Data Analytics
Assignment-6
Arman Gupta - Mtech CSA(SRN 19220)

Question 1

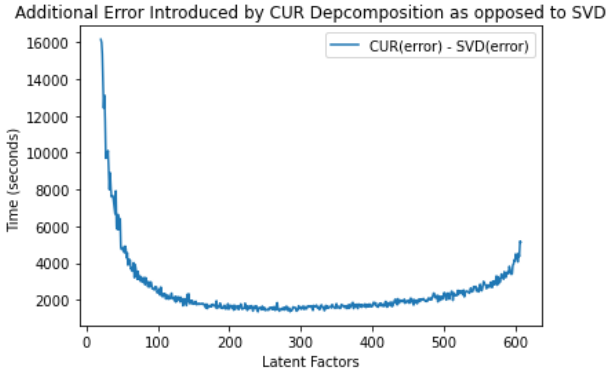
SVD states that any matrix D (represents users * items matrix here) can be factorised as $D = U\Sigma V^T$ and we know that the total variance of the data equals the trace of the sample covariance matrix which equals the sum of squares of its singular values. Equipped with this, we can calculate the ratio of variance lost if we drop smaller σ_i terms. This reflects the amount of information lost if we eliminate them. Here, we want to preserve 90% of the data's variance. Since the total variance is ~ 1345925 and it's 90% is ~1211333 which can be achieved with 240 highest singular values. Thus, I choose to capture 240 latent factors.

Question 2

In order to calculate the error, I have used the Frobenius norm to calculate the distance between the original rating matrix and the reconstructed matrix ie. error calculation in case of

- SVD : $|| D - U\Sigma V^T||_F$
- CUR : $|| D - CUR||_F$

The error observed in the case of SVD is 366.43 while the error in the case of CUR Decomposition is 1714.20m, the additional error introduced by CUR decomposition as opposed to SVD is 1347.77.



Plot 6.1

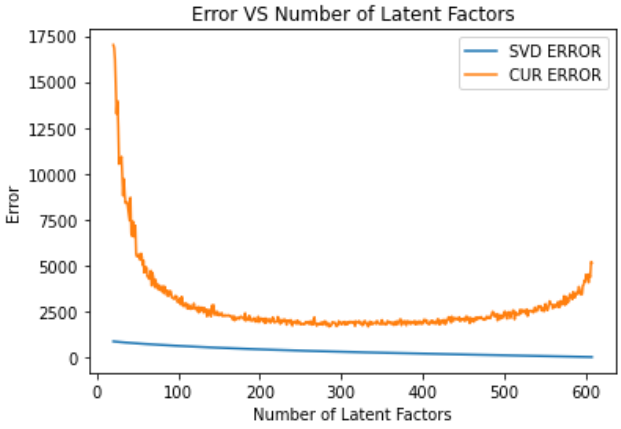
Plot 6.1 is the addition error introduced by CUR Decomposition as opposed to SVD vs the number of latent factors.

Question 3

For 240 latent factors, the running time of the following algorithms was-

- SVD - 0.52 sec
- CUR - 0.21 sec

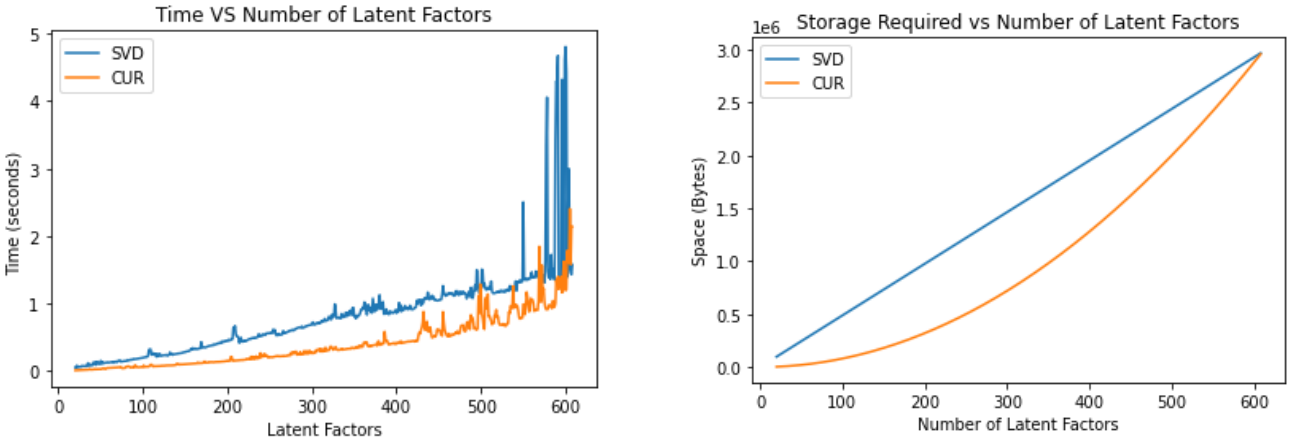
Question 4



Plot 6.2

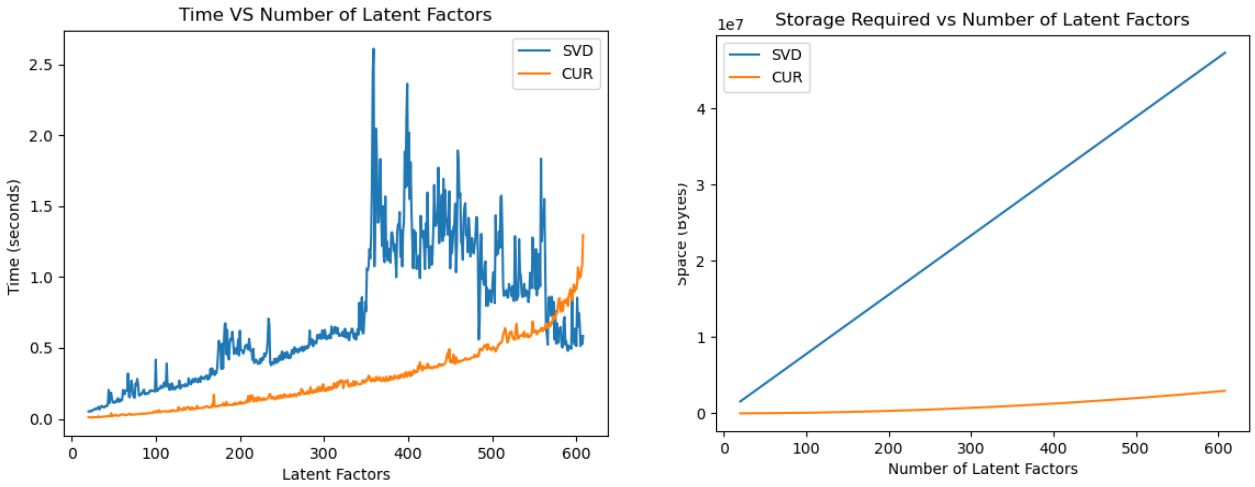
Observation: In Plot 6.2, the error of SVD is always less than CUR decomposition which should be the case according to the Theorem in Lecture 2 - slide 21. The minimum error in the case of CUR decomposition is less when we select the number of latent factors in the range of 200-300.

Question 5



Plot 6.3(a)

Observation: We are taught in class that the time complexity and space complexity of CUR Decomposition is less than that of the SVD. In plot 6.3, we can observe that the time (left plot) and space (right plot) taken by the CUR decomposition algorithm is less than the time taken by the SVD algorithm which verifies the complexity statement made about these algorithms in class. For the large latent factors, the cur decomposition time may shoot because the high probable columns and rows are picked up frequently and the algorithm has to deal with them which increases the time taken by cur decomposition.
Note: I have run the code on two different processors. Though the observations remain the same the plot may differ depending on number of processes running and various other processor depending factor. For eg.



Plot 6.3 (b)

Plot 6.3(a) is achieved when running the code on Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz while Plot 6.3(b) is achieved when running the code on Intel(R) Core(TM) i5-8500 CPU @3.00GHz

Question 6

PQDecomposition Model -

The model with the hyperparameters k, learning_rate and lambda is trained for 40 epochs for a low learning rate or 10 epochs for a high learning rate. The hyperparameters I have tried and the loss I have got in training and test set are given in the below table. Apart from the square loss, I have used RMSE (Root Mean Squared) metrics to check the efficiency of the model on the test dataset.

Epochs	# latent factors (k)	Learning rate	Regularizer coef (lambda)	Train loss	Testing Loss	Testing RMSE
40	50	1e-3	1e-4	36249.99	29482.23	1.46
40	100	1e-3	1e-4	25991.07	31373.15	1.55
40	200	1e-3	1e-4	15718.43	34290.48	1.7
10	50	1e-2	1e-4	14245.32	30034.08	1.48
10	100	1e-2	1e-4	6065.27	30880.44	1.53
10	200	1e-2	1e-4	1857.86	32750.52	1.62
10	200	1e-2	1e-3	1979.86	32204.96	1.596
10	200	1e-2	1e-2	3454.43	29779.73	1.476
10	200	1e-2	1e-1	34403.97	26511.197	1.314

As we can observe for the above table the lowest testing loss we've got is 26511.197 (last row).

Question 7

Architecture of my neural network is similar to the figure Fig 6.4. In Neural CF layers, I have used only 2 layers with 50, 25 hidden units and tanh activation function followed by an output layer. In the Input layer, I have converted the sparse matrix into a dense matrix before passing it to the embedding layer in order to make the implementation easier. Using the dense input matrix of one-hot encoding, the embedding layer is implemented by multiplying the one-hot encoding input with the embedding matrix.

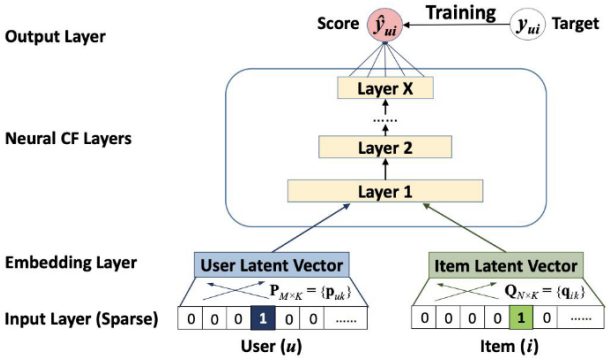


Fig 6.4

Epochs	# latent factors / Embedding dim(k)	Matrix Factorization Test loss (Square loss)	Matrix Factorization Test RMSE	NCF Test loss (Square loss)	NCF Test RMSE
10	50	29482.23	1.46	22869.96	1.1339
10	100	30880.44	1.53	23328.41	1.1567
10	200	26511.197	1.314	22625.21	1.1218

Few Important points-

- Here K represents the number of latent factor in case of Matrix Factorization model while it represents the embedding dimension in case of NCF model.
- In the above table, I have compared the best loss and rmse for matrix factorization with NCF corresponding to three different values of k.
- The learning rate is kept fixed for the NCF model which is 1e-3.
- The batch size in the case of matrix factorization is 1 while in the NCF model, the batch size is 32 (a large batch size is taken in NCF to increase the speed of model training).

Seeing the results from both tables, I have selected 200 dimensions of the neural embedding for users and movies , and also 200 latent factors for matrix factorization