Assignment 4 - Neural Network Recommendation System

Team name: M2 Robo

Chan Kwan Yin	Lee Chun Yin	Chiu Yu Ying
3035466978	3035469140	3035477630
Team leader		

1. Background

Among all collaborative filtering techniques, matrix factorization (MF) become a standard approach to the model-based recommendation. Many research teams have put their efforts on enhancing it to break the accuracy record.

However, it has several limitations such as the cold-start problem – how the system provide recommendations to new users, how to recommend when new item is added without feature vector or embedding [2]. Also, the use of dot product for recommending popular items will lead to the tendency of recommending popular items to those users without specific user interests. With the use of inner product of User and item feature embeddings, MF limits the capture of the complex relations of the users and items interactions [1].

In this assignment, we aim to implement the new approach, Neural Collaborative Filtering (NCF), proposed by He team in 2017 [1]. They suggested that their design of deep neural network can overcome the limitation of MF.

1.1. Neural Network Model

This model adopts the multi-layer perceptron (MLP) to model the user-item interaction y_{ij}.

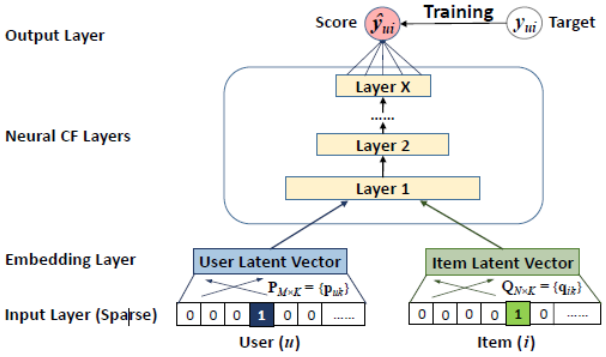

Figure 1. Neural Collaborative Filtering Model Design [1]

We take two vectors from user and item respectively in the input layer and then they will be embedded to pass the neural collaborative filtering (neural CF) layers. In our neural CF layers, we first do matrix multiplication. The product then passes into the rectified linear unit (ReLU) and next passes through an activation function $f1(x)$.

This model takes two sparse feature vectors from representation of users and items respectively.

- User vector:v_i with m users
- Item vector:v_j with n items.

The identity of users and item will be transformed to a binarized vector with one-hot encoding.

With such generic feature representation, the model can handle various modelling types such as content-based and neighbor-based. The limitation of the cold-start problem can be solved by using the content features to represent users and items.

After the input layer, the next is the embedding layer. It is fully connected to project the sparse vector representations (v_i and v_j) for users and items in the input layer into dense vectors.It can also be seen as the latent vector matrix, P for users and Q for items, in the context of latent factor model.

The obtained latent vectors from the embedding layers will then be fed into the the neural CF layers,and be mapped to predicted probability scores by mapping fuction ϕ_X. Those layers will get the complex user-item relations from the vectors.

The final output layer produces a predicted score $\hat{y_{ij}}$, which is the predicted user-items interactions of y_{ij} through the mapping function ϕ_{output}.

The formula of the NCF is as follow:

$$\hat{y_{ij}} = f(P^T v_i Q^T v_j | P, Q, \Theta_f)$$

The variables with m users and n items are denoting as:

- $P \in \mathbb{R}^{(m \times K)}$ latent factor matrix for users from embedding layer.
- $Q \in \mathbb{R}^{(n \times K)}$ latent factor matrix for items from embedding layer.
- f interaction function.
- Θ_f model parameters of the interaction function f

For interaction function f, it can be formulated:

$$f(P^T v_i, Q^T v_j) = \phi_{output}(\phi_X(...\phi_2(\phi_1(P^T v_i, Q^T v_j))...))$$

The mapping functions are as follow:

- ϕ_{output}: the mapping function for output layer
- ϕ_x the mapping function for the xth neural CF layer. X is the number of the layers

2. Technical Details

The training data set provided by Netflix consists of more than 100 million ratings with 17770 movies and 480189 users.

2.1. Data preprocessing

The movies are loaded into a list with columns of Movie ID, User ID and Rating. Then, it will be fed into the test loader with batch size.

2.2. Optimization and scheduler

We select Adadelta as our optimizer.

To have a better performance on the converge of loss function,we implement a scheduler with different combinations of hyperparameters of scheduler

2.3. Hyperparameters selection for scheduler

There are two hyperparameters, namely

- step size
- γ

2.3.1 Step size

tbc

2.3.2 γ

tbc

2.4. Hyperparameters selection for model

In this MLP model, there are five hyperparameters to be selected, namely

- Learning Rate (α)
- Regularization (λ)
- Rank of factorization (k)
- Number of epoch (x)
- Batch Size (N)

2.4.1 Learning Rate (α)

The learning rate indicates the speed at which the model learns. It controls how much to change the modal to respond the estimate error each time when the model weights are updated.

Rate with too small value may cause a long training time while a large value may allow the modal learning too fast to have less accurate or even not meaningful result.

2.4.2 Regularization (λ)

2.4.3 Rank of factorization (k)

Intuitively, each rank represents some characteristic of a user or a movie. Users and movies that interact strongly on a rank would be affected.

A large value of k increases the training time, memory consumption and probably the chance of overfitting, while a low value of k reduces the representativeness of the model and hence reduced accuracy.

2.4.4 Number of epoch (x)

The number of epochs defines the number of times that the modal will work through the whole training dataset. Since we already report the training score for each epoch, it is adjusted to a value such that the training score appears to converge negligibly.

2.4.5 Batch size (N)

The batch size defines the number of samples to be passed through before the update of model parameters.

The small batch size usually can give a better performance since the model can start learning with a small subset and have more cycles to improve.

However, the smaller batch size costs the slower learning and hence more training time.

2.5. Predictive test set score (RMSE)

The model is evaluated by computing the RMSE between predicted and actual rating values:

$$\text{RMSE} = \sqrt{\sum_{(i,j)\in E} \frac{\left(\hat{r}_{ij} - r_{ij}\right)^2}{|E|}}$$

where E is the set of retained evaluation data.

3. Model performance

The following table exhaustively lists our test results. Train RMSE and Test RMSE refer to the RMSE value calculated for the training dataset and testing dataset respectively.

100,40 Epoch 39, mse = 0.8843765864636746 Test RMSE: 1.0292723480987729

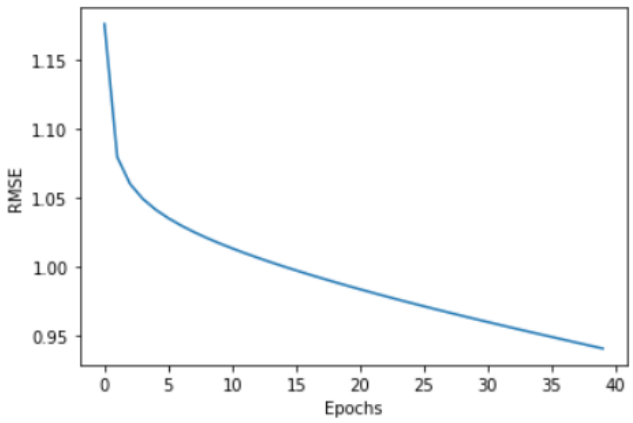

Figure 2. RMSE graph for rank =100 and epochs = 40

200,40 Epoch 39, mse = 0.8379997956340094 Test RMSE: 1.062824159761129

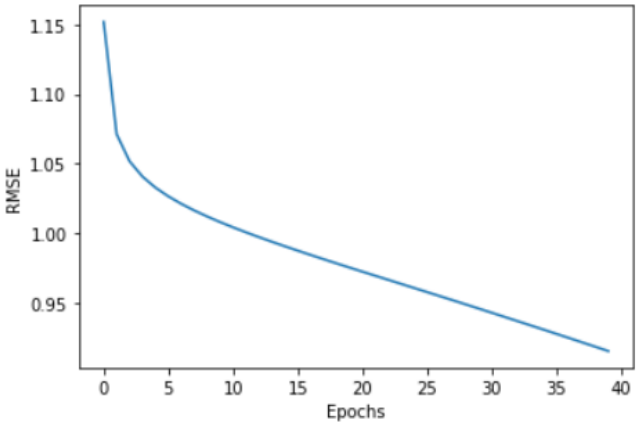

Figure 3. RMSE graph for rank =200 and epochs = 40

50,40 Epoch 39, mse = 0.9447392710872188 Test RMSE: 1.0427012386780807

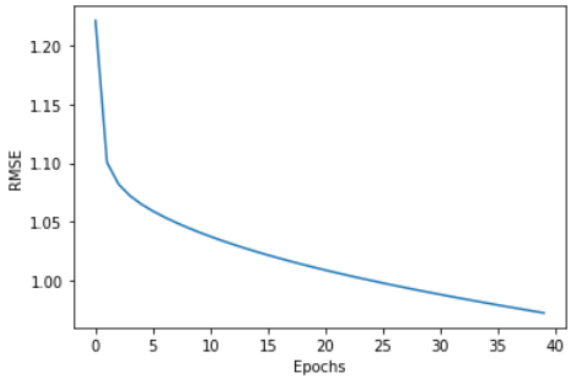

Figure 4. RMSE graph for rank = 50 and epochs = 40

100,60 Epoch 59, mse = 0.9101701523855344 Test RMSE: 1.188729648171058

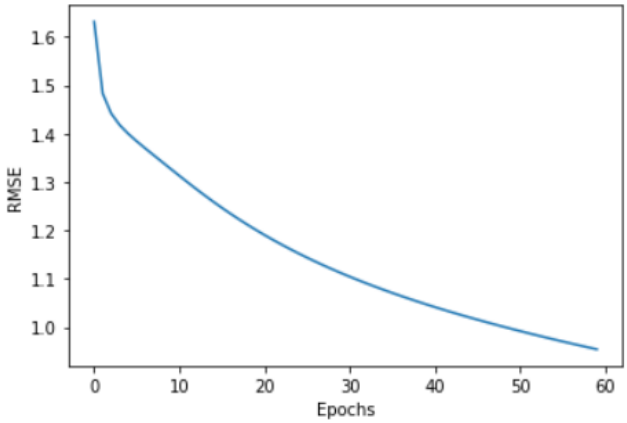

Figure 5. RMSE graph for rank = 100 and epochs = 60

References

[1] Liao L. Zhang H. Nie L. Hu X. Chua T. S. He, X. Neural collaborative filtering. *Proceedings of the 26th international conference on world wide web*, 2017. 2

[2] Yang S. H. Zha H. Zhou, K. Functional matrix factorizations for cold-start recommendation. *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011. 2