Matrix Factorization Collaborative Filtering Algorithm

CAROLINA BRANDÃO, UFMG

1 INTRODUCTION

This Documentation provides a surface level analysis and discussion of my implementation of a recommender system algorithm, as well as what led my to my final result in the Research Challenge 1, a Recommender Systems' project.

2 DATA STRUCTURES

Due to the sparsity of the data in the dataset, since, in collaborative filtering, the user-item interaction matrix is often highly sparse (meaning that most entries are empty (unrated)), the choice of data structures was critical.

NumPy arrays were chosen over matrices for two main reasons: They offer greater memory efficiency compared to matrices when dealing with sparse data, allowing to store and process such data more efficiently and they provide a versatile and high-performance framework for numerical operations, making them suitable for matrix factorization and gradient descent optimizations.

Firstly, I selected the unique users and items, and mapped them to an array each, then created an array for the ratings provided. This approach ensured that the data elements were organized and systematically indexed. With that settled, I manipulated and created new arrays from them in order to perform my algorithm, which I will get into detail next.

3 ALGORITHM

Again, keeping in mind the need for a light (memory wise) implementation, and now also taking into consideration the 5 minute limit, I searched for what would fit these constraints.

The star of this recommendation system is the use of matrix factorization. This technique is widely acclaimed in collaborative filtering, as it allows the decomposition of the user-item interaction matrix into two lower-dimensional matrices: user and item matrices, each matrix capturing latent factors that influence the ratings.

To optimize the factorization process and fine-tune the model, the stochastic gradient descent (SGD) optimization method was employed. SGD is well-suited for this purpose because it allows the iterative minimization of the error between predicted and actual ratings. This iterative approach aligns with the memory-efficient implementation goal and works effectively even with large datasets. By continuously updating user and item matrices via SGD, the model's parameters are adjusted to enhance recommendation accuracy.

After a large amount of submissions, without huge improvements, I implemented mini-batches. It not only significantly decreased the amount of time taken to predict the ratings, but also minimized the error substantially. Mini-batch processing divides the dataset into smaller, manageable subsets or batches. This approach significantly reduces memory overhead by allowing the processing of a portion of the data at a time. As a result, efficient SGD updates are performed on each mini-batch, instead of performing on the whole dataset at once, striking a balance between efficiency and effectiveness, ultimately enhancing the accuracy and efficiency of the algorithm.

Author's address: Carolina Brandão, UFMG.

4 IMPLEMENTATION

Here is a more in depth description of how I implemented the algorithm.

The user and item matrices are initialized using the Xavier initialization, which sets their initial values randomly but within a range that aids efficient training and convergence, and an user and an item bias array are initialized to zeros. Then the SGD optimization with mini-batch is employed and process to iteratively update these matrices and biases, aiming to minimize the error between predicted and actual ratings. The biases calculations play a big part in the results, since they adjust them to the typical rating range of both the specific user and the item in question.

Regularization terms, L1 (Lasso) and L2 (Ridge), are introduced into the model during training. L1 regularization encourages sparsity in the learned parameters, and„ L2 regularization adds stability to the model by constraining the magnitude of parameter values. These regularization techniques prevent overfitting, enhance generalization, and ensure robust and reliable predictions. Additionally, ratings are shuffled during each epoch to introduce randomness and prevent again overfitting during training.

Predicted ratings in the recommendation system are calculated by taking the dot product of the user and item vectors from the user and item matrices. Additionally, the user and item biases are added to this dot product, along with the dataset's mean rating. These biases represent deviations from the overall mean rating in the dataset, helping to fine-tune predictions. Additionally some curation happened after the model final predictions, such as keeping the results between 1 and 5 and rounding predictions that were really close to some integer.

5 COMPLEXITY

$O(\text{num\_epochs} \cdot \text{batch\_num} \cdot \text{num\_factors} \cdot (M + N))$ where M and N denote the total number of unique users and items, respectively. This complexity is justified by the iterative nature of the algorithm, where at each epoch, for each mini-batch, the user and item matrices are updated based on the computed gradients.

6 RESULTS AND CONCLUSION

To achieve my final result, an extensive series of experiments were conducted, focusing on fine-tuning hyperparameters and batch sizes. Throughout these iterations, the Mean Squared Error (MSE) served as an important metric for evaluating performance and understanding the model's behavior. I observed that MSE values ranging from 0.67 to 0.73 consistently led to the most promising results. Ultimately, the result was a final recommendation system with an MSE of 0.70385.

The combination of the hyperparameters in the submission is: Learning rate = 0.0095, number of epochs = 19, number of latent factors = 15, Lambda L2 = 0.095, Lambda L1 = 0.01 and the Batch size is 400.

In conclusion, the algorithm successfully leveraged matrix factorization, SGD optimization, and mini-batch processing to deliver efficient and accurate recommendations within the limitations.

REFERENCES

[1] Rodrygo Santos, *PowerPoints in RecSys*, *UFMG*, 2023.
[2] Yehuda Koren, Robert Bell, Chris Volinsky, *Matrix factorization techniques for recommender systems*, *Computer*, 2009.
[3] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, John Riedl, *An algorithmic framework for performing collaborative filtering*, *SIGIR*, 1999.
[4] Paolo Cremonesi, Yehuda Koren, *Performance of recommender algorithms on top-N recommendation tasks*, *ACM*, 2010.

Received 25 September 2023