Matrix Factorization Algorithms for Recommender Systems

Zhifan Liang
Department of Electrical and Computer Engineering
Washington State University
Pullman WA, USA
zhifan.liang@wsu.edu

Abstract ---- For content-based platforms recommender system is an essential part of their service. To keep users in their platform, the recommender system needs to predict the estimated users' rating on an item based on a sparse known users' rating matrix. In this case matrix factorization algorithm is required to be precise to enable the system to make accurate prediction. This project implemented different matrix factorization algorithms and compares them in terms of accuracy.

1.Introduction

Online platforms are playing a more and more important part of our lives nowadays. To keep the users' attentions is a critical part for the platform to succeed. To achieve this goal, recommender system plays a significant role by making recommendations for the users to extend their time on the platform. It is important to make recommendations accurate and time efficient for recommender system. These two aspects are essential to keep the users in a platform. In order to make a prediction, recommender system usually take a sparse users' rating matrix and factorize it into two matrixes. These two matrixes contain user's feature and item's feature. With those two matrixes the recommender system can make prediction on unknown position in ratings matrix by just multiple the relative user's feature vectors and item's feature vectors. With those reasons, matrix factorization algorithms are widely used in various online platform's recommender systems.

Singular Value Decomposition (SVD) [1] is a popular algorithm for matrix factorization. SVD factorize the algorithm in to 3 matrixes.

$$M = U \times \Sigma \times V^T$$

The matrix Σ stores the weights information of U and V^T which shows the value importance. In this case we can use Σ as a standard to form the two output matrixes. The algorithm selects the top n values in matrix Σ, with n is depends on the feature numbers input. According to the value selected from Σ, the algorithm extracts the needed row and column to generate U and V^T.

In this project, I started by a random approach as the simplest method for solving the matrix factorization. Then I implemented SVD as a common solution for matrix factorization, evaluate both algorithm in terms of accuracy and time efficiency. After that, I tried to optimize the current algorithm and evaluate the optimized algorithm. This approach provides three different implementations for comparison, with two parameters (learning rate and features number) to tune. Those comparison will help us to evaluate algorithms as well as finding the potential problem for a specific algorithm.

2. Problem Setup

This project is intended to implement matrix factorization algorithms and evaluate the algorithms in terms of accuracy. The problem assumed the algorithm only need to train for the current data set of rating matrix, the better fitting for current rating matrix the better the algorithm would be. Finding a good features number is also important in this project. The features number has influence on both accuracy and time efficiency.

3. Solution Approach

To start building a matrix factorization algorithm, I came up with a simple way by simply guessing random values for first user's features vector, then according to the known ratings in the rating matrix, updates the possible item's features vectors. Repeat this process until filling user's matrix and item's matrix, multiply them and verify with known rating values in rating matrix. Update with the same random methods if wrong.

After implementation, I found this approach has major flaws by not making full use of known data and it is very hard to converge. It has an unstable accuracy with the number of iteration increase, this makes it not very useful in practical situation.

To achieve a usable accuracy, I decide to implement SVD. This algorithm updates the factorized matrixes user by user, item by item with singular gradient. One drawback of this algorithm is it run sequentially. In practical application the matrix size is usually large, because the platforms usually have a large number of users and contents. This approach might take too long is practical situation as it is running sequentially.

In the SVD algorithm, the factorized matrixes are updated user by user, item by item. This means it will update every feature of a user, then every feature of an item, then move to next user and item. This approach utilizes two sets of features data while computing, and usually users set, and items set a much larger than a single feature set. To utilize more data, this optimized SVD algorithm update the factorized matrixes feature by feature. The algorithm computes one feature for every user or item before it moves to the next feature. In theory, this approach will utilize more data each calculation. At the same time, it will increase the calculation time.

4. Experiments and Results

To test the accuracy of different approaches I use a small matrix as data set with only 4 values inside a 3 by 3 matrix. I use this data to run through different approaches and collect mean square error function (MSE) [2] every 50 iterations in a total of 1000 iterations to evaluate their performance.

$$MSE = 1/n \sum_{i=0}^{n} (y_i - \hat{y_i})^2$$

Comparing between three algorithms, I found the random approach is very unstable in terms of accuracy. The MSE of it does not steadily decrease with iteration goes up (Figure 1). The MSE of this approach stays between 20 and 38.

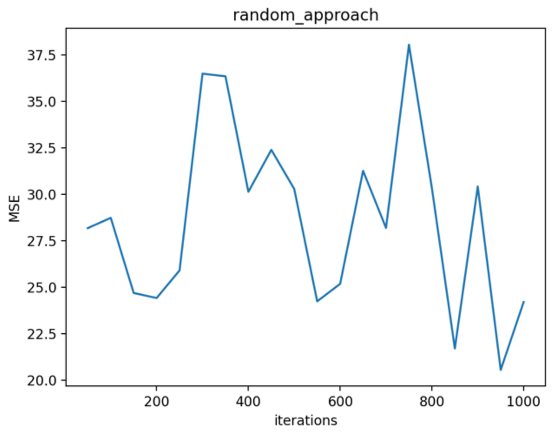

Figure 1

The SVD approach shows the pattern of gradually improve the accuracy. With a smaller size data, the algorithm converges at around 100 iterations (Figure 2). With a larger data size, the SVD approach reaches a low MSE at around 100 iterations, but it is still improving after 1000 iterations (Figure 3).

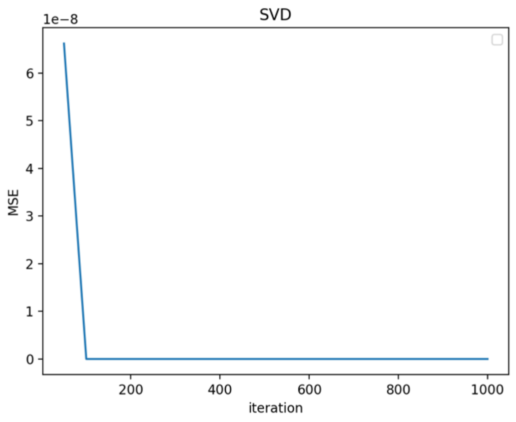

Figure 2

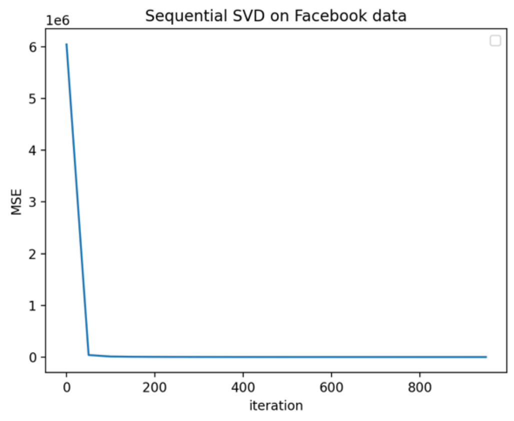

Figure 3

The optimized SVD approach shows a similar pattern of converge as SVD does. But with lower feature number, the optimized SVD is performing worse than SVD. However, with the feature number higher than around 15, the optimized SVD will perform better than SVD in terms of accuracy. (Figure 4)

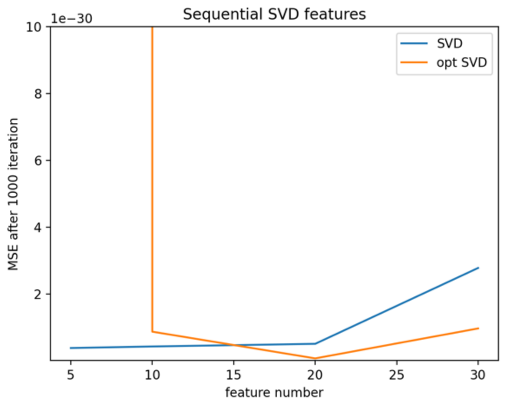

Figure 4

In order to get the relationships between feature numbers and accuracy, I tested multiple feature number to collect MSE in SVD. The reason I use SVD to perform this test is the SVD has a more stable accuracy curve relating to feature number showed in figure 4. According to figure 5, the accuracy decreases as the feature number goes on with the iteration limited to 1000.

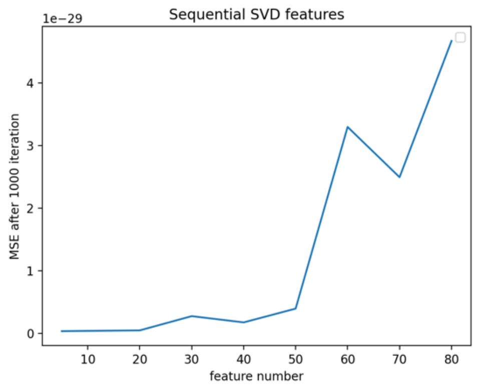

Figure 5

I test the algorithm with a 4039 by 4039 sparse matrix "facebook_combied.mtx" and collect MSE. I did one full 1000 iteration run for SVD algorithm, and it took around 70 hours to complete. For the optimized SVD, I only ran 100 iterations to collect data for comparison due to the large run time. Table 1 shows with large data size, feature size 20, the optimized SVD has advantage in terms of accuracy.

Algorithm\ Iteration	50	100
SVD	6047031.40496596	38965.018873682326
Optimized SVD	5145994.648103562	21142.895750632104

Table 1

5. Conclusions and Futures Work

Matrix factorization's accuracy is influence by multiple aspects. For feature number lower than 15, SVD performs better than the optimized SVD in this project. The optimized SVD reaches its best performance at feature number equal to 20. However, the time efficiency of all three algorithm in this project is not good enough, as the sequential algorithm is slow on computing large matrix. To improve this, we can parallel those algorithms with GPU, the parallelized approaches will allow the algorithms to calculating different values in matrix at the same time. With the large number of GPU cores inside a GPU, runtime should be significantly improved compared to sequential approaches implemented on CPU [3].

6. Acknowledgements

This project is complete with the help of articles on "towards data science" [4] [5] and Wikipedia. The large testing data was provided by professor A Sukumaran Rajam in previous semester.

References:

[1] X. Hu, X. Meng, and L. Wang, "SVD-based group recommendation approaches," Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation - CAMRa '11, 2011.

[2] "Mean squared error," Wikipedia, 30-Nov-2021. [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error. [Accessed: 16-Dec-2021].

[3] L. He, T. Miskell, R. Liu, H. Yu, H. Xu, and Y. Luo, "Scalable 2d k-svd parallel algorithm for dictionary learning on gpus," Proceedings of the ACM International Conference on Computing Frontiers, 2016.

[4] C. Maklin, "Singular value decomposition example in Python," Medium, 05-Aug-2019. [Online]. Available: https://towardsdatascience.com/singular-value-decomposition-example-in-python-dab2507d85a0. [Accessed: 16-Dec-2021].

[5] J. Moore, "Recommender Systems in python from scratch!," Medium, 20-Jul-2020. [Online]. Available: https://towardsdatascience.com/recommender-systems-in-python-from-scratch-643c8fc4f704. [Accessed: 15-Dec-2021].