APPLICATION OF MATRIX FACTORIZATION FOR RECOMMENDER SYSTEMS WITH PARALLEL COMPUTING

Süleyman Onur Doğan

Antalya Bilim University, Department of Computer Engineering
onur.dogan@std.antalya.edu.tr

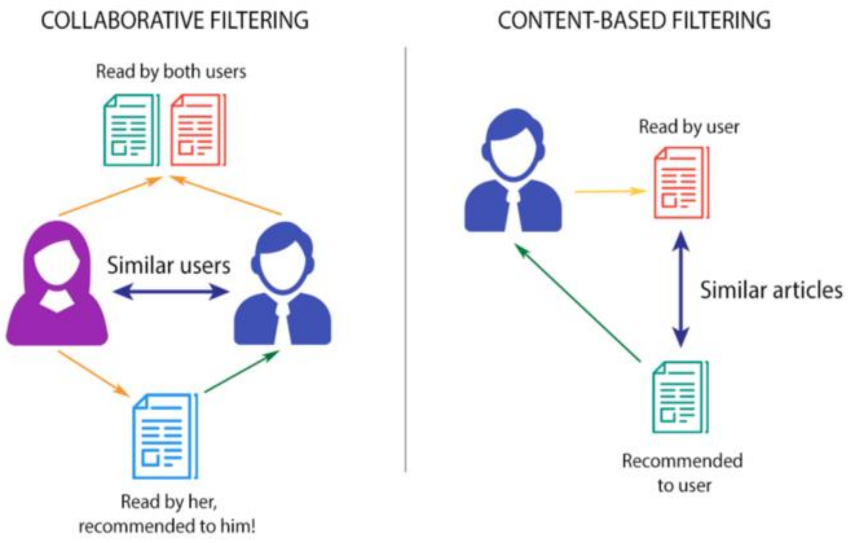
Abstract

Recommender systems (RS) have become a hot topic in study, with the goal of assisting consumers in finding best choice online by delivering choices that closely match their interests. We can see the applications of recommendation systems in many places on the internet such as Youtube, Amazon, Netflix and so on. Many mathematical methods have been proposed and developed for recommender systems such as KNN. In this article, I will discuss the application of Matrix Factorization technique with ALS optimization algorithm on the Collaborative Filtering-based recommendation system. I will cover the model [1] which is developed, used in the Netflix competition [2] and won Netflix Prize in this paper. I will be focusing application of the model to the movieLens [3] data, which consist of ratings of movies by different users with Spark [4], which is Large-scale data analytics engine with a unified engine.

Keywords: Matrix Factorization, Recommender Systems, Parallel Computing Algorithms, Linear Algebra, Application Of Linear Algebra
General area of research: Application of Linear Algebra

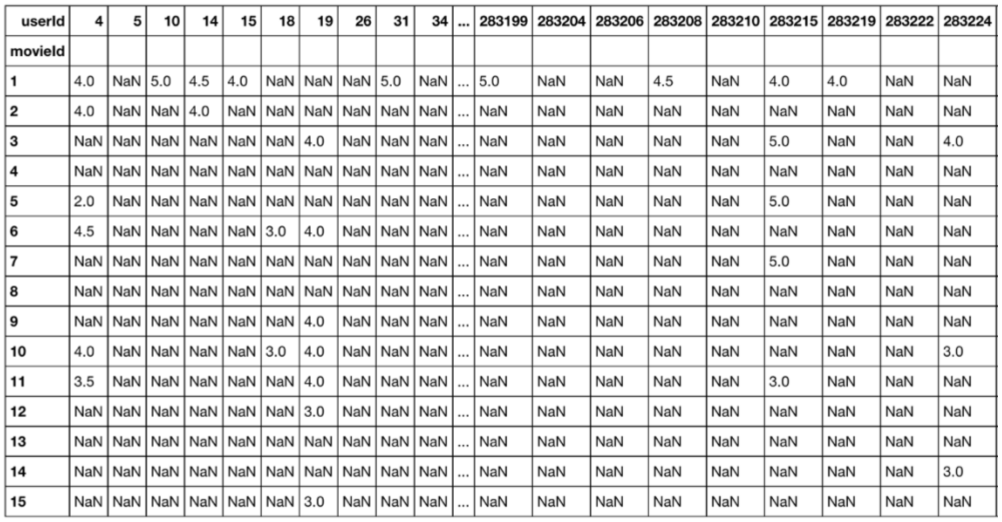
1 Introduction

Today 's consumers are spoiled for options. Electronic shops and content providers have never before offered such a diverse range of products, allowing them to fulfill a wide range of particular demands and tastes. Enhancing user pleasure and loyalty requires matching customers with the most appropriate offerings. As a result, more shops are interested in recommender systems, which examine patterns of user interest in products in order to deliver customized recommendations tailored to the customer's preferences. E-commerce heavyweights like Amazon.com and Netflix have made recommender systems a prominent element of their websites because effective personalized recommendations can add another layer to the customer experience. Entertainment products, such as movies, music, and television shows, benefit greatly from such systems. Many customers will see the same film, and each client is likely to see a variety of films. Customers have shown a willingness to rate their satisfaction with certain films, resulting in a massive amount of information about which films appeal to which customers. Companies can use this information to make movie recommendations to specific clients. Furthermore, based on the state-of-the-art in recommender systems, the Recommender Systems (RS) can be classified into three techniques: Content-based (CB), Collaborative Filtering (CF), and Hybrid approaches [5]. Basically, a content-based approach uses a set of specific qualities of an item to suggest other objects with comparable features, the collaborative filtering strategy creates a model based on a user's previous actions (things previously purchased or selected, and/or numerical ratings assigned to those items), as well as comparable selections made by other users. This model is then used to estimate which things (or item ratings) the user would be interested in. The hybrid strategy combines the two preceding methods. In their production recommender systems, most organizations most likely adopt a mixed method.



(Figure 1)[6]

CF is one of the most successful approaches for providing recommendation service. The Collaborative Filtering system's content is straightforward, but it suffers from a serious flaw: missing values. In the actual world, the great majority of movies earn only a few or no user ratings at all. We're dealing with a matrix that's incredibly sparse, with more than 99 percent of the entries being missing values. Although it has been well known since the Netflix Prize Challenge, matrix factorization is the state-of-the-art solution for sparse data problems in collaborative filtering.



userId	4	5	10	14	15	18	19	26	31	34	...	283199	283204	283206	283208	283210	283215	283219	283222	283224
movieId																				
1	4.0	NaN	5.0	4.5	4.0	NaN	NaN	NaN	5.0	NaN	...	5.0	NaN	NaN	4.5	NaN	4.0	4.0	NaN	NaN
2	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	4.0
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN
6	4.5	NaN	NaN	NaN	NaN	3.0	4.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	4.0	NaN	NaN	NaN	NaN	3.0	4.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.0
11	3.5	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN
12	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.0
15	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

(Figure 2)[6]

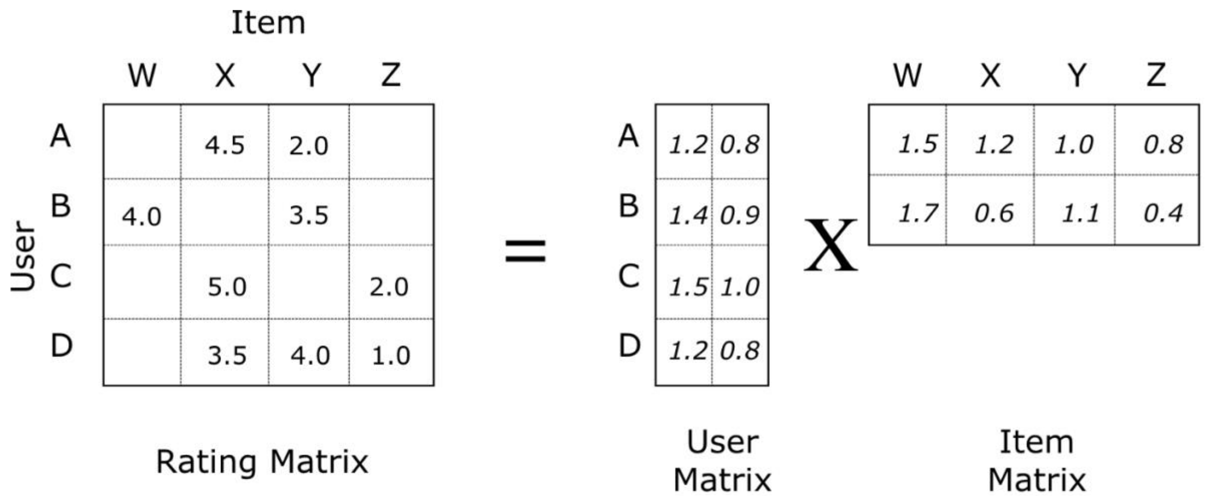
In the [1] article, which is the winner of the Netflix prize challenge, the matrix factorization method was presented with 2 different learning algorithms which are stochastic gradient descent and alternating least squares (ALS) to overcome this significant problem [1]. In the second chapter, the ALS method, which is more effective algorithm, and the matrix factorization are discussed in more detail.

In the other hand, the Importance of the computational side of algorithms has been increased widely with big data. Big companies like Amazon, Google, Netflix have massive amounts of data in their data centers. When we consider Netflix and the fact that it uses collaborative filtering for recommendation, it requires the ratings of other users in order to make a movie recommendation to another user. Because Netflix has a large number of users, they store these ratings in their data centers and use them in the collaborative filtering algorithm, which processes the data and makes recommendations to other users based on these ratings. When we think this system, we can see that this kind of recommendation requires very fast computing process. For that purpose, companies like Netflix uses parallel processing techniques. In this article, I will be using Spark which is described in chapter two.

2 Methods and Applications

2.1 Matrix Factorization Method for Recommendation Systems

As I mentioned in first chapter, Matrix Factorization is very critical for collaborative filtering-based recommendation systems so, what is matrix factorization? Matrix factorization is a collection of matrices-related mathematical procedures in linear algebra. A matrix factorization, to be precise, is the transformation of a matrix into a product of matrices. Matrix factorization techniques function in collaborative filtering by decomposing the user-item interaction matrix into the product of two smaller dimensionality rectangular matrices. One matrix is the user matrix, in which the rows represent users and the columns represent latent factors. The item matrix, on the other hand, has rows for latent factors and columns for items.



(Figure 3)[6]

As a result, when we consider how does matrix factorization? First of all, Model learns to factorize rating matrices into user and movie representations, allowing it to better predict user-specific movie ratings [6]. And secondly, Matrix factorization allows lesser-known films to have as rich latent representations as well-known films, which increases the recommender's capacity to recommend lesser-known films [6]. Users and objects are mapped to a joint latent factor space of dimensionality f in matrix factorization models, and user-item interactions are described as inner products in that space. As a result, each object i is associated to a vector $q_i \in R^f$ whereas each user u is associated to a vector $p_u \in R^f$.

The aspects of $q_i$quantify the extent to which an item possesses certain factors, positive or negative, for a given item i. The elements of p_u, for a particular user u, measure the user's level of interest in items that score high on the associated variables, which can be also positive or negative. The resulting dot product, $q_i^T p_u$, represents the interaction between user u and item i the user's overall interest in the item's features.

This calculates the estimate based on user u's rating of item i which is denoted by $\overline{r_{ui}}$.

$$\overline{r_{ui}} = q_i^T p_u \quad (1)[1]$$

Computing the mapping of each item and user to factor vectors $p_u, q_i \in R^f$is the most difficult task. By applying Equation 1 after the recommender system has completed this mapping, it can easily estimate the rating a user will give to any item. Different methods such as singular value decomposition tried to map of each item and user to work with Equation 1. As a result, more recent research [7],[8] has recommended modeling only the observed ratings while avoiding overfitting using a regularized model. To learn the factor vectors q_i, p_u, system need to minimize fallowing equation:

$$(2)[1] \qquad min_{q,p} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \alpha(\|q_i\|^2 + \|p_u\|^2)$$

In the equation, r_{ui} is known from the training data, K is the set of the (u,i) pairs. Basically, the system learns the model by fitting the previously observed ratings. However, the idea is to generalize those previous ratings in a way that predicts future, unknown ratings.

System needs to avoid overfitting the observed data because, if system does overfitting based on training data, system will not do correct prediction on other data. So, for that purpose, we have α constant in Equation 2 to penalize system. This concept is very important for machine learning algorithm as well as this algorithm.

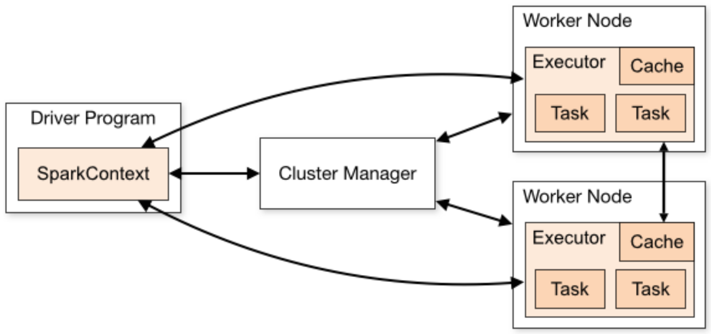
To minimize the Equation 2, stochastic gradient descent and alternating least squares (ALS) approaches can be used [1]. Within this paper, ALS approach will be covered and used.

Basically, idea of ALS comes from fixing one of the variables then fixing other variable so, In our Equation 2 both q_i and p_u are unknows. ALS techniques rotate between fixing the q_i's and fixing the p_u 's. When all p_u 's are fixed, system recomputes the q_i's by solving a least-squares problem and reverse. This ensures that each step decreases Equation 2 until convergence [9]. When we consider ALS optimization algorithm, we can clearly see that this algorithm requires computational power, and it is hard. However, when we apply this algorithm, we have a change to changes to paralyze the system [9]. Each qi is calculated independently of the other item factors, and each p is calculated independently of the other user factors. As a result, the algorithm might possibly be massively parallelized [9]. So, we can use this algorithm to minimize Equation 2 for getting movie recommendation using Matrix Factorization. Also, we can implement this algorithm by coding in parallel computing tool which described in chapter 2.2 and coded in chapter 3.3

2.2 Parallel Computing Method

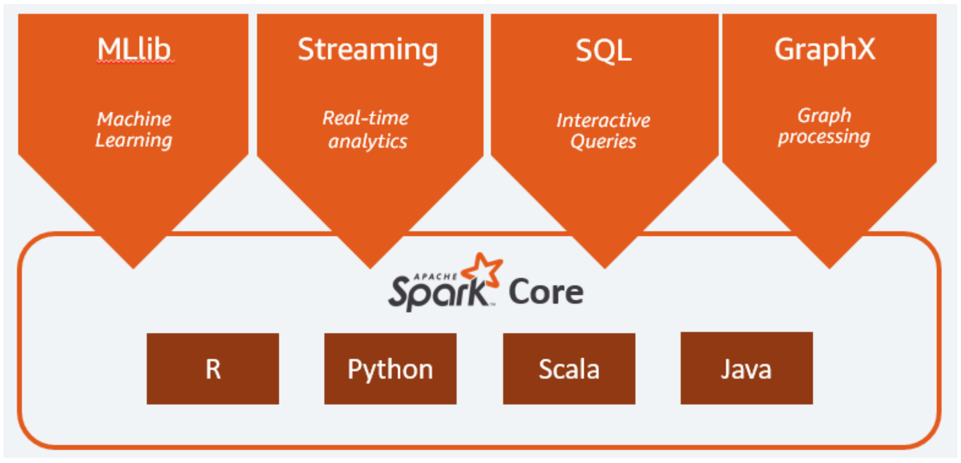
With web 2.0, applications entered our phones and tablets. As a result, humanity started to produce a huge amount of data and the concept of big data emerged. Actually, Big data is just not related to web 2.0, order to development of new experimental and measurement equipment data becomes bigger on science. For years, scientists and engineers have acted with the importance of data because data equals to information. Huge data and machine learning frameworks have risen in popularity as a result of this. In addition to these, when we have big data, it is equals to big information about the application. Not all of the big data is used efficiently, so we need to process big data and do some analysis on it by programming. However, processing and analyzing big data poses a huge computational challenge.

There have been major innovations in the field of open-source developments that can be used to deal with massive data. One of the best innovation is Apache Hadoop which developed by map-reduce and Hadoop Distributed File System (HDFS) ideas. The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. Basically, assume that we are using bunch of computer systems which is called node then, Hadoop divides files into big chunks and distributes them among cluster nodes with HDFS idea and at the same time divides the data processing work, distributes it to node cluster, then processes the data in distributed system and combines this distribution to get a single result with map-reduce idea [10]. Hadoop has worked well for years and still works well. In 2014, Apache Spark is developed at the University of California, Berkeley's AMPLab then, it gets popular in the area. Apache Spark is an open-source unified analytics engine for large-scale data processing, and it is depending on Hadoop Map Reduce algorithm [11]. One of the biggest differences of Spark from Hadoop is that it does not distribute the data file to other nodes, Spark keeps it in a single system and processes that data in parallel. These and other ideas makes spark 10 times faster than Hadoop.



(Figure 4)

Another important and rewarding side of the Apache Spark is core of the Apache Spark. Spark can be executed in different programming languages such as Python, Scala which showed in Figure 5. Also, Spark has very beneficial framework such as MLlib which we will use. These frameworks created by Spark and we can use these frameworks by processing data.



(Figure 5)

To implement Recommendation System, I will be using PySpark which is Spark in Python with Machine Learning Library of Spark which showed in chapter 2.3.

2.3 Application

Idea of Recommendation Systems explained in chapter 1, mathematical approach for recommendation explained in chapter 2.1 and computational side of the recommendation systems discussed in chapter 2.2. So, We can combine these approaches in application which is movie recommendation to users depends on ratings. For the application, I will be using one of most popular dataset in the are that is MovieLens. GroupLens Research has collected and made available rating data sets from the MovieLens web site. The data sets were collected over various periods of time, depending on the size of the set. Original dataset contains over 25 million movie ratings. I will use smaller version of the original dataset in the application. Mainly, This data have 2 file that are movies and ratings. Movies file consist of MovieId, title, genres and ratings file consist of userId, movieId, rating, timestap.

In the coding part of the application, I used Pyspark considering the big data. As I described in chapter 2.2, there is useful machine learning library of Spark which called Mllib so, I used that library to implement Matrix Factorization with ALS optimization algorithm. Matrix Factorization method with ALS optimization algorithm works like a machine learning algorithm, we will be training our algorithm and doing prediction with the algorithm. As I mentioned, It works like machine learning algorithm since it tries to minimize Equation 1. In Machine Learning algorithms, People divides their data into training and testing part or training, testing, validation so, in our application I will divide the MovieLens data into training and testing. It means that, I will use the %70 of the data to train my model than, I will %30 of data to test my model with real and predicted data. PySpark code showed and explained below:

```
[1]from pyspark.ml.recommendation import
ALS
[2]from pyspark.sql import SparkSession
[3]from pyspark.ml.evaluation import RegressionEvaluator

[4]spark = SparkSession \
.builder \
.appName("Recommendation System") \
.config("spark.some.config.option", "some-value") \
.getOrCreate()
[5]ratings=spark.read.csv('ratings.csv',inferSchema=True,header=True)
[6]movies = spark.read.csv('movies.csv', inferSchema=True, header=True)
[7]ratings=ratings.join(movies,"movieId")

[8]data=ratings.select("userId","movieId","title","rating")
[9]data.show(2)
output:
+------+-------+--------------------+------+
|userId|movieId|               title|rating|
+------+-------+--------------------+------+
|     1|     31|Dangerous Minds (...|   2.5|
|     1|   1029|        Dumbo (1941)|   3.0|
+------+-------+--------------------+------+
[10](train, test) = data.randomSplit([0.7, 0.3])

[11]als=ALS(maxIter=10,regParam=0.01,userCol="userId",itemCol="movieId",ratingCol="rating",coldStartStrategy="drop")
[12]model = als.fit(train)
[13]prediction =model.transform(test)
[14]prediction.filter(prediction.userId==46).show(2)
output:
+------+-------+--------------------+------+----------+
|userId|movieId|               title|rating|prediction|
+------+-------+--------------------+------+----------+
|    46|  58559|Dark Knight, The ...|   5.0| 5.6347127|
|    46|  81834|Harry Potter and ...|   5.0| 5.1254735|
+------+-------+--------------------+------+----------+

[15]evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
                                          predictionCol="prediction")
[16]rmse = evaluator.evaluate(prediction)
[17]print("Root-mean-square error = " + str(rmse))
output:
Root-mean-square error = 1.2116538308750624

[18]users = ratings.select("userId").filter(ratings.userId==46)
[19]userSubsetRecs = model.recommendForUserSubset(users, 1)
[20]userSubsetRecs.show(1)
output:
+------+-----------------+
|userId|  recommendations|
+------+-----------------+
|    46|[[44761, 9.018951]]|
+------+-----------------+
[21]data.select("title","movieId").filter(data.movieId==44761).show(1)
Output:
+------------------+-------+
|        title|movieId|
+------------------+-------+
|Brick (2005) |   44761|
+------------------+-------+
```

In the code block, we also can see the libraries which are imported and outputs of the code. Explanation of the code:

- In the first three lines, Libraries are imported.
- In the fourth line, Spark application defined to make our work parallel. However, I used one single computer, so this process didn't complete parallel.
- In the fifth and sixth line, MovieLens data defined.
- In the seventh line, Ratings data and movies data merged by their movieId columns and got one data.
- In the eighth line, Columns which will be use selected and in the nineth line I displayed the last version of the data.
- In the tenth line, Data splinted to training and testing that I described above.
- In line 11, we created our recommendation model using the ALS function of the Pyspark library. This function stands for Matrix Factorization with ALS which described in chapter 2.2. In the model, hyper parameters defined as a maximum iteration is 10 and regularization parameter (α) is 0.01. User column, movie column and ratings column defined for the model.
- In line 12, Model is trained by using training data.
- In line 13, Model predict movie ratings of user by using test data, basically, we tested our model and got predict movie ratings for users in test data.
- In line 14, Displayed 2 movie rating for user with userId 46. We can see that, rating column represent real rating for the movie and prediction column represent predicted rating for the same movie. As a result, Values are closer and model works well.
- In lines 15,16 and 17, Error of the model evaluated with root mean square error(RMSE) and it is 1.2116538308750624
- Our model is ready for the movie recommendation since we trained the model and In lines 18,19 and 20, We recommend a movie for user with userId 46. Id of the Recommended movie is 44761 and predicted rating for this movie for the user with userId 46 is 9.018. In line 20, We checked the name of the movie with movieId 44761. As a result, Brick movie recommended for specific user.

3 Discussion and Conclusions

Linear algebra is one of the important topic of mathematics. There are so many applications of Linear Algebra in different areas as well as Computer Engineering. We can see that the value of some products is being decided by using the Simplex method which uses matrix operations. It is one simple example of the application of Linear Algebra to economics. When we look at Computer Engineering, there are so many applications of linear algebra. For example, one of the important topics in machine learning is Principal Component Analysis (PCA). It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. While calculating PCA, operations are performed with eigenvector and eigenvalues. This is another application of Linear Algebra on Computer Engineering.It has many application areas as discussed and implemented in this article on Recommendation System. By developing the application areas of linear algebra, very different and useful applications can be obtained.

In conclusion, I have used Matrix Factorization that is one of the topics of Linear Algebra and implemented this topic on Recommendation system area with parallel computing. This kind of model is used by Netflix, Amazon and so on and one of the models in this article created by implementing the winner article [1]of the Netflix prize in PySpark.

References

[1] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.

[2] Netflix Prize [Internet]. Wikipedia, The Free Encyclopedia; 2021 Oct 10, 03:29 UTC [cited 2021 Dec 11]. Available from: https://en.wikipedia.org/w/index.php?title=Netflix_Prize&oldid=1049141816.

[3] MoviLens Data [Internet]. Grouplens. Available from: https://grouplens.org/datasets/movielens/

[4] Zaharia, Matei & Chowdhury, Mosharaf & Franklin, Michael & Shenker, Scott & Stoica, Ion. (2010). Spark: Cluster Computing with Working Sets. Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. 10. 10-10.

[5] Al-bashiri, Hael & Abdulgabber, Mansoor & Romli, Awanis. (2018). A Developed Collaborative Filtering Similarity Method to Improve the Accuracy of Recommendations under Data Sparsity. International Journal of Advanced Computer Science and Applications. 9. 10.14569/IJACSA.2018.090423.

[6] Kevin Liao [Internet]. Medium Article. Available from: https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1

[7] S. Funk, “Netflix Update: Try This at Home,” Dec. 2006; http://sifter.org/~simon/journal/20061211.html.

[8] R. Salakhutdinov and A. Mnih, “Probabilistic Matrix Fac- torization,” Proc. Advances in Neural Information Processing Systems 20 (NIPS 07), ACM Press, 2008, pp. 1257-1264.

[9] Y. Zhou et al., “Large-Scale Parallel Collaborative Filter- ing for the Netflix Prize,” Proc. 4th Int’l Conf. Algorithmic Aspects in Information and Management, LNCS 5034, Springer, 2008, pp. 337-348

[10] Yicheng Huang, Xingtu Lan, Xing Chen, and Wenzhong Guo, Towards Model Based Approach to Hadoop Deployment and Configuration, in 2015 12th Web Information System and Application Conference (WISA), Jinan, 2015, pp. 79-84

[11] Bandi, Raswitha & Jayavel, Amudhavel & Karthik, R.. (2018). Machine Learning with PySpark - Review. Indonesian Journal of Electrical Engineering and Computer Science. 12. 102-106. 10.11591/ijeecs.v12.i1.pp102-106.

Süleyman Onur Doğan
200201203
Computer Engineering