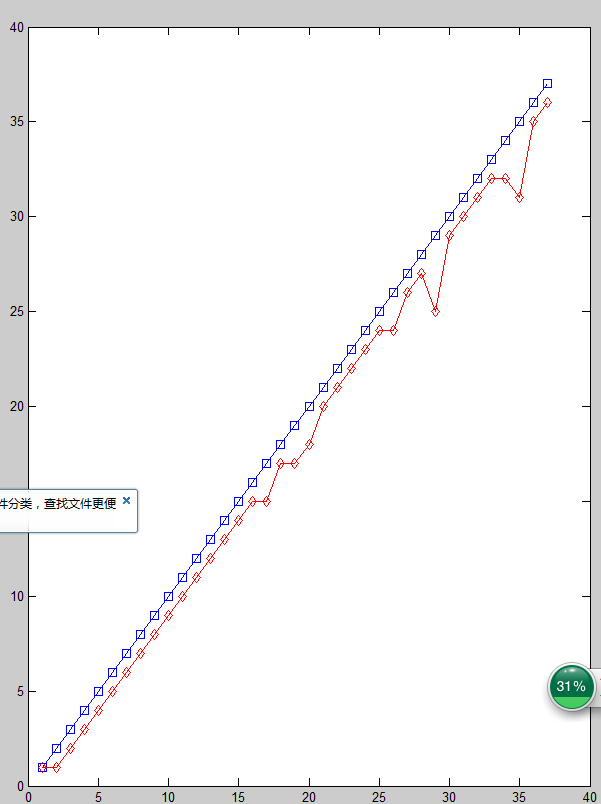
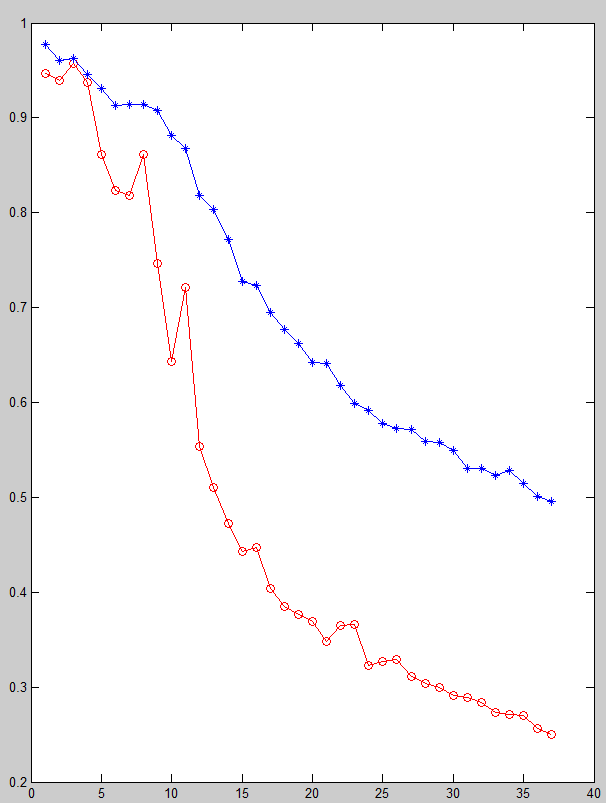
I choose the one who performs best among all 49 results: not only take care of the error rate, but alse look upon on the rank.

However, I found that it performs worse than that of paper KDD13 in LDA on MNIST dataset. The experiments of KDD13 shows sparsity of W on MNIST dataset, though we don't have.I wonder if I can choose the one who has a worse error rate( better than USSL) but has a low rank phemonemon. I'll do it later to check my idea, either using the same lambda1 and lambda2 or not, I'm crazy, I just want a better performance.

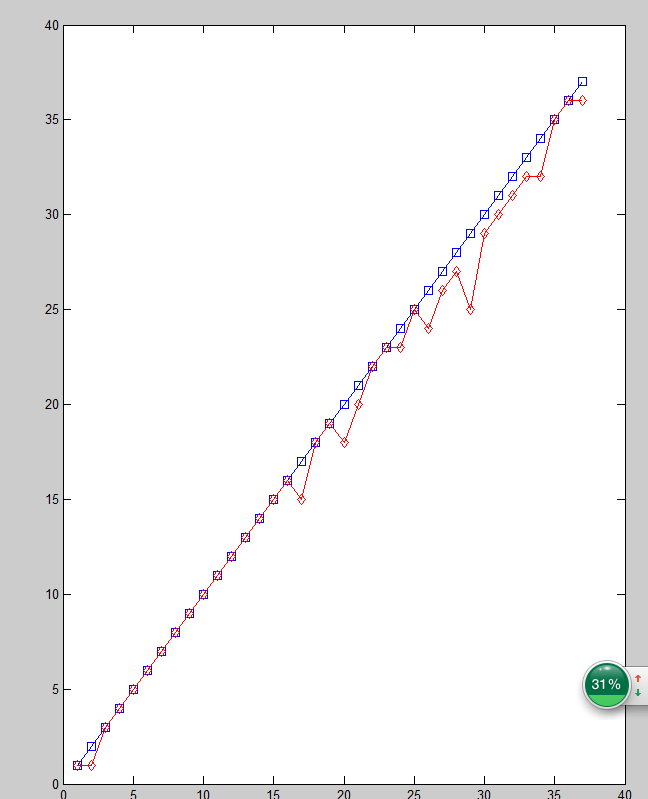
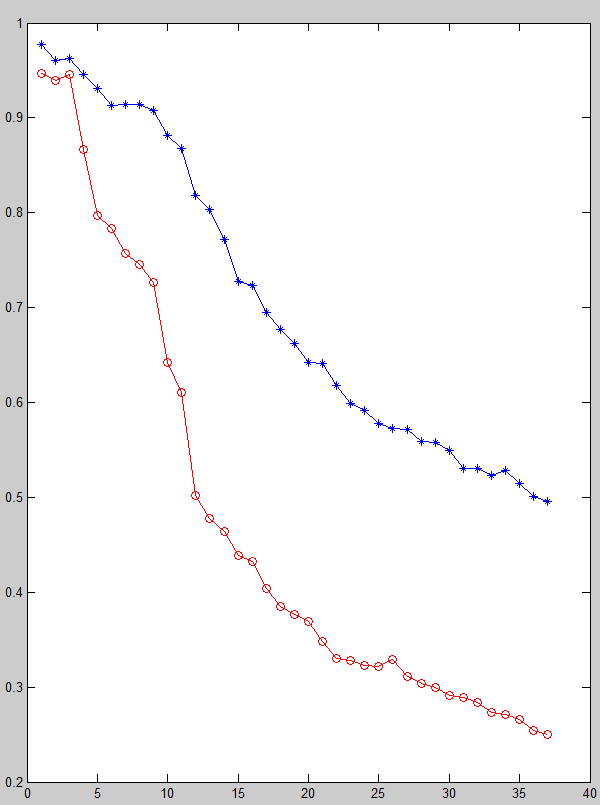
I’ve changed the code, choosing the best result from all dimension with all possibility of all lambda1 and lambda2. I choose the priority of factors that may affect the final output: rank, smooth, superiority than USSL.   
And I choose Yale-B have a try. The results are as follows. 

I saved the result in rank\_first.fig and YaleB\_\_LDA\_rank\_first.mat

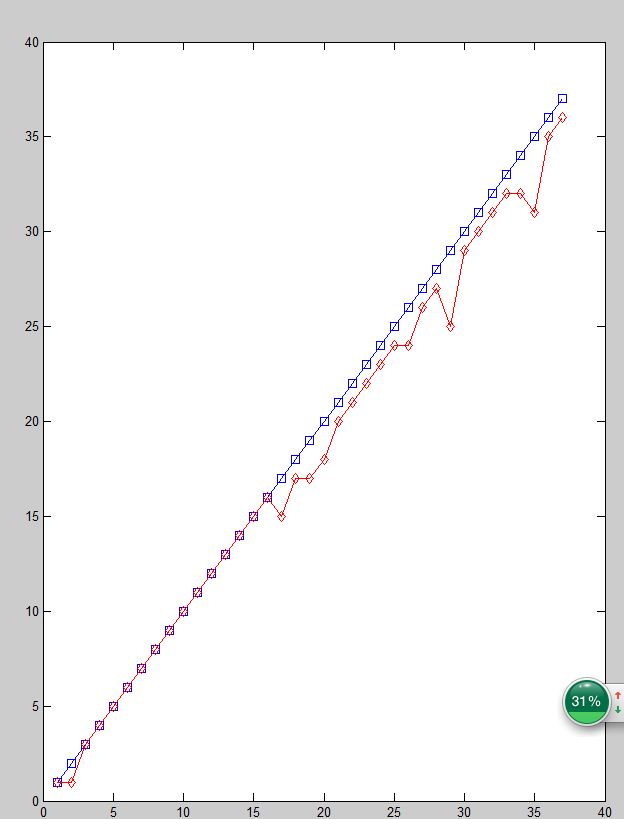
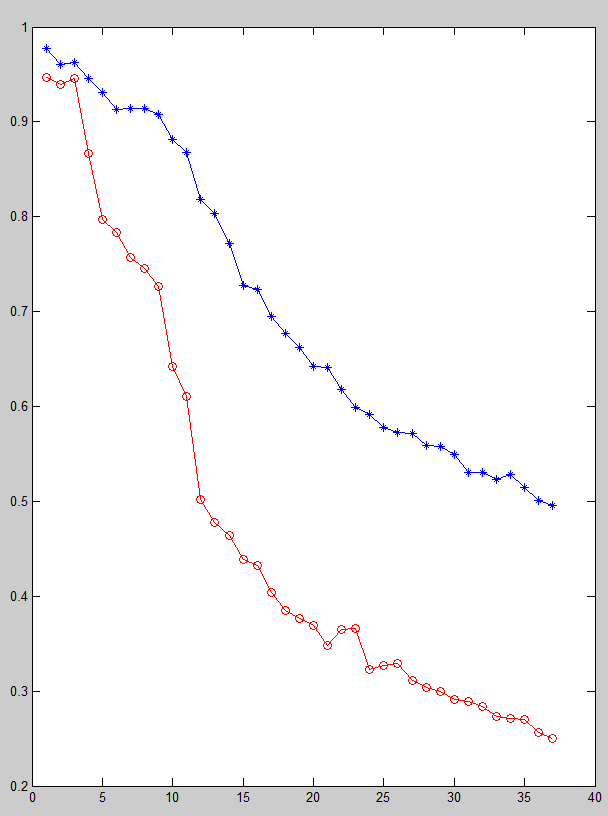
==========================change 2===================

I change the superiority of the three factors mentioned above to smooth > rank >superiority

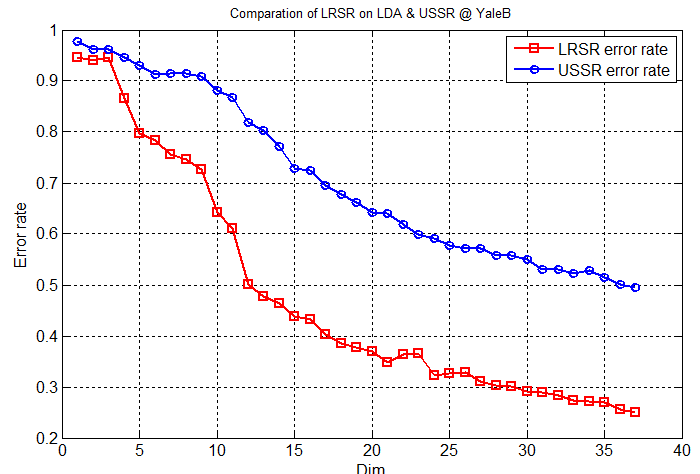
The results are shown below:

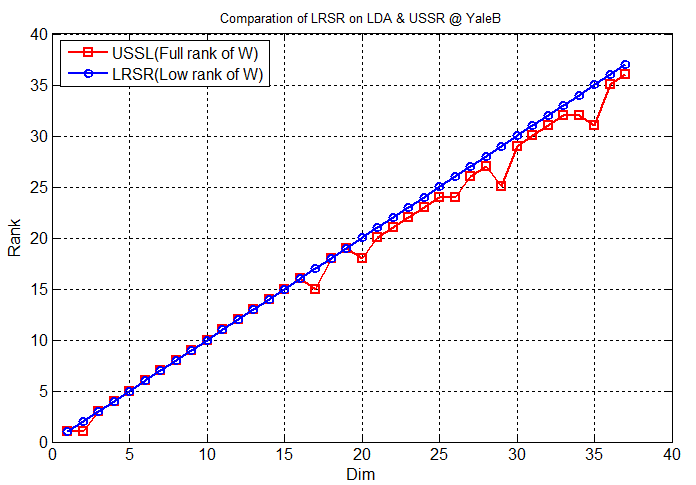


I got an idea, if I can make a trade-off between the two solutions above: the front half dimensions use smooth-first solution and the last half dimension use rank-first solution.



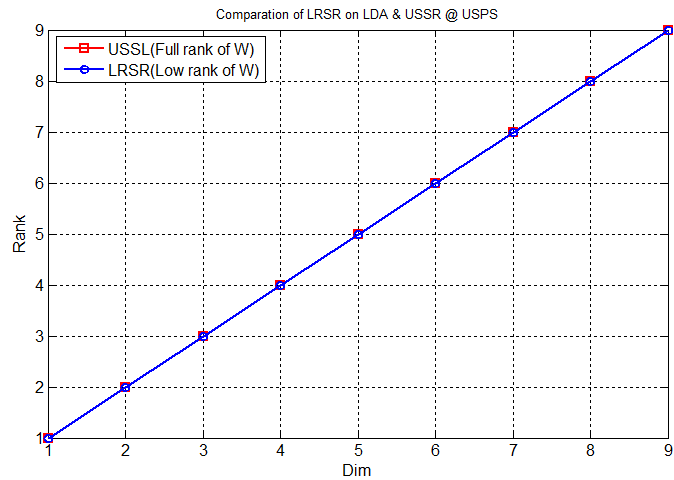
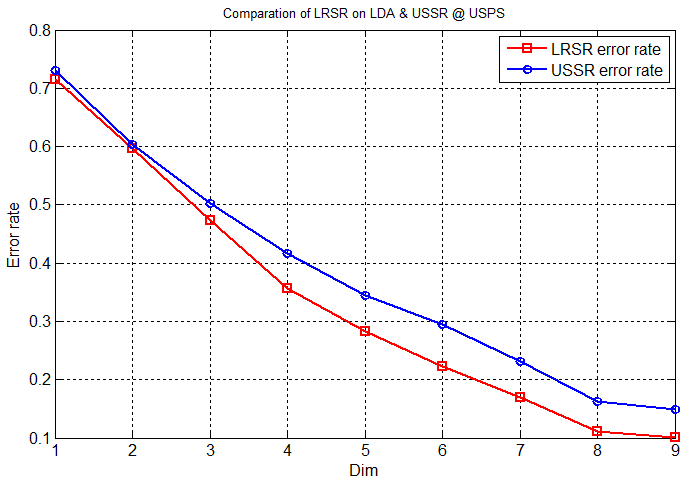
I’ve changed my idea, I want choose the former 25 using smooth\_algorithm and the rest using rank\_algorithm.



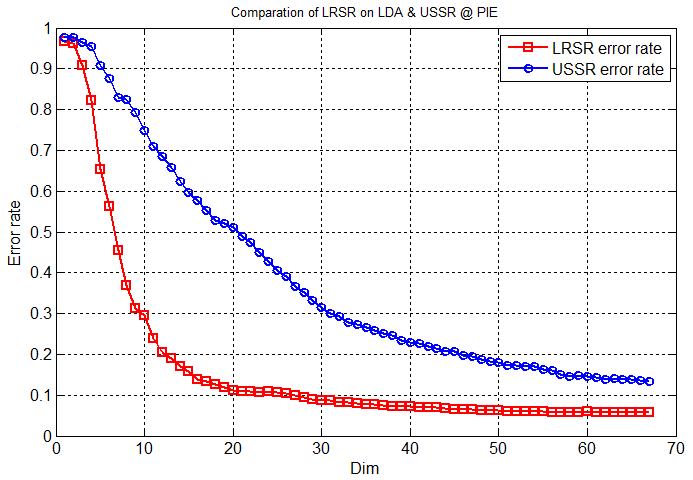
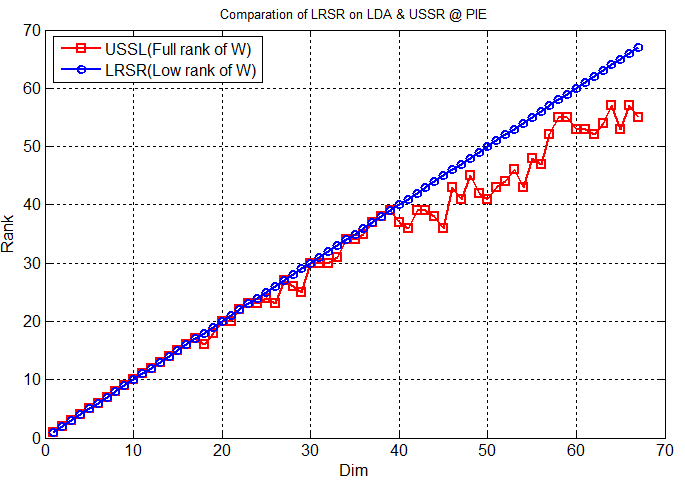


The figure above shows that the error rate plot performs good, well the rank plot performs a little worse.

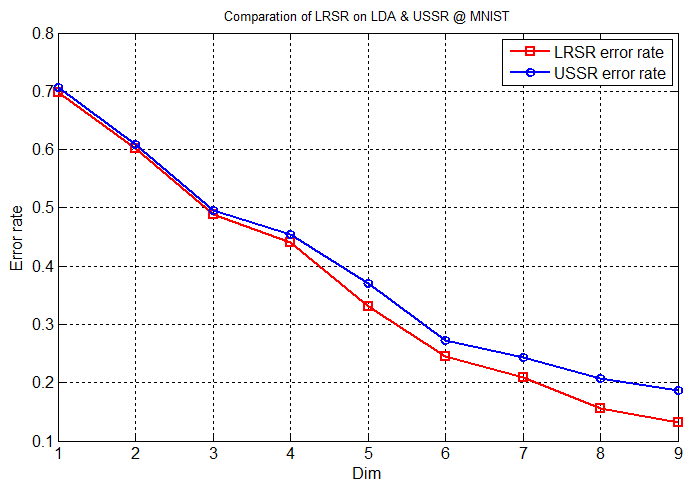
USPS:

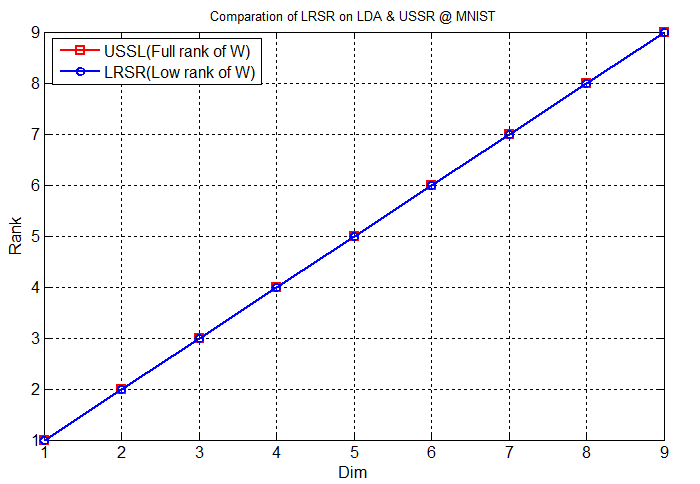
 

PIE: change the trade off from half and half to former 40 and the rest

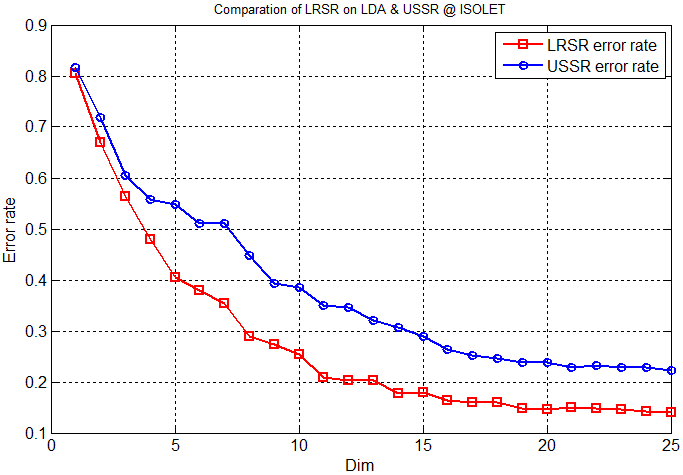
MNIST:

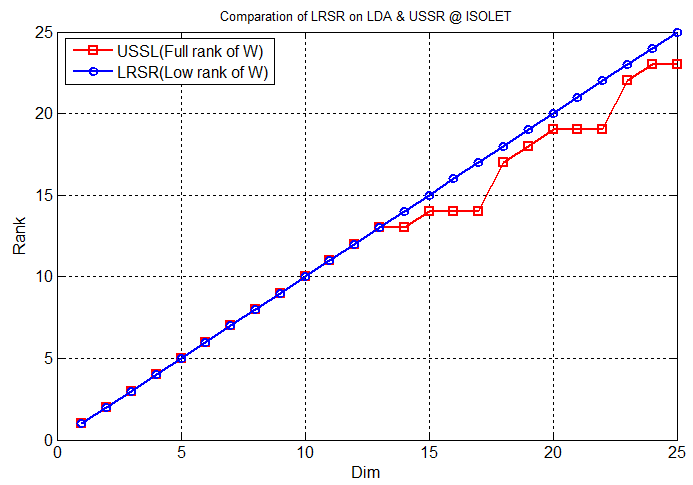




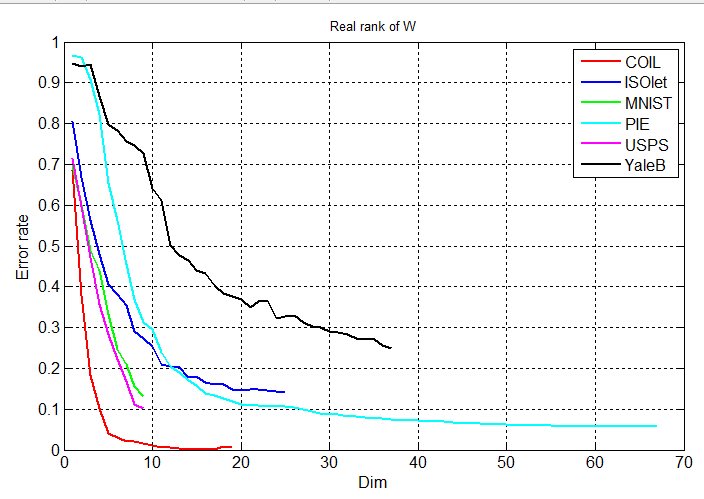
ISOlet: change the trade-off from half and half to the former 15 instances and the rest

COIL20:



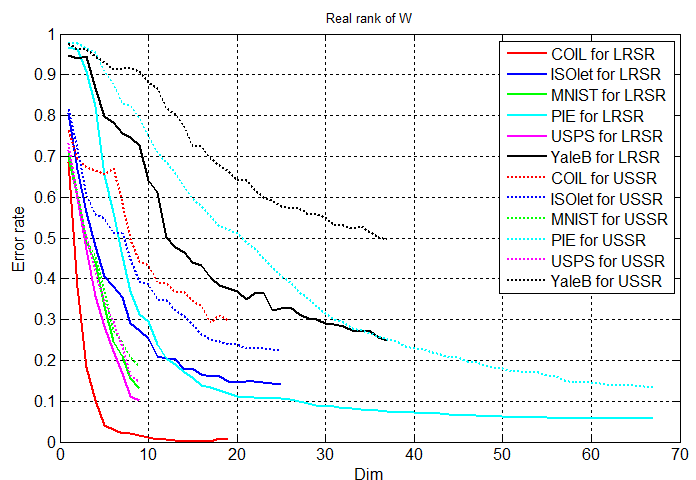


I draw the figure of all the error rate of all datasets.



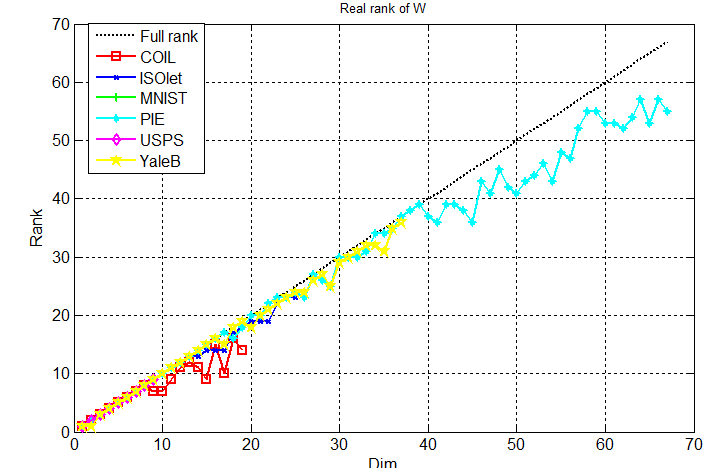
Error\_rate\_all.fig

And make a comparision with USSL



error\_rate\_all\_withUSSR.fig

All the rank situations of all datasets.



rank\_all.fig

**LPP**

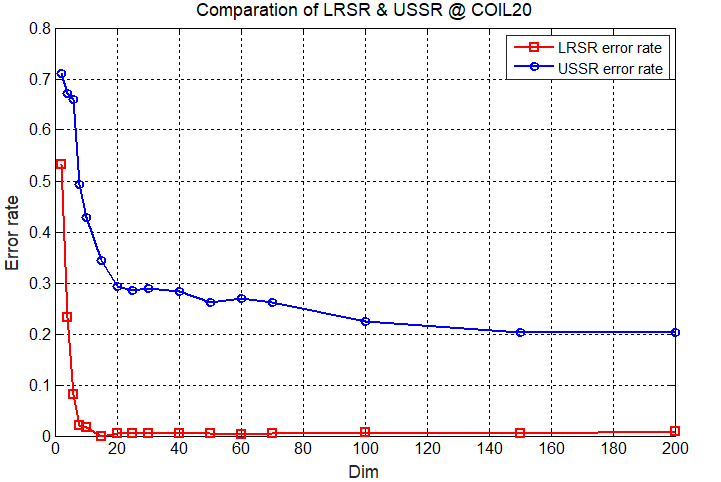
As for LPP algorithm, We choose the result who carry the best performance among all 49 situations of different lambda1 and lambda2 as the final result. We embed the dimension of instance into spaces with 1 to 200 dimensions. We use a set of cardinality :

[2:2:10,15:5:30,40:10:70,100:50:200] to simulate 1 to 200 dimensions. And for each dimension that larger than 20, we pick up average 20 dimension as the intermediate rank that is small than the dimension we embedded into.

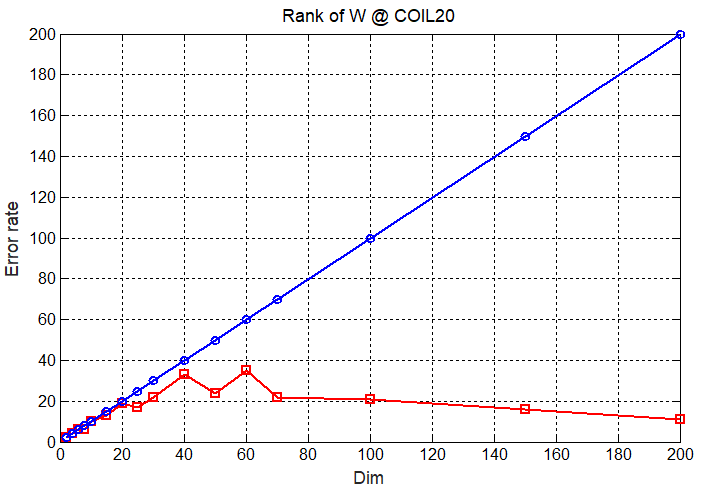
The final results are saved in the LPP -> final folder in LRSR\_results directory.

The figures are showed as follows.

COIL20

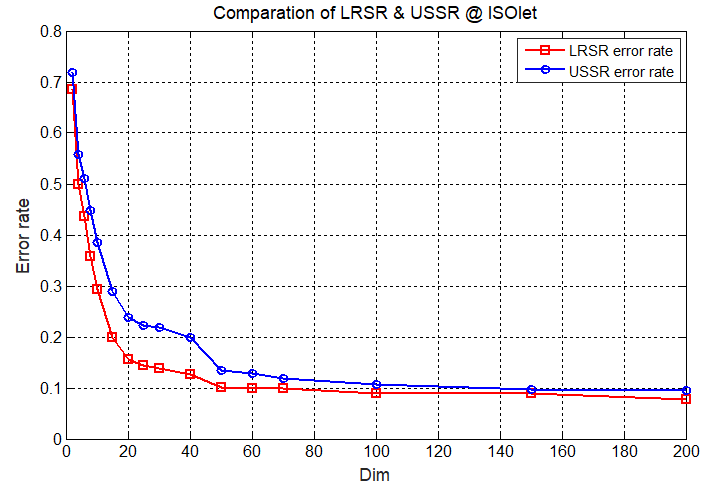


The error rate of COIL20 using LRSR and USSR way

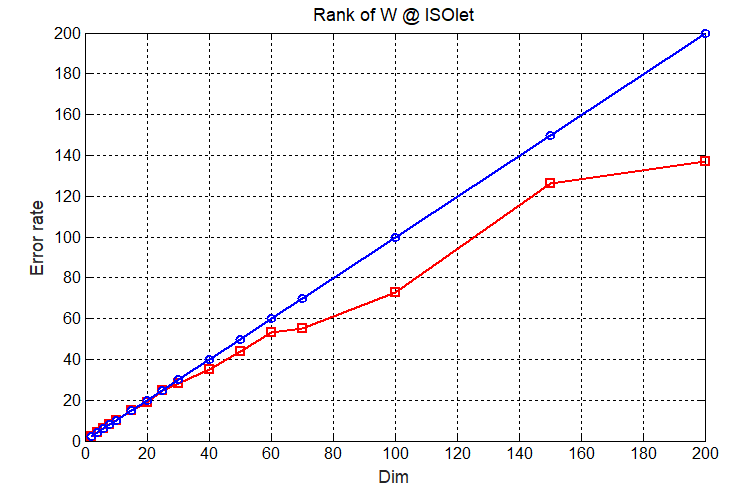


The rank of each dimension of COIL20 using LRSR and USSR way

**ISOLET**

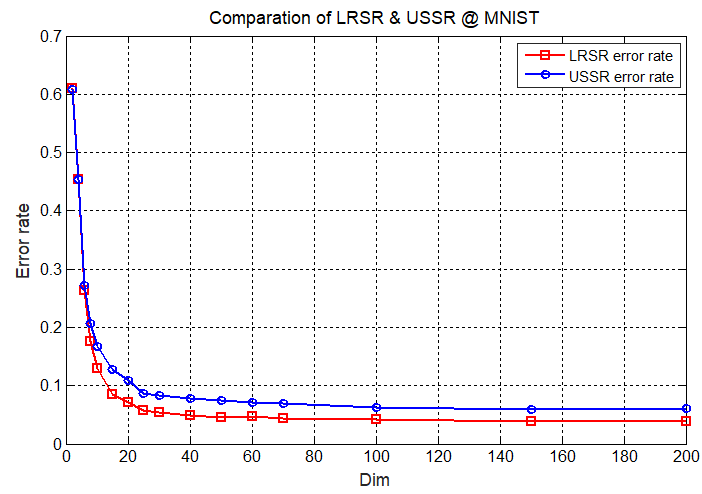


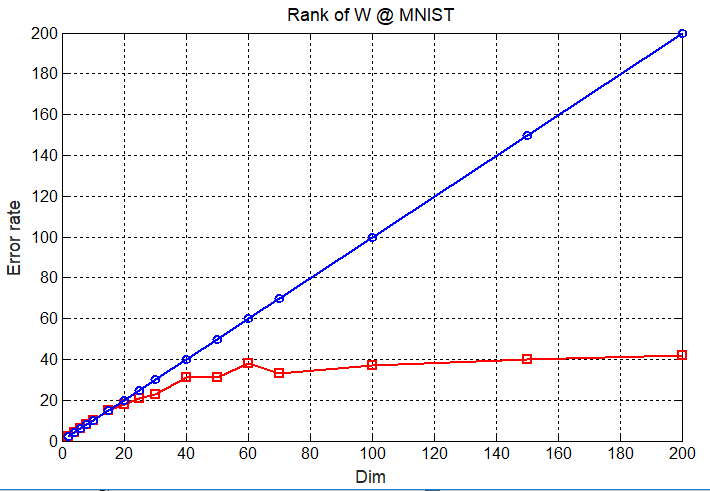
The error rate of ISOLET using LRSR and USSR way



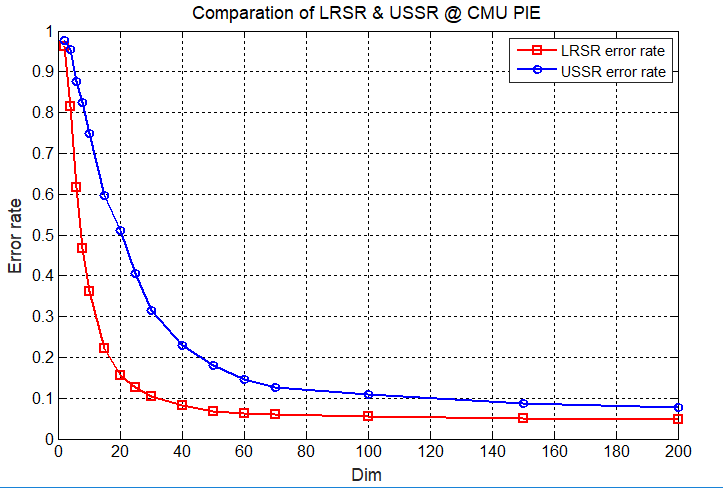
The rank of each dimension of ISOLET using LRSR and USSR way

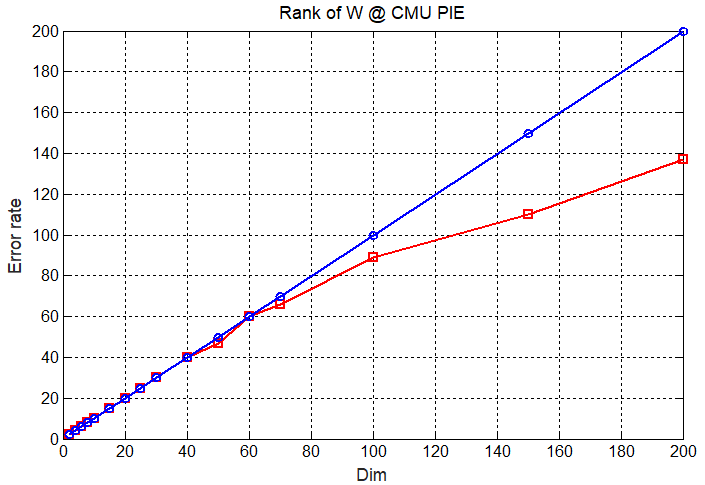
MNIST



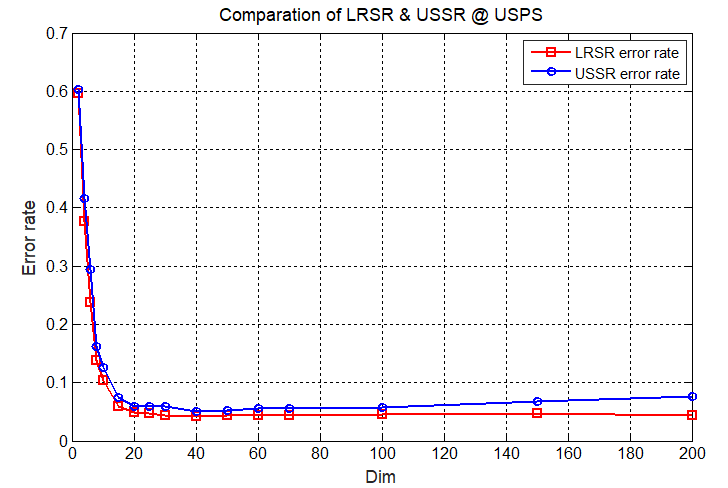


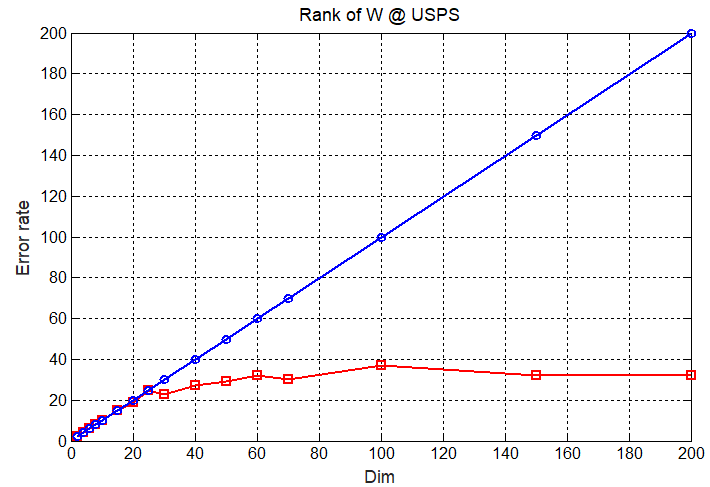
PIE



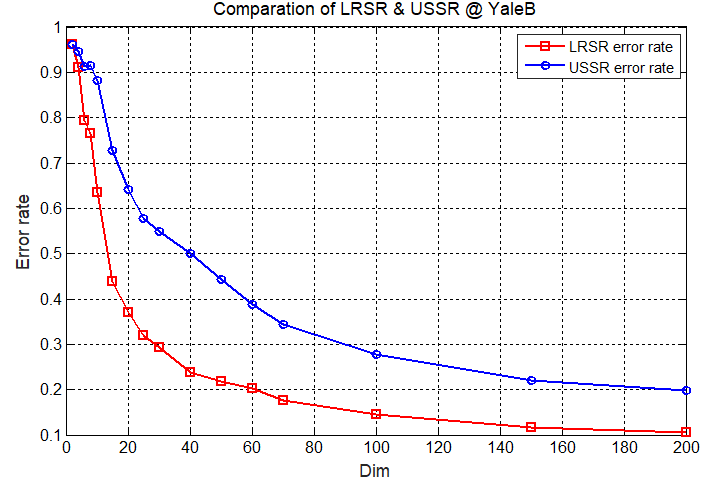


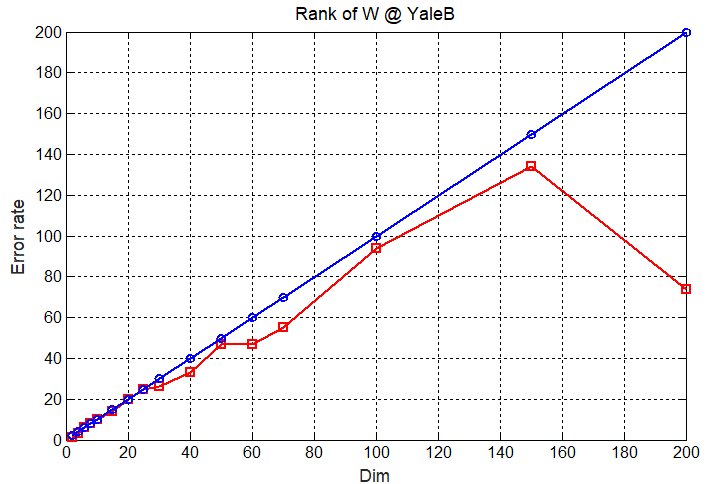
USPS





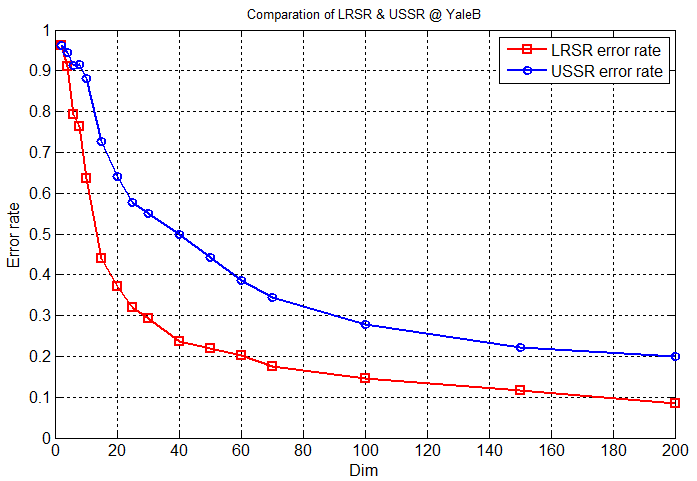
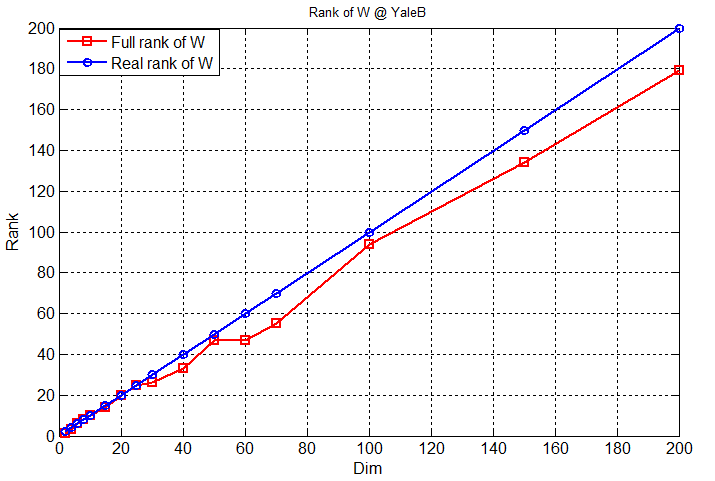
YaleB





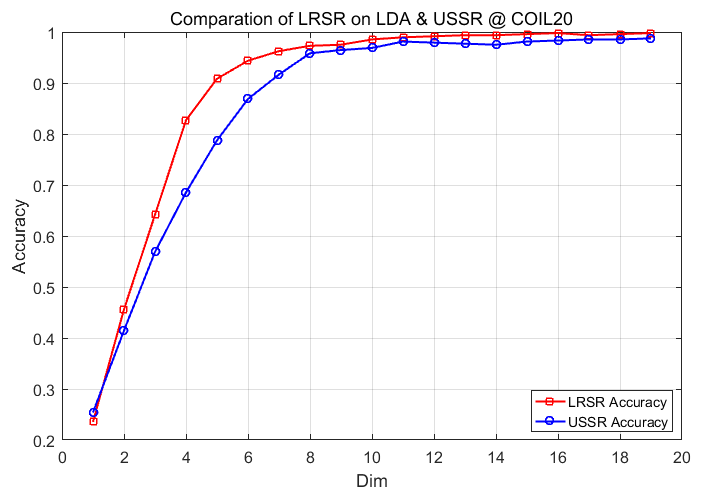
It performed not good at 200 dimension, so I decided to choose a better one to replace the error rate and the rank of 200 dimension.

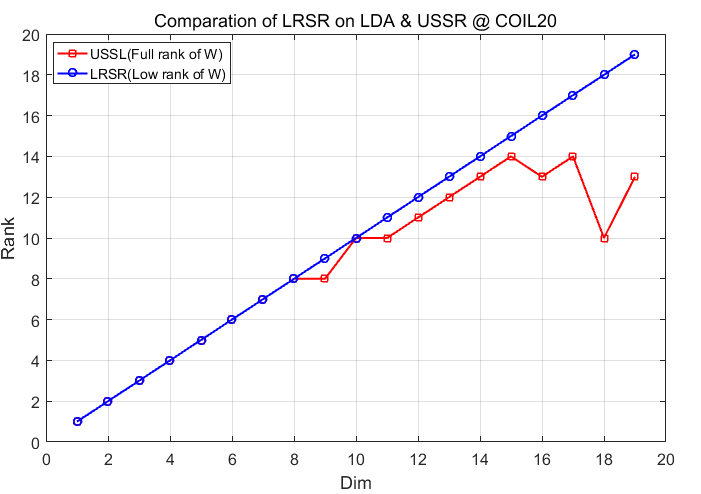
I choose the one whose lambda1 equals 10 and lambda2 equals 1000 using which to embedded into 200 dimension to replace the data drew obave.

 USSR, find a best result from parameters (-10,30): almost the best result from all parameters.

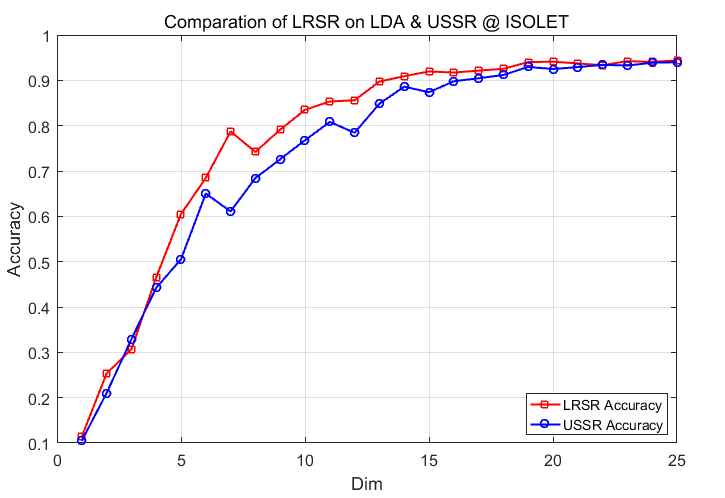
After changed my code, the results are as follows:

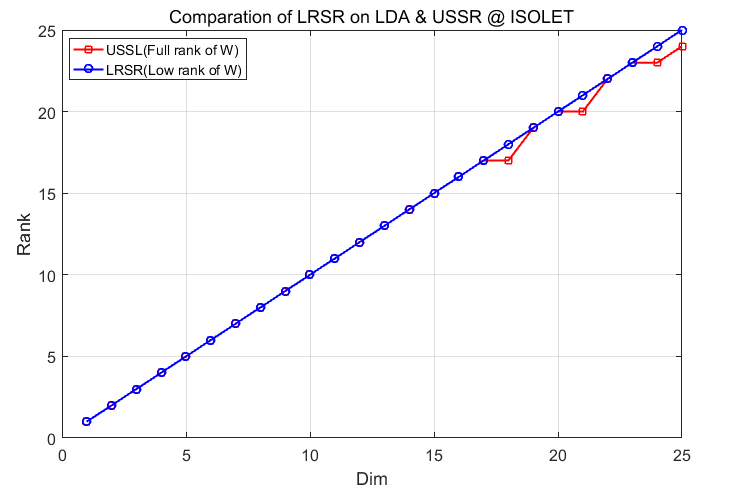
COIL



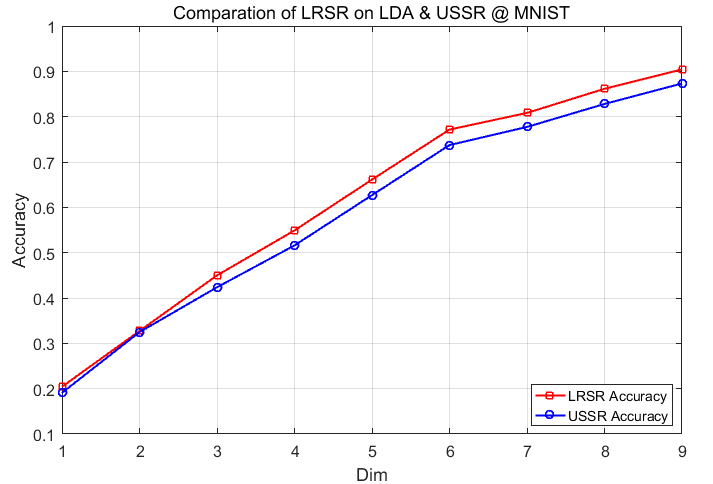


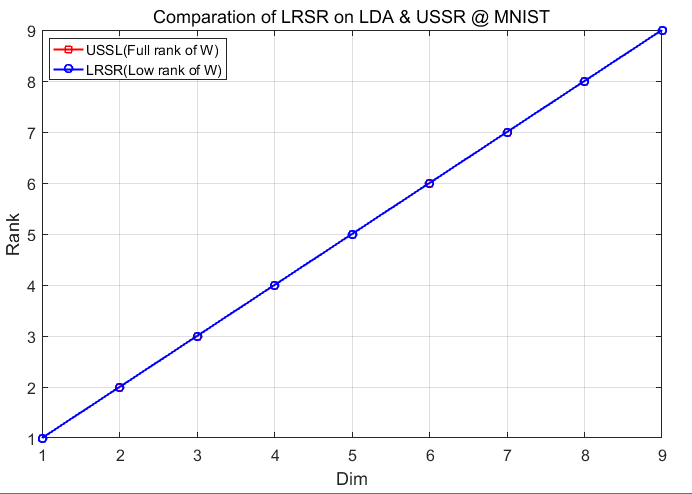
ISOLET



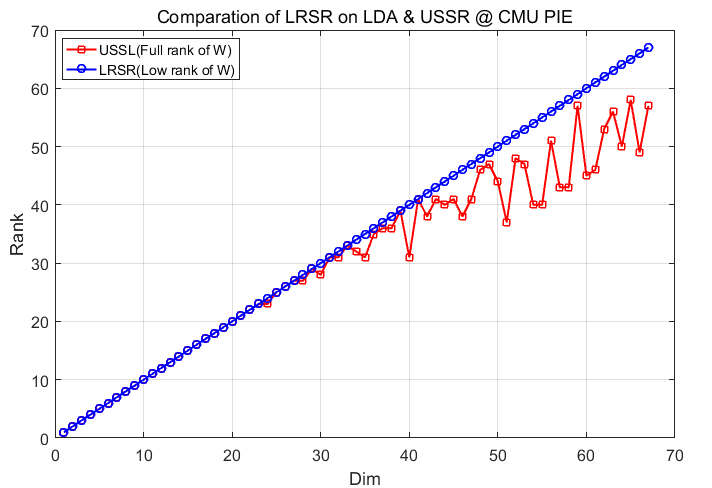


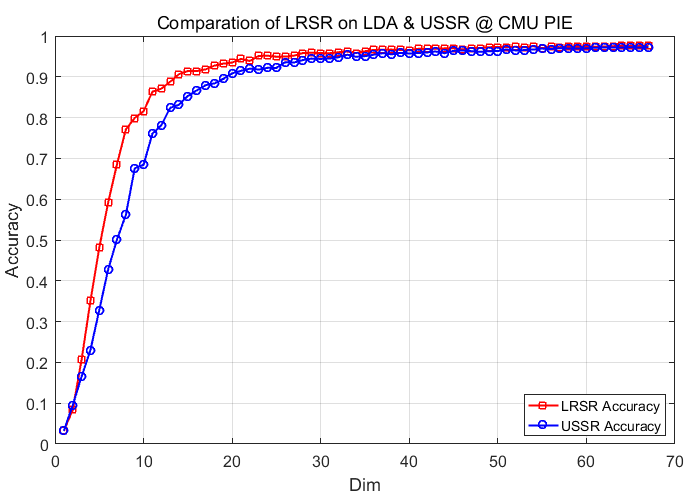
MNIST



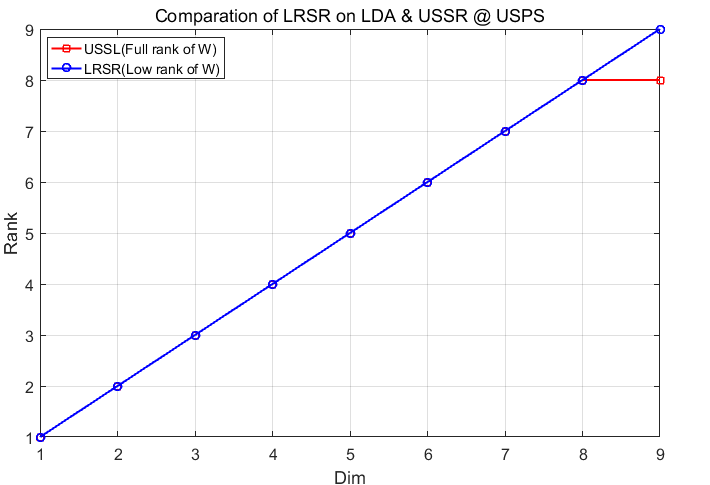


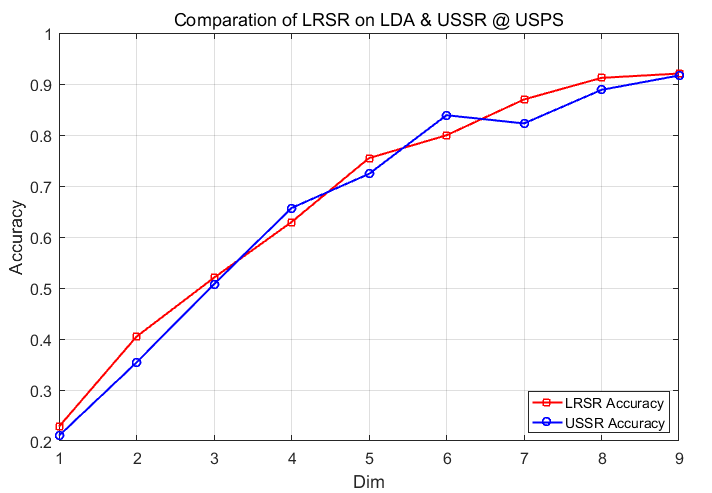
CMU PIE





USPS





YaleB

LPP

COIL20:

