Recommendation system using Matrix Factorization(MF) and Pyspark

Jaehyun Yoon
University of Maryland,
College Park
jyoon125@umd.edu

Abstract — **Among the recommendation algorithms, one of the most widely used algorithms is by far Collaborative Filtering (CF). Among them, the one with the highest performance as a single algorithm is the Matrix Factorization (MF) algorithm, which is a model-based CF. The MF recommendation model can be viewed as a process of filling in the missing data in the sparse user-item matrix through the model. In this paper, we study how we can split the rating matrix into user and item matrices and explain how it can train the objective function to minimize error in the Latent Factor matrix of users and items.**

Keywords—Collaborative Filtering (CF), Content-Based Filtering (CBF), Matrix Factorization (MF), Latent factor, Stochastic Gradient Descent (SGD), Sparse Matrix, Coordinate (COO), Compressed Sparse Row (CSR), Pyspark, Spark MLlib, Root-Mean Square Error (RMSE), Recommendation algorithm, Optimization,

I. INTRODUCTION

The recommendation system for movies, TV shows, etc. helps the show selection by predicting the audience's preference based on the audience's past behavior or their reviews. According to an interview with The New York Times, Neal Mohan, YouTube's Chief Product Officer, said, '70% of YouTube users' viewing time is the result of the recommendation algorithm, and the introduction of the algorithm increases the total video viewing time by more than 20 times than before.[1] Netflix collects the data based on viewers' preferences and uses the same to help you find content which you will be interested in. Netflix held a contest that award a prize of $1 million to the team that was able to improve their recommendation system back in 2006[1]. Disney plus, Apple Tv, Hulu, etc., all of the over-the-top (OTT) media services put much effort into providing a better recommendation system to the customers. Of course, dealing with human tastes and providing the right preferences is a challenging problem. There were many different algorithms researched and developed for the recommendation system, and the OTT companies like YouTube, Netflix, etc. have continued to seek the state-of-art algorithm.

This paper explores to find out how does the recommended system work and what is the logic behind the system. In the real-world problem, hundreds of million data would be used to build the model and recommendation system, but one million datasets is used for this paper because of the limit of handling big data. The dataset is a movie dataset from MovieLens at Grouplens.org, which contains the id of users and movie, and. The dataset contains the 20 variables which are the information of the 1 million ratings from 6000 users on 4000 movies, released in 2/2003.

Recommendation systems are largely divided into Collaborative Filtering (CF) and Content-Based Filtering (CBF). The CBF configures each profile database for users and items. In the case of an item, for example, if it is a movie, it can contain the actors, genre, year of production, etc. The CF identifies the relationship between existing users and items and makes recommendations between new users and items. Unlike the CBF, explicit data is used, and experience data such as the rating of movies are mainly used. In this case, the performance of the CBF model is better, but the CF model works better for most cases.[3]

This paper seeks to understand the CF movie recommendation system using the Matrix Factorization (MF) which is the latent factor model, the average of ratings of movies and correction of bias. The Latent Factor model learns the relationship between the user and the item as each of the Latent Vector. The interaction between the item and the user is calculated as the value of the inner product between these two vectors. That is, it expresses the user's interest in the item. Since the data on item-user interaction is sparse, we use the regularization method to learn to reduce the error of the model. For that method, the Stochastic Gradient Descent (SGD) or the Alternating Least Squares (ALS) are used. [3]

This paper seeks to understand the CF movie recommendation system using the Matrix Factorization (MF) which is the latent factor model, the average of ratings of movies, and correction of bias. The Latent Factor model learns the relationship between the user and the item as each of the Latent Vector. The interaction between the item and the user is calculated as the value of the inner product between these two vectors. That is, it expresses the user's interest in the item. Since the data on item-user interaction is sparse, we use the regularization method to learn to reduce the error of the model. [3] For the different methods of gradient descent and optimization, the Stochastic Gradient Descent (SGD) or the Alternating Least Squares (ALS) are used, but the study is more focused on the SGD than the ALS methods. This paper also suggests how can the CF recommendation system run in the big data environment using the Pyspark. To deal with large datasets, the three methods of Pyspark are used. The first method is that it reads the dataset with Pyspark DataFrame and creates Scipy's sparse matrix to feed the matrix to the MF model. Secondly, Pyspark's MLlib is used to read data and build a model using MLlib's ALS package. Finally, Keras is used to build the MF recommendation model. Then, the performance of the recommendation system is evaluated using the Root-Mean-Square-Error (RMSE) to see which methods perform better than other.

II. THEORY

A recommender system is a system that recommends items for a user. There are two filtering algorithms in the CF which are the Memory-based filtering and Model-based filtering. Memory-based filtering focuses on individual user data by calculating and recommending the data in memory. Model-based filtering focuses on the entire user pattern by constructing a model in advance from data and recommending it when necessary.[4] Matrix Factorization (MF) is a representative algorithm belonging to model-based filtering among collaborative filtering (CF). MF is used in many recommendation systems because of the good performance.

A. Background

1. Matrix Factorization (MF)

$$R \xRightarrow[\substack{Decomposed \\ by}]{} P * Q^T = \hat{R}$$

The above equation is the base logic of MF. The decomposed matrix of R, the rating matrix to two metrics such as P and the transpose of Q metrics. P is the m * k dimension, which is the user-latent factor matrix with the value of the relationship between the user and the latent factor. In other words, P is a matrix composed of k factor values representing the characteristics of each user. Q is an item-latent factor matrix of $n * k$ dimension with values of the relationship between the item and the latent factor. In other words, Q is a matrix composed of k factor values representing the characteristics of each item. The goal of the MF recommend system is to find matrices P and Q that minimize the difference between $\hat{R}$ and R. From the restored matrix $\hat{R}$, it is possible to obtain a latent factor not observed in the original matrix R. Using the gradient descent method, P and Q are inferred through cost function optimization. The predicted $\hat{R}$ can have the least error with the actual R-value.

2. Stochastic Gradient Descent (SGD)

Latent Factor Matrix, where P and Q have the k latent values, initially has random values. Now, the decomposed matrices are treated as parameters, and the matrix values are updated through model training. Finally, the dot product of the two decomposed matrices is made to the original matrix shape $\hat{R}$, and the actual R value and error calculation and P and Q are corrected. By repeating this, it is possible to fill in the sparse part of the data, and it is possible to predict the optimal user preference for an item that has not yet been experienced. P and Q are used as training features, and the model is trained based on the objective function below.[5]

$$\min \sum \left(r_{(u,i)} - p_u * q_i^t\right)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

In the process of training two features, the first part of the objective function is updated by the new p and q with the error, which is the formulas down below. The second part of the objective function is to avoid overfitting which is the regularization term using $L2$.

$$\underset{Error}{e_{ui}} = r_{ui} - q_i{}^T p_u$$

$$p'_u = p\_u + \eta(\, e_{(u,i)} * q_i - \lambda * p_u)$$

$$\acute{q}_i = p_u + \eta\big(\, e_{(u,i)} * q_i - \lambda * p_u\big)$$

p_u : the user u row vector of matrix P
q_{it} : the item i column vector of matrix Q
$e_{(u,i)}$:$r_{(u,i)} - \hat{r}_{(u,i)}$: Difference error between the actual and predicted matrix values located in row u and column I
λ : lambda value

The r_{ui} is predicted, and the prediction error is calculated accordingly. It is to update P and Q towards to the negative direction using the gradient and the error of the objective function. In addition to the regularization term, the tendency of users, who tend to give high ratings or low ratings to certain movies, can be considered. Those tendencies, the formulas down below, can be updated for each iteration to have accurate outputs. [5][6]

$$\hat{r}_{ij} = b + bu_i + bd_j + \sum_{k=1}^{k} p_{ik} q_{kj}$$

$$b\grave{u}_i = bu_i + \alpha\,(e_{ij} - \beta bu_i)$$

$$b\grave{d}_j = bd_j + \alpha\,(e_{ij} - \beta bd_j)$$

b : overall average
bu_i : difference between user i and overall average
bd_j : difference between item j and overall average

3. Alternating Least Squares (ALS)

In this SGD method, since two matrices P and Q need to be learned, the optimization calculation is a complex non-convex problem, and it has the disadvantage that it takes a long time to learn. Since it is difficult to optimize both P and Q matrices at the same time in the SGD method, the method that appeared is ALS. In simple terms, ALS is a method of optimizing P and Q one by one alternately. When optimizing P, Q is treated as a constant, and when optimizing Q, P is treated as a constant. [7] Pyspark MLlib has the ALS as a recommendation system package.

III. IMPLEMENTATION

A. Sparse matrix – SGD

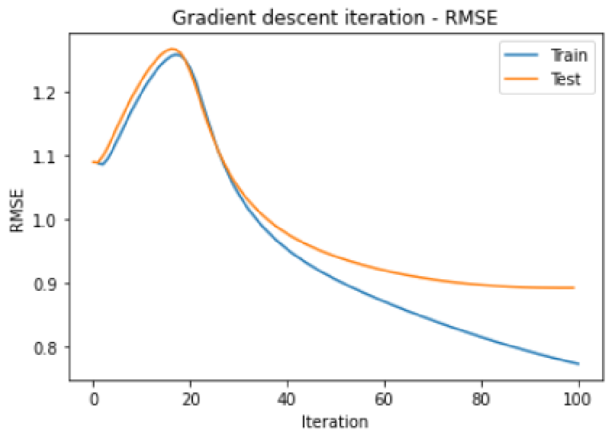
When most of the data used in the recommendation system are converted into full matrices, many elements are empty metrics or zero values. That is, it becomes a sparse matrix in which some data does not exist. To solve this problem, it is much more efficient to store and use a metric that has the same value as the actual index. However, the sparse matrix increases the overhead cost because it has to check whether all indices and their values exist. There are two methods of the sparse matrix: Coordinate (COO) and Compressed Sparse Row (CSR) format. In the COO format, only non-zero data is stored in a separate array and the columns and rows pointed to by the data are stored in separate data. [9] In other words, we use a matrix of values and a matrix of positions representing those values. The CSR format engages the rows by rearranging them in a horizontal order and compacting them. [9] Wikipedia explains it like "...a matrix M by three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices."[9]

a. Pyspark & Scipy – sparse matrix

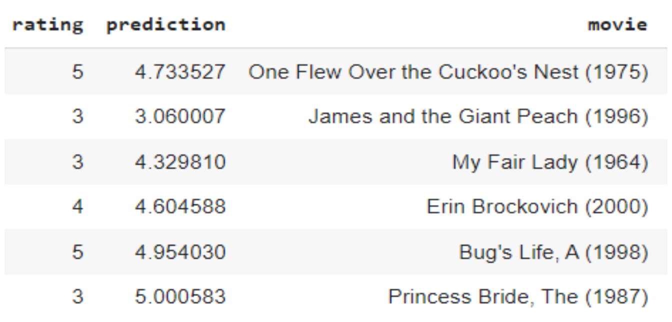
To read the dataset on Pyspark, the SparkSession is created, and read dataset as a CSV format and convert to a data frame. Each column is converted into the 1-D array, and each array is converted into a sparse matrix using the SciPy library. To be more specific, each array is converted into a COO matrix which requires the shape and the size of the matrix, and then it is converted into a CSR matrix. For this method, any other recommend system libraries are not used to implement the MF. Most of the hyperparameters are randomly selected and get references from the common values that use for other optimization models. Sci-Ki learn package is used to split the data into the train and the test dataset

b. Result

The plot down below shows how the RMSE is changed for every iteration. The RMSE goes up for the first two iterations and drop down to 0.8920 as the best result after the 100 iterations



The table down below is the partial result of what we predict for user id one. The full table is in the coding notebook. The prediction shows how well the model predicts each rating.



rating	prediction	movie
5	4.733527	One Flew Over the Cuckoo's Nest (1975)
3	3.060007	James and the Giant Peach (1996)
3	4.329810	My Fair Lady (1964)
4	4.604588	Erin Brockovich (2000)
5	4.954030	Bug's Life, A (1998)
3	5.000583	Princess Bride, The (1987)

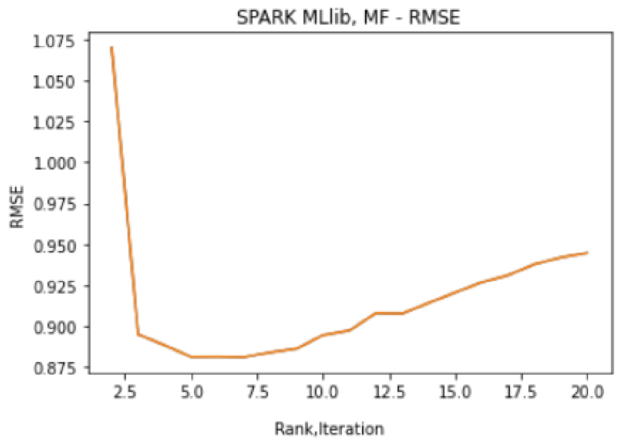
B. Pyspakr MLlib – ALS

a. Pyspark MLlib

The second method is implemented using Pyspark MLlib. Pyspark provides the recommendation system packages in MLlib. The Pyspark documentation, it recommends using 20 iterations or less to get a reasonable solution. We run 1 iteration to 20 iterations to find the best result. From splitting the dataset into the train and the test dataset to evaluating the result, it was able to implement using Pyspark.

b. Result

The RMSE value gets dropped after the third iteration, but it goes up after the eighth iterations to 0.9447. The best result is 0.8810 RMSE in the seventh iterations, which is a little bit better than the previous SGD method.



C. Keras – SGD

a. Pyspark – Keras

For the last method, we use and implement the MF using Keras package. We wouldn't say this is a deep learning architecture since we didn't add any hidden layers in this architecture. The input layer is used the user one-hot representation and the item one-hot representation which is the same as one-hot encoding using dummy variables, so the input layer has the M(user)*N(item) matrix. For the user input layer, the embedding layer has the K number of the latent value which is the K number of nodes. Therefore, there are the M * K numbers of connections between the input layer and the embedding layer. The previous MF methods, P has M * k dimension which is the same number of connections of this method, and the same works to Q matrix. The last layer, Element-wise Product Layer, has the $P * Q^T$ dimension of matrix. [11] Fig 1 can help your understanding of MF in a shallow neural network.

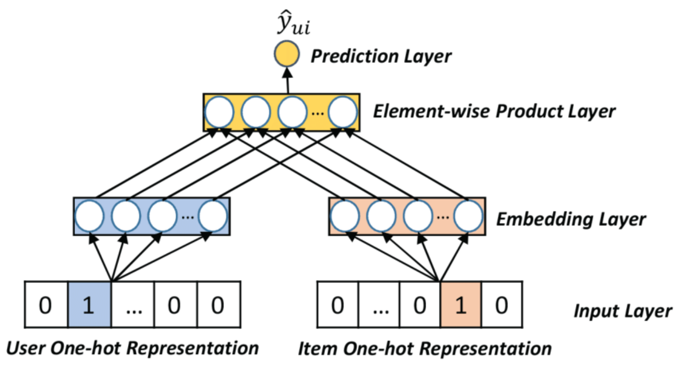
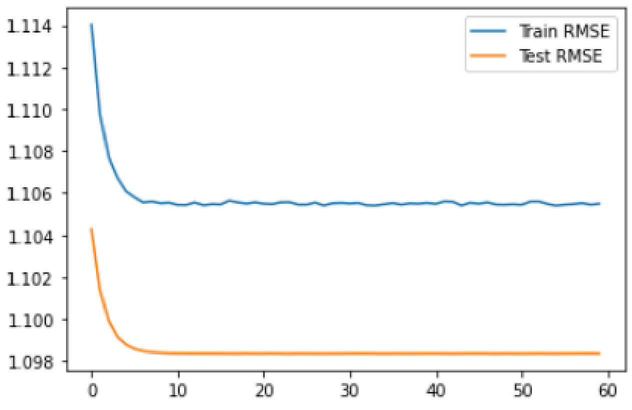

$\hat{y}_{ui}$
Prediction Layer
Element-wise Product Layer
Embedding Layer
Input Layer
0 1 ... 0 0
0 ... 0 1 0
User One-hot Representation
Item One-hot Representation

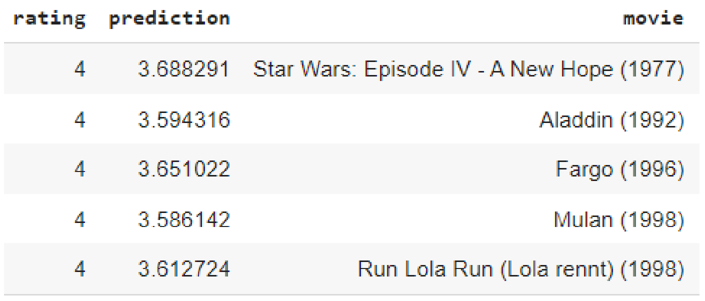
Fig. 1 -Matrix factorization (MF) as a shallow neural network model [11]

b. Result

This Keras neural network runs for 60 epochs, but the RMSE doesn't drop after the 8 epochs which results in the 1.0984 RMSE value. It doesn't have a better result than the two previous methods.



As you can see the table below, it doesn't predict well than the output of the first method if you compare the rating and the prediction.



rating	prediction	movie
4	3.688291	Star Wars: Episode IV - A New Hope (1977)
4	3.594316	Aladdin (1992)
4	3.651022	Fargo (1996)
4	3.586142	Mulan (1998)
4	3.612724	Run Lola Run (Lola rennt) (1998)

IV. CONCLUSION

This paper explores one recommendation system, MF, and implements it in a big data environment. As mentioned in the introduction, hundreds of millions or billions of data and more complex algorithms than the simple MF would be used to build a real recommendation system. Also, there would be a different issue like the cold start problem. For users who use the system or the platform for their first, there won't be any information about their preference items. Comparing the first method and the second method, it can tell how useful and efficient to use and integrate the system within Pyspark can help us to minimize the amount of coding or the better result. It doesn't say that Pyspark MLlib is always better than other machine learning packages, but it handles well with a large amount of dataset and get possibly better outputs since we train with big data.

REFERENCES

[1] *YouTube's product chief on online ... - The New York Times*. (n.d.). Retrieved December 10, 2021, from https://www.nytimes.com/2019/03/29/technology/youtube-online-extremism.html.

[2] Vena, D. (2021, November 18). *You won't believe what Amazon and Netflix just partnered on*. The Motley Fool. Retrieved December 10, 2021, from https://www.fool.com/investing/2021/11/18/you-wont-believe-what-amazon-and-netflix-just-part/.

[3] *Recommender Systems [netflix] - datajobs.com*. (n.d.). Retrieved December 10, 2021, from https://datajobs.com/data-science-repo/Recommender-Systems-%5bNetflix%5d.pdf.

[4] *A comparative analysis of memory-based and model-based ...* (n.d.). Retrieved December 10, 2021, from https://qoribmunajat.github.io/files/comparative-analysis-memory-based-model-based-recommendation-systems.pdf.

[5] Google. (n.d.). *Matrix factorization | recommendation systems | google developers*. Google. Retrieved December 10, 2021, from https://developers.google.com/machine-learning/recommendation/collaborative/matrix.

[6] Luo, S. (2019, February 06). Intro to Recommender SYSTEM: Collaborative filtering. Retrieved February 19, 2021, from https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

[7] *Matrix Completion via Alternating Least Square(als)*. (n.d.). Retrieved December 10, 2021, from http://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf.

[8] *Scipy.sparse.coo_matrix*. scipy.sparse.coo_matrix - SciPy v1.7.1 Manual. (n.d.). Retrieved December 10, 2021, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.html.

[9] Brownlee, J. (2019, August 9). *A gentle introduction to sparse matrices for machine learning*. Machine Learning Mastery. Retrieved December 10, 2021, from https://machinelearningmastery.com/sparse-matrices-for-machine-learning/.

[10] Wikimedia Foundation. (2021, December 9). *Sparse matrix*. Wikipedia. Retrieved December 10, 2021, from https://en.wikipedia.org/wiki/Sparse_matrix.

[11] Wang, X., He, X., Nie, L., & Chua, T.-S. (2017). Item silk road. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. https://doi.org/10.1145/3077136.3080771