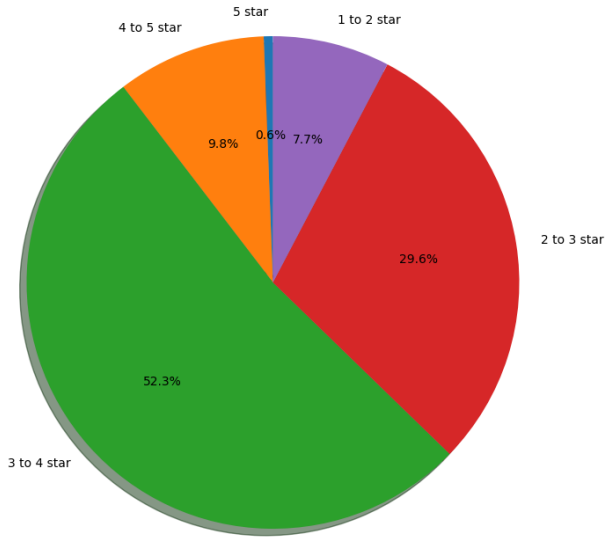
Neural collaborative filtering for movie recommendation engine

Jayveersinh Raj
BS20-DS-01
j.raj@innoplis.university

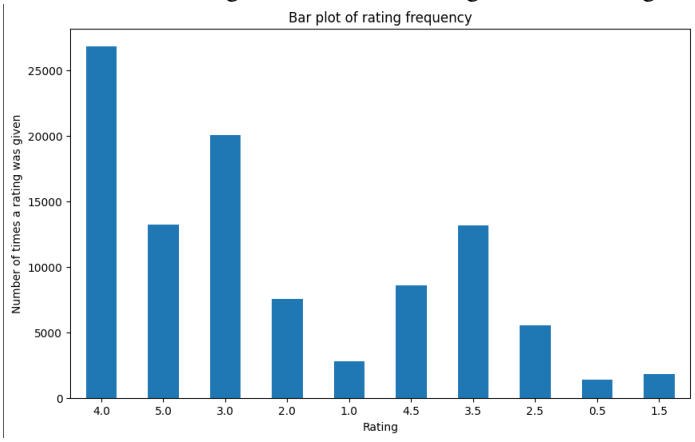
1. Introduction

Recommender systems, which are extensively used by a variety of online services, such as social networking, online news, and e-commerce, are essential in reducing information overload in the age of exponentially expanding information. Collaborative filtering, which models users' preferences on things based on their prior interactions (e.g., ratings and clicks), is essential to a tailored recommender system [1, 2]. This project investigates the application of deep neural networks, as opposed to a manual method used by several other studies [3, 4], for learning the interaction function from data. It has been demonstrated that neural networks may approximate any continuous function [5].



2. Data analysis

The movie lens dataset was found to be having 100836 samples. The ratings dataset with **userId movieId** and **rating** was used for training as well as inference, and then the recommended movie ids were mapped to the title. More details regarding it would be followed in the paper. However, only 610 were unique users while there were 9724 unique movies in the dataset. The following chart shows the histogram of the ratings.



It can be seen that most of the users have rewarded movies they watched with a 4 star rating and followed by 3 star and 5 star. For a better perspective a pie chart is as follows:

Train set was 100226 samples for training. **Test set** however has 610 samples which were unique from the train set. The rest were corrupted or had null values.

3. Model implementation

The model is based on [6] with the layers as [64,32,16] in the multi-layer perceptron. The following is the model architecture which is approximately 6 million parameters.

```
MLP(
  (user_embedding): Embedding(611, 32)
  (item_embedding): Embedding(193610, 32)
  (MLP_model): Sequential(
    (0): Linear(in_features=64, out_features=32, bias=True)
    (1): ReLU()
    (2): Linear(in_features=32, out_features=16, bias=True)
    (3): ReLU()
    (4): Linear(in_features=16, out_features=8, bias=True)
    (5): ReLU()
  )
  (predict_layer): Linear(in_features=8, out_features=1, bias=True)
  (Sigmoid): Sigmoid()
)
```

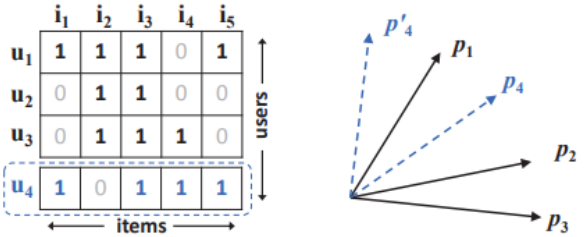
As it can be seen from above, both the user and item embeddings are 32 dimensional.

3.1. Learning from implicit data

Following [6] a user-item interaction matrix was defined based on the formula below from [6].

$$y_{ui} = \begin{cases} 1, & \text{if interaction (user } u\text{, item } i\text{) is observed;} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The issue with implicit feedback recommendations is expressed as the task of predicting the unobserved entry scores in Y, which are utilized to rank the items. Model-based strategies make the assumption that data can be produced (or explained) by a foundational model. [6] Authors of [6] describes the difference between using matrix factorization versus latent space representation in the following figure.



(a) user–item matrix (b) user latent space

Figure 1: An example illustrates MF's limitation. From data matrix (a), u_4 is most similar to u_1, followed by u_3, and lastly u_2. However in the latent space (b), placing p_4 closest to p_1 makes p_4 closer to p_2 than p_3, incurring a large ranking loss.

Hence, the figure below is the figure of the neural network that creates neural collaborative filtering.

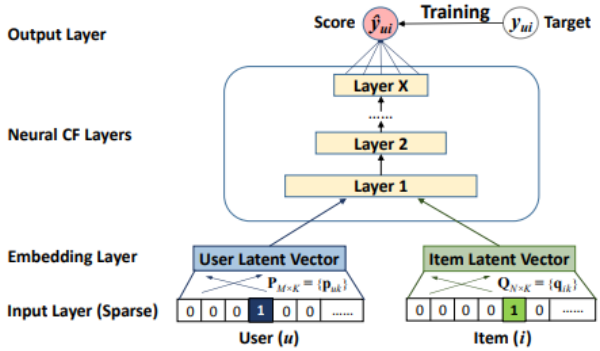


Figure 2: Neural collaborative filtering framework

3.2. *Model conclusion*

In a nutshell an NCF instantiation that uses a multi-layer perceptron (MLP) to train the user-item interaction function in order to investigate DNNs for collaborative filtering.

4. Advantages and disadvantages

4.1. *Advantages*

- Simplicity: The model takes only user ID, movie ID, and ratings as input, hence, not data-extensive. Moreover, the model is approximately 6 million parameterized.
- Scalability: The solution is scalable mainly because it takes into account the user, and its past interactions without the need for extensive user data.
- The ability to capture complex patterns in user-item interactions.

4.2. *Disadvantages*

- Cold Start Problem: The model might struggle when dealing with new users or items that have little to no interaction history. Since it relies on past interactions to make predictions, it may not perform well for users or items with limited data.
- Lack of Interpretability: The model operates as a black box, making it challenging to interpret why certain recommendations are made. If interpretability is crucial for your application, this might be a drawback.
- Limited Context Awareness: The model only considers user-item interactions and doesn't take into account contextual information like genres, temporal trends, or user demographics. For certain recommendation scenarios, incorporating additional features may lead to more accurate predictions.

However, if a model is "good"depends on your specific use case, data characteristics, and performance metrics. It might be effective if one's primary goal is to provide personalized recommendations based on past interactions. However, one could improve recommendations by exploring more advanced models that incorporate such more features (if it would be better or not is a subject of experimentation).

5. Training Process

The model is trained using pytorch library, the following are the details:

- epochs: 10
- batch size: 32
- device: cuda (Tesla T4)
- objective function: Cross Entropy
- optimizer: Adam with default learning rate
- training time: 587.61 seconds (9.8 minutes)
- k for top k recommendations: 10

6. Evaluation

6.1. *Metrics*

Two metrics were used for evaluation.

6.1.1. *Hit rate (HR)*

- **Definition**: Hit Rate is a binary metric that measures the proportion of correct recommendations in a given recommendation list. It answers the question: "Did the user find at least one relevant item in the top-k recommendations?"

- **Calculation**: HR is calculated by counting the number of times at least one relevant item appears in the top-k recommendations and dividing it by the total number of users.
- **Interpretation**: A higher HR indicates that the recommendation system is successful in suggesting at least one relevant item to users.

6.1.2. Normalized Discounted Cumulative Gain (nDCG)

- **Definition**: NDCG is a ranking metric that evaluates the quality of the entire ranked list of recommendations. It considers both the relevance of the items and their positions in the list.
- **Calculation**: NDCG is calculated by summing up the discounted relevance of items in the recommendation list and normalizing it by the ideal DCG (Discounted Cumulative Gain). The relevance is typically binary (relevant or not).
- **Interpretation**: A higher nDCG indicates that the recommendation system not only suggests relevant items but also prioritizes them higher in the list. It provides a more nuanced evaluation compared to HR, considering the ranking of recommendations.

Both of the formula are present in the below image.

$$Hit\ Ratio = \frac{of\ Hit\ User}{of\ User}$$

$$CG_K = \sum_{i=1}^{K} relevance_i \qquad DCG_K = \sum_{i=1}^{K} \frac{relevance_i}{\log_2(i+1)} \qquad IDCG_K = \sum_{i=1}^{K} \frac{relevance_i^{opt}}{\log_2(i+1)}$$

$$NDCG_K = \frac{DCG}{IDCG}$$

7. Results

Below are the numbers on the movie lens dataset using the metrics above as mentioned in evaluation.

- **Hit Rate (HR)**: 0.762
- **Normalized Discounted Cumulative Gain (nDCG)**: 0.537

In [6] the authors have a different architecture, but for a 3-layer MLP with k = 10, and 64 neurons each, they have the following results:

- **Hit Rate (HR)**: 0.702 (50 iterations)
- **Normalized Discounted Cumulative Gain (nDCG)**: 0.426 (50 iterations)

The model architecture presented in this project has better results than the authors.

8. References

1. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In WWW, pages 285–295, 2001.
2. H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua. Discrete collaborative filtering. In SIGIR, pages 325–334, 2016.
3. X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In SIGIR, pages 549–558, 2016.
4. Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In KDD, pages 426–434, 2008
5. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, 1989.
6. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural Collaborative Filtering. Proceedings of the 26th International Conference on World Wide Web.