Author: Sayan Chaudhuri

SR No.: 05-01-00-10-42-21-1-19743

Implementation Summary

Objective

- Apply SVD and CUR decomposition on provided dataset
- Find the running and storage time for the two methods and compare them
- Plot error vs number of latent factors
- Apply PQ decomposition on the training data obtained after performing an 80:20 split
- Tune the regularization parameter.
- Scripted a generalized neural CF approach.

Libraries Used

Numpy, Pandas, Matplotlib, Sklearn, Time

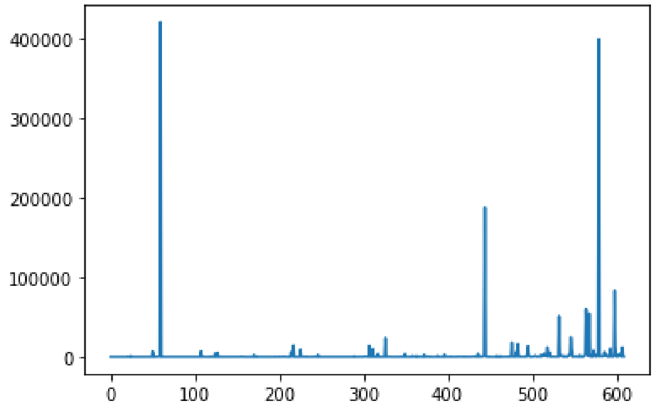
Procedure

- Imported relevant modules and the dataset.
- Made 2 dictionaries corresponding to userID and movieId.
- Using these dictionaries and the ratings data, formed the matrix of user vs movies where in each element was the rating user had provided to the movie.
- The matrix was initialized as a zero matrix.
- Made an indicator matrix which was of same size as above matrix. The values were 1 if a user has rated a certain movie, and 0 if the user hasn't rated it.
- Split the data into Test and Train.
- After the Split, made separate matrices for test and train in a similar fashion as above. This Training matrix was used for performing the PQ decomposition.
- Performed the SVD and CUR decomposition and plotted the error vs the number of latent factors.
- Computed the storage and running time requirements and compared them.
- For the PQ deocmposition, scripted a function " my_PQ (indicator , matrix, P , Q , K, iters = 500 ,eta = 0.002 , lmbda = 0.02) ".
- Trained PQ matrices for 100 epochs and made a log of the errors. The PQ matrices were trained using the training matrix and the training indicator matrix.
- For the neural network CF approach, transformed all the users and movies into one-hot encoded vectors. This I did using 2 identity matrices whose sizes were the number of users and number of movies. The dictionaries made earlier are employed to access the users and movies.
- The input data is a matrix of the concatenations of one-hot encoded vectors of users and movies.
- The ratings are transformed into one-hot encodings of 10 classes.
- These matrices are used for training the neural network.
- Sklearn's MLPClassifier is employed to perform this feat.

#Note: For applying the neural network on entire training set, the time taken is too much. I sliced the dataset of train and found satisfactory results.

#Note : Since the CUR error was very high intermittently, I performed the method 5 times on every latent factor and took into account the one which was the least one.

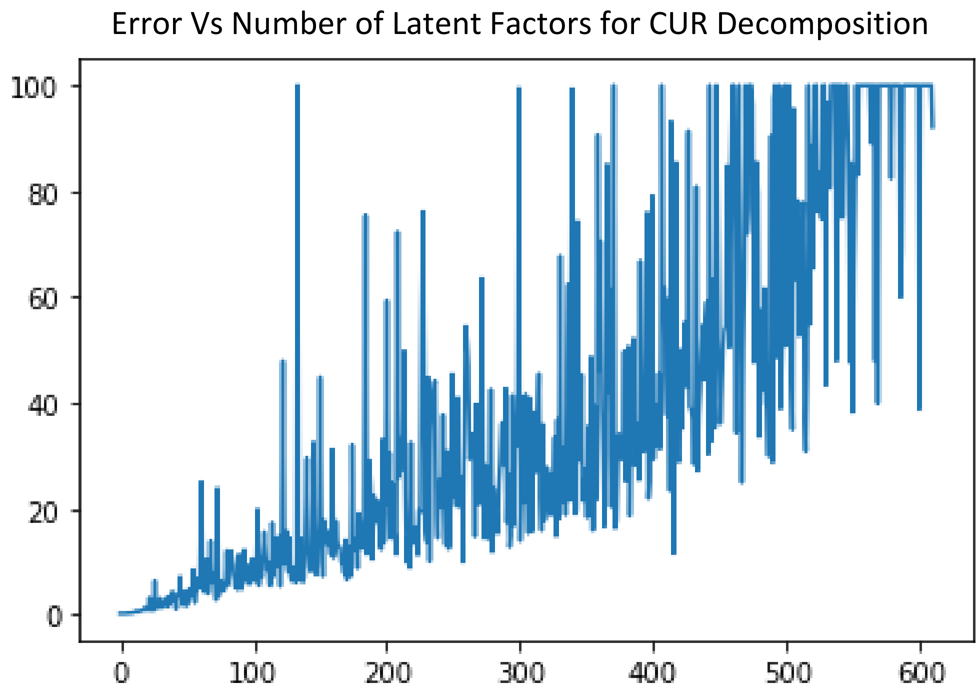
For time calculations, however, I removed the loop so that the time taken is lesser.

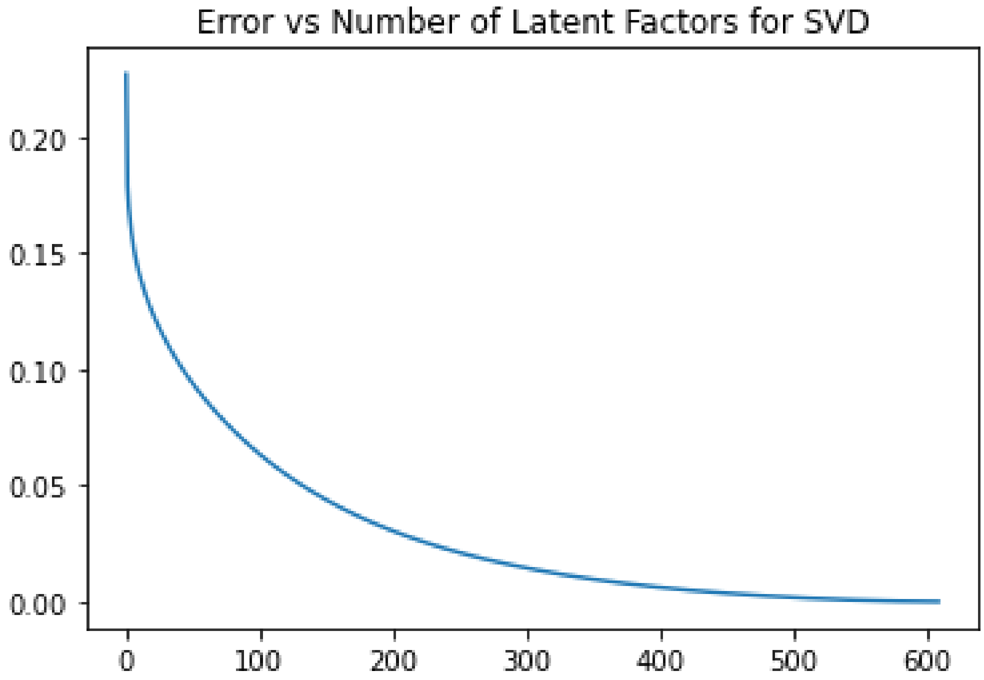


This is the plot I obtain when I perform CUR without putting an upper cap of error values

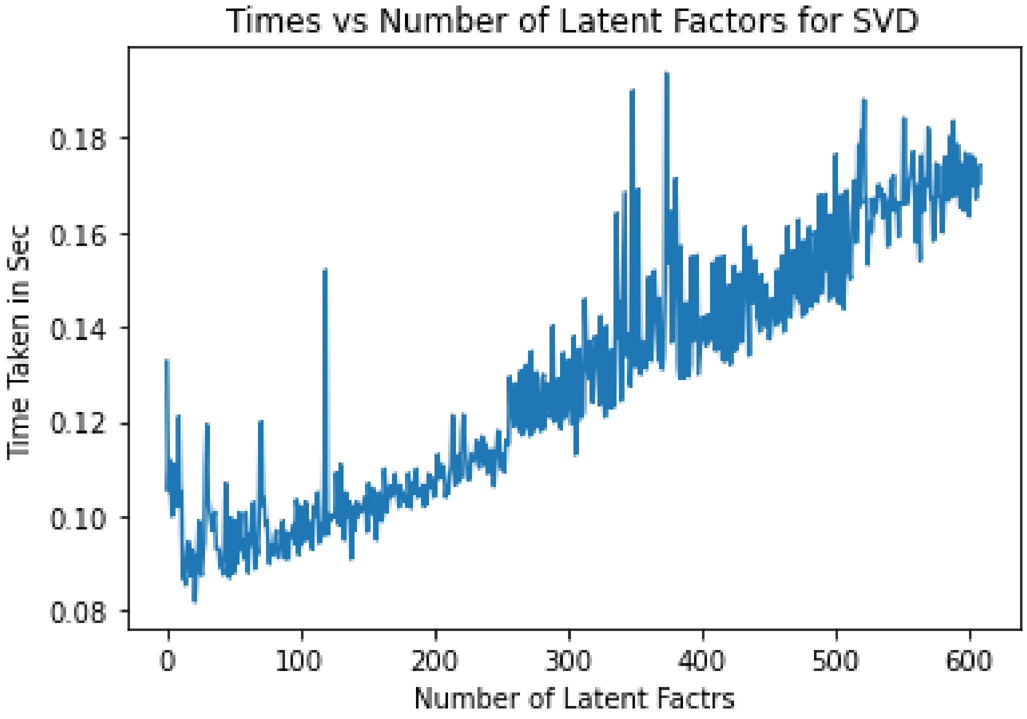
Results

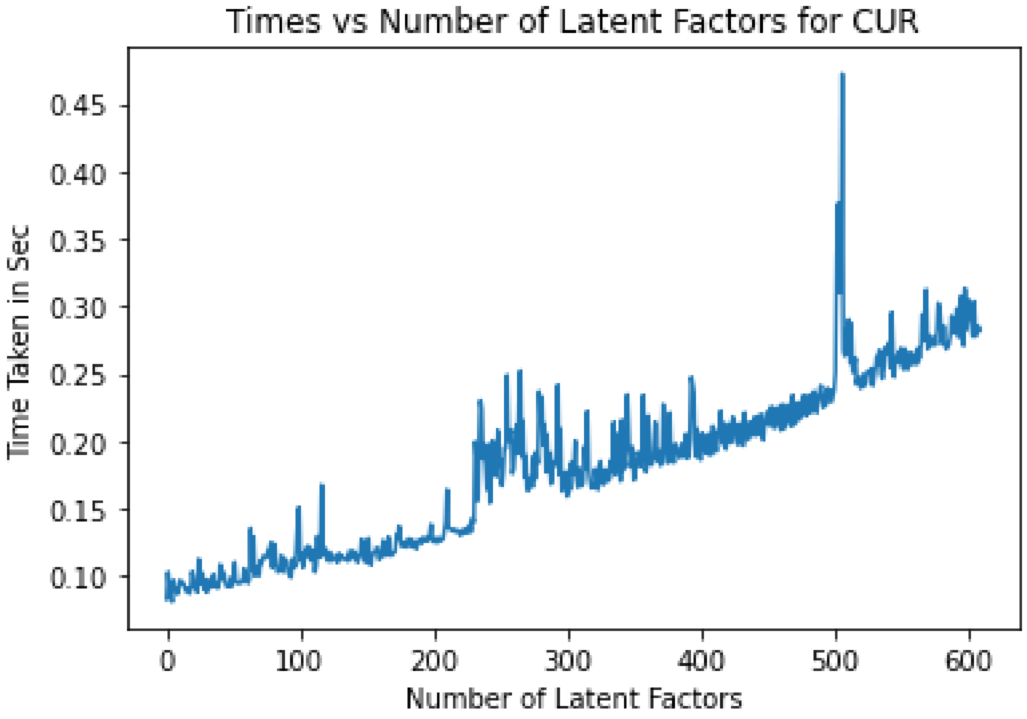
Errors



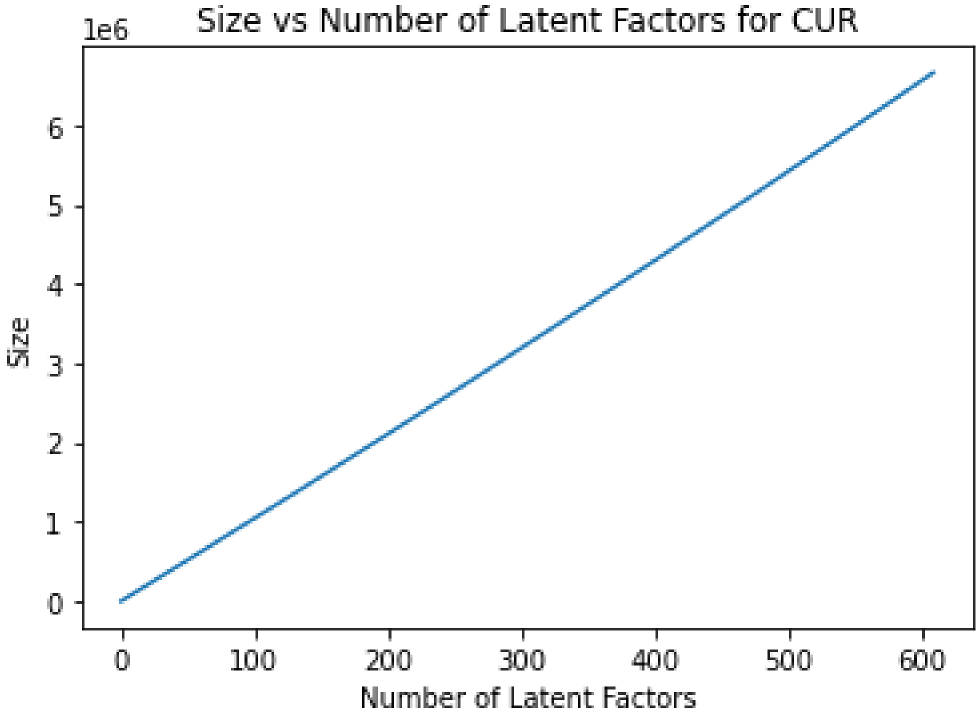


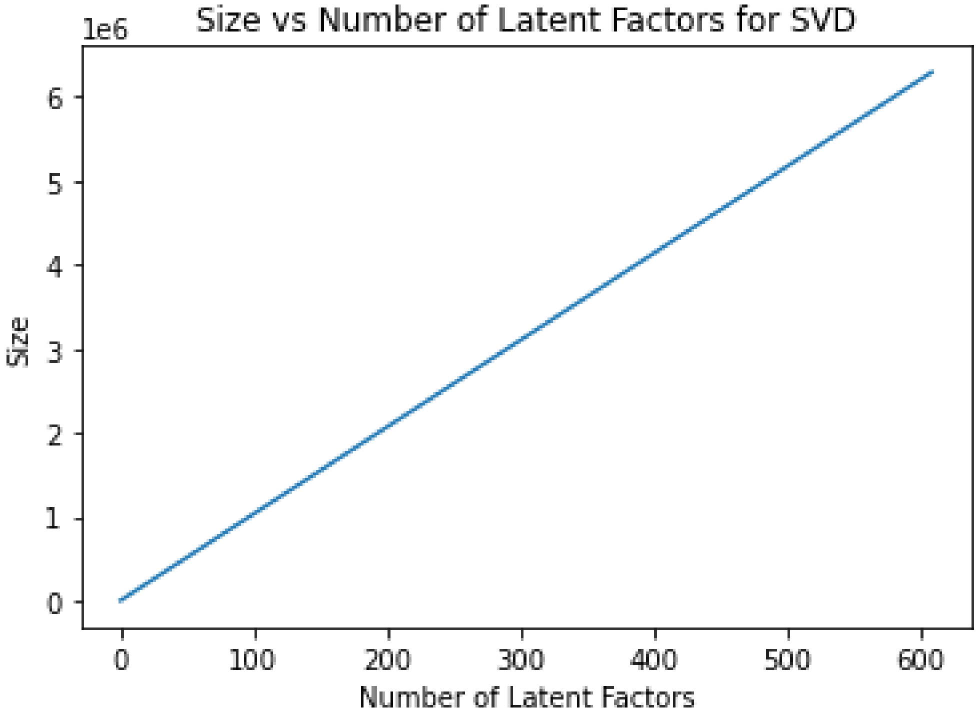
Running Times



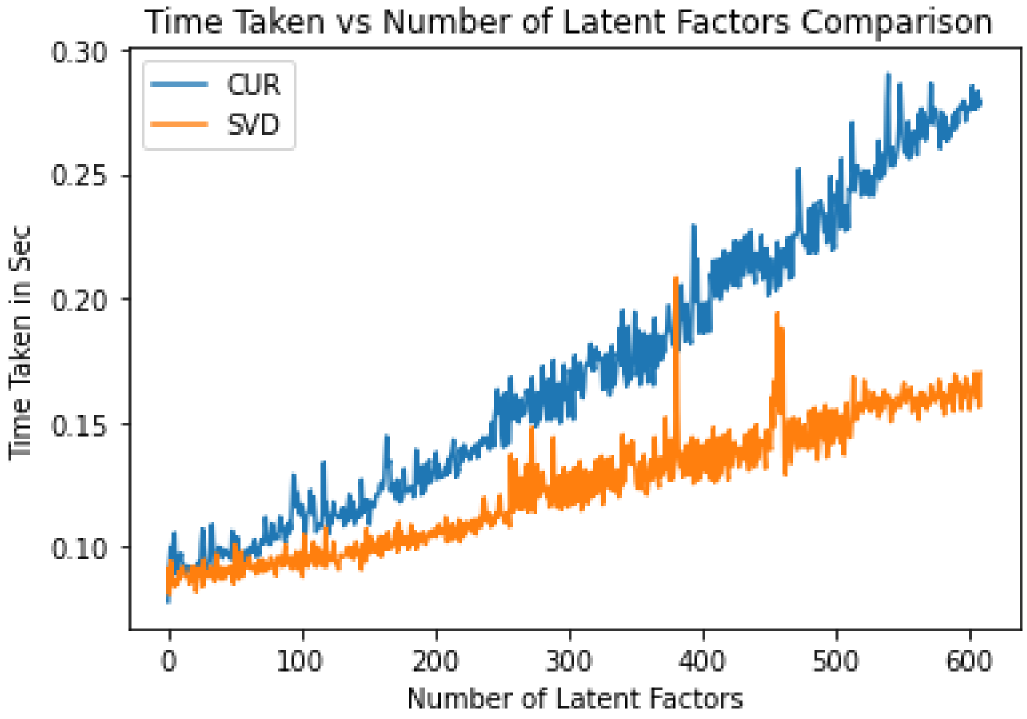


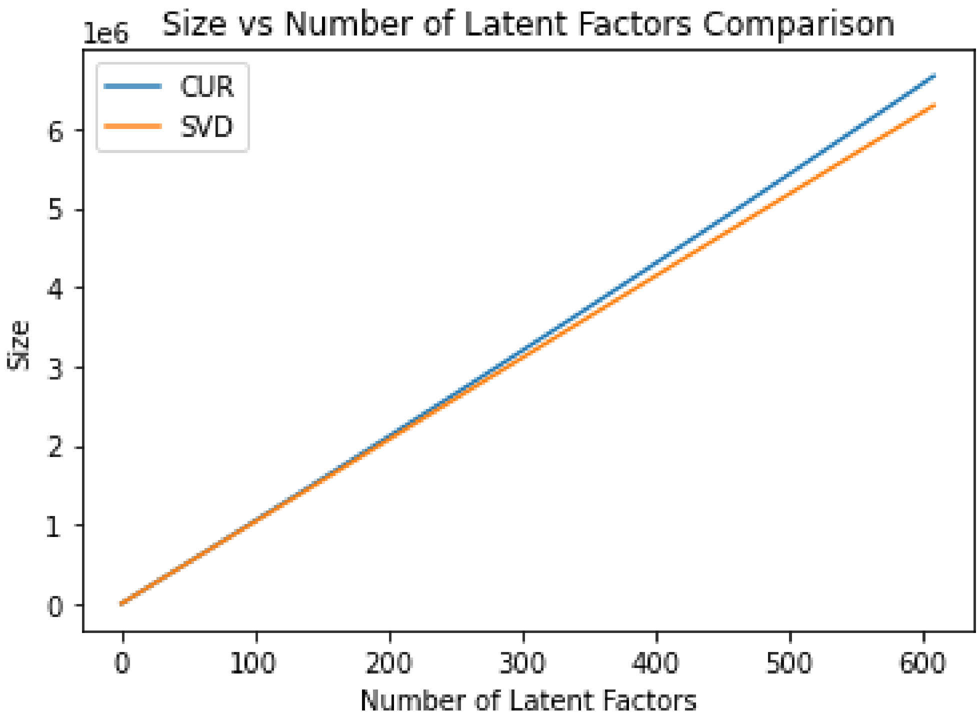
Sizes


Size vs Number of Latent Factors for CUR
1e6
Size
0
1
2
3
4
5
6
0
100
200
300
400
500
600
Number of Latent Factors


Size vs Number of Latent Factors for SVD
1e6
Size
0
1
2
3
4
5
6
0
100
200
300
400
500
600
Number of Latent Factors

Comparisons





My NN Architecture:

Two hidden layers of sizes 100,100. I wanted to experiment with bigger networks but ran out of memory.