# User-based Recommender System using Matrix Factorization and ANN

Firas Jolha
f.jolha@innopolis.university
May. 2021

---

**Course: Advanced Machine Learning**
**Project Github Link:** https://github.com/firas-jolha/hw2

---

## Motivation

One of the benefits of AI is to help people in decision making. To make good decisions, the user of the system needs recommendations, these recommendations should be data-driven and unbiased. A lot of researchers have conducted experiments to improve the recommendations with different objectives. It is common that the most powerful learning algorithm in machine learning is ANN(Artificial Neural Network). In this work we investigate and implement two algorithms for building recommendations (in fact, building the full rating matrix). The first algorithm is Matrix Factorization which could be trained using ALS(Alternating Least Squares). The second is a deep neural network which is called neural collaborative filtering.

Our recommender system is applied on a dataset of movies, the objective is to build the full rating matrix. The report is divided into sections as follows:

1) **Description** gives an explanation of our work and specifies our design choices.
2) **Results and Discussion**
3) **Used Tools and Technologies**
4) **Conclusion**

## Description

In this work, we implemented a user-based Recommender system using collaborative filtering. The system takes some ratings as training data and tries to build the full rating matrix in which every user has a rating for each movie (item). We used two collaborative filtering methods but the data preprocessing steps are shared between them. In this section, we will explain the stages of data preprocessing and the implemented methods of collaborative filtering. The results will be discussed in the next section.

### 1. Data Description and Preprocessing

The input to our recommender system is user ID and the output is a list of movies with ratings for that user.

Our dataset consists of users and movies with ratings. The training set has 6687 users and 5064 movies and 761972 ratings. It is clear that we don't have the rating of every user to every movie. Our objective in this project is to fully build this ratings matrix. A sample of training data is shown in table 1. If we do a simple analysis on the values of user ids and movie ids, we see that the ids have different scales and don't start from zero. To overcome this issue we remap or reset the ids to the basic indexing such that ids of users and movies are in a range [0, n-1], where n is the number of users and movies respectively.

| | **userId** | **movieId** | **rating** |
|---|---|---|---|
| 0 | 1 | 32 | 3.5 |
| 1 | 1 | 47 | 3.5 |
| 2 | 1 | 50 | 3.5 |
| 3 | 1 | 253 | 4.0 |
| 4 | 1 | 260 | 4.0 |

Table 1: A sample of training data

We use the new values of user and movie ids to build the rating matrix. We use a sparse matrix coo_matrix from scipy library to define the rating matrix for fast computation.

## 2. Matrix Factorization using Alternating Least Squares

This method decomposes the rating matrix into multiplication of two matrices P and Q such that R can be reconstructed using the equation R_hat = P @ Q.T

The idea in this algorithm is to learn the matrices P and Q from given ratings. This could be accomplished using ALS (Alternating Least Squares) algorithm. This method works by alternating the procedure of adjusting P and Q matrices. This means when we adjust P, we consider Q as fixed and vice versa. This facilitates the computation of gradients that are used to update the values of the matrices.

## 3. Neural Collaborative Filtering (NCA)

In this method, we train a neural network which consists of two embedding layers for users and items followed by a stack of fully connected layers, the last layer is a single output linear layer in which the output is mapped to the ratings range [1-5].

## Results and Discussion

In this section, we will show the obtained results and discuss more on them. First we will present the results of matrix factorization which we denote it as basic collaborative filtering, then the results of neural collaborative filtering will be presented.

### 1. Basic Collaborative Filtering (Matrix Factorization using ALS)

The configurations in the experiments are shown in table 2. The values in this table are chosen empirically after a series of experiments.

| **Hyperparameters** | **Value** |
|---|---|
| Latent space dimension | 7 |
| Learning Rate | 1e-6 |
| Regularization Rate | 1e-12 |
| Epochs | 30 |

Table 2 : Configurations of the experiments for basic collaborative filtering method

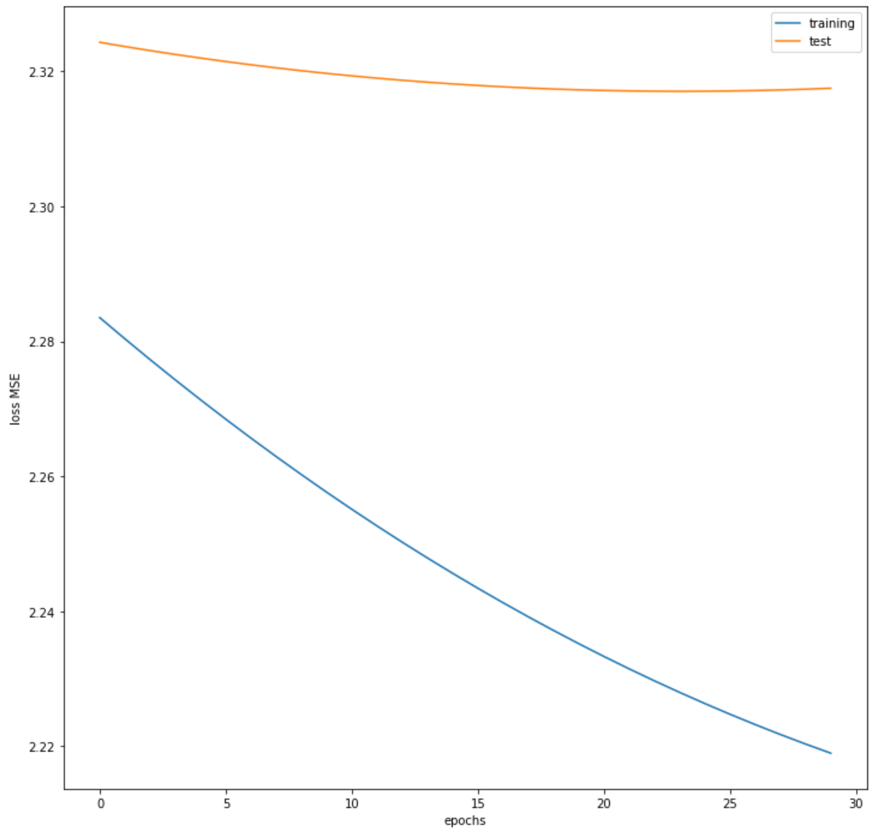

**Figure 1: Training loss vs. Test loss for basic collaborative filtering**

The figure 1 shows the decreasing loss for both training and test dataset trained for 30 iterations. It is clear that this model is underfitting. The high learning rate increases the step toward the minimum value in the objective function and makes so increases the cost. The low learning rate slows down the learning procedure.

The latent space dimension has an impact on the loss such that increasing the dimension will increase the loss, but decreasing it will let the algorithm to give up learning early.
It is better to increase the epochs, as it is shown in the figure 1 that the next epochs could decrease the loss more.

The average loss for test data in case of basic collaborative filtering was `2.31702`

## 2. Neural Collaborative Filtering

In this case, our model is a neural network and has the following schema.

```
NCA(
  (embed_user): Embedding(6687, 7)
  (embed_item): Embedding(5064, 7)
  (fc_layers): ModuleList(
    (0): Linear(in_features=82257, out_features=64, bias=True)
    (1): Linear(in_features=64, out_features=16, bias=True)
    (2): Linear(in_features=16, out_features=8, bias=True)
  )
  (dropout): Dropout(p=0.2, inplace=False)
  (output): Linear(in_features=8, out_features=1, bias=True)
  (output_f): Sigmoid()
)
```

The loss for training data run for 10 epochs is shown in figure 2.

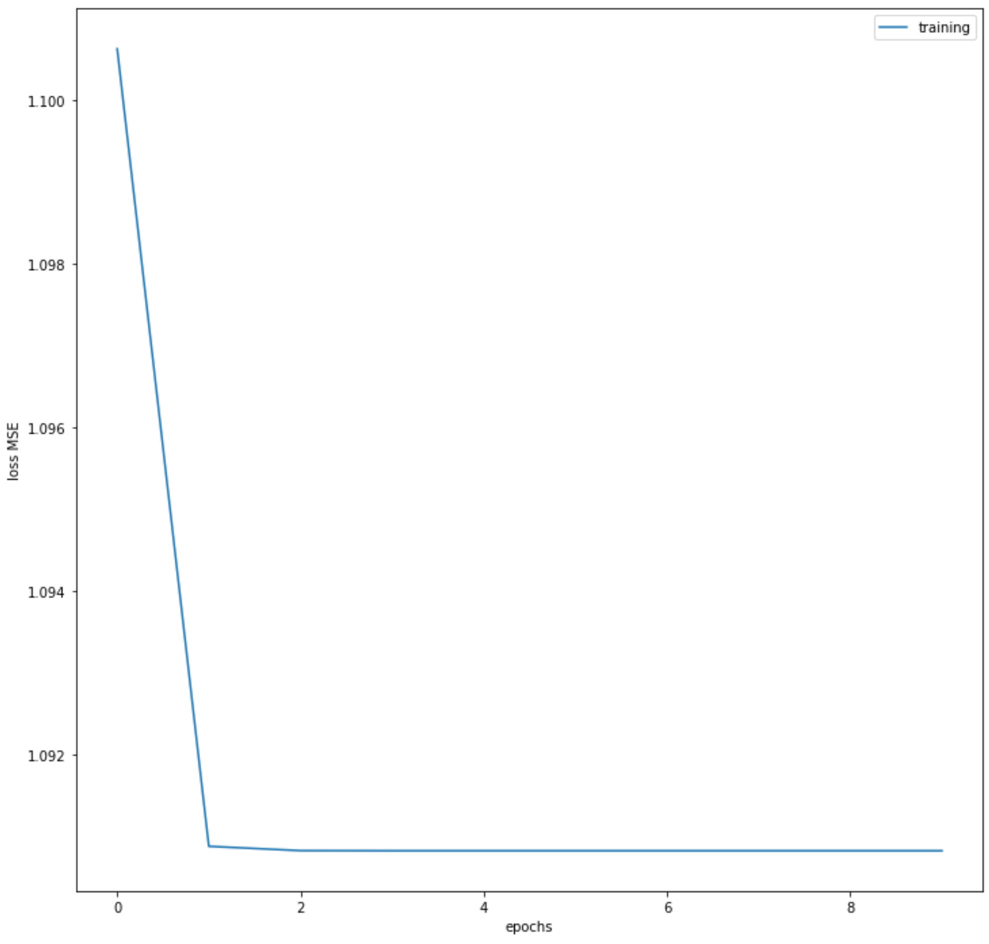

Figure 2: Training loss for Neural Collaborative Filtering

The average loss for test data in case of neural collaborative filtering is `1.09135`

Neural CF outperforms basic CF.

## Used Tools and Technologies

Google Colab, Python, Atom editor, Github and Pytorch.

## Conclusion

This assignment was a good practice on designing a recommender system using ALS and NCF. The additional preprocessing that is used is the one hot encoding of user and movie ids but it makes the model more complex in training and testing.