Recommendation Systems

First Research Challenge

Marcelo Mascarenhas Ribeiro de Araújo
mascarenhassrbeiro@gmail.com
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brazil

Abstract

The research challenge was developed utilizing the stochastic gradient descent algorithm and can run in a Linux machine in under 5 minutes with the latest parameter tuning. The final public score achieved, which is measured by calculating the RMSE of the predicted entries, was approximately 1.195.

1 Introduction

This document presents details regarding the implementation strategies adopted in the first research challenge of the recommendation systems course.
It also promotes a brief discussion about the achieved results in Kaggle's submissions. For more details, consult the following sections:

- Section 2 describes the development environment, the utilized data structures and the implemented algorithm.
- Section 3 contains a short complexity analysis of the major functions in the program.
- Section 4 discusses the accomplished results through parameter tweaking.
- Section 5 concludes the document.

2 Implementation

2.1 Computational Environment

The program was developed and tested in the following specifications:

- Language: Python
- OS and System Version: Pop_OS! (based on Ubuntu), version 21.10.
- Hardware: AMD Ryzen 5 3600, 16GB RAM.

2.2 Data Structures and Algorithm

The chosen algorithm to predict the scores for the challenge was the stochastic gradient descent(SGD), which can combine satisfactory performance with decent forecast capabilities. So, first of all, the implementation retrieves the number of unique users and items contained in the "ratings" data frame.
After this process, two matrices are created. One of them represents the matrix P in SGD, with dimensions MxK, where M is the number of unique users and K is the number of latent dimensions[1]. The second matrix represents the matrix Q in SGD, and has dimensions NxK, where N is the number of unique items. Afterwards, two dictionaries are created with the main purpose of mapping each unique user and item to a row or column in the matrices P and Q, respectively.
Thereafter, the next step is simply train both matrices, P and Q, and predict the scores contained in "targets" data frame for each one of the tuples, using the following equation:

$$Predicted\_Score = \underset{user\_vec \times K}{\mathrm{P}} \times \underset{K \times item\_vec}{\mathrm{Q}} \tag{1}$$

If the predicted score is higher than the maximum possible value for a score in the predicted classification system, which in this specific case is 5, than the score is replaced with the mean score of the "ratings" data frame. It is also important to note that the code has four parameters which can be alter by the user, namely - number of latent dimensions, the learning rate, the number of epochs and the beta parameter, responsible to avoid an overfitting model.

3 Complexity Analysis

Some of the main operations made during the execution of the program are the processes of reading the data frames, the creation of the two matrices P and Q, the matrices' training utilizing SGD algorithm and the procedure of generating prediction scores to the targets data frame.
All the methods described above, excepting the SGD, have the complexity of O(n), where n is the size of the read data frame. The complexity of creating a matrix could be described as $O(m \times k)$, where m is the number of rows or columns, represented by the number of unique users and items, and k is the number of latent dimensions. However, considering the latest parameters' tuning, the K is a constant number, causing the complexity to rely solely on the size of the input data frame.
Considering that the SGD iterate through the entire input data frame a specific number of epochs, the method has the complexity of $O(m \times n)$, such that m is the number of epochs of the algorithm and n is the size of the input data frame.

So, as the complexity of the program is determined by the complexity of the costlier function, the complexity of the code is the same as SGD's complexity.

[1]The number of latent dimensions is parameterizable

4 Results

The most successful combinations of parameters found was 4 dimensions, with learning rate of 0.005, 40 epochs and beta parameter evaluated in 0.02, generating the RMSE of approximately 1.195. It is interesting to note that a low number of dimensions showed a better performance than a high number of dimensions. Also, as some of the predicted scores were higher than the maximum score in the data set that a user can give to an item, which is 5, those values were replaced with the average mean in the input data set. These modification had a significant impact on reducing the RMSE. To improve the current model, a possible adjustment should be to increase the number of epochs and reduce the learning rate. Some other combinations were tried, and can be analyzed in the table below:

Dim Num	Learning Rate	Epochs	Beta	RMSE
4	0.005	40	0.02	1.19557
5	0.005	40	0.02	1.25648
8	0.005	37	0.02	1.30890
8	0.01	27	0.02	1.35199
12	0.01	25	0.02	1.60674

5 Conclusion

Considering all the aforementioned information, the implemented solution proved to be satisfactory and could be used in real scenarios, although better strategies to attack the problem of building a recommendation system do exist and aren't much harder to utilize.