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.

2. Neural Collaborative Filtering (NCF) model

This is the model proposed by He team in 2017 [1]. Their proposed model uses a combination of matrix factorization and multi-layer perceptron techniques, and the hidden layers from the two techniques are combined by a fully-connected layer. However, their paper mainly focuses on the prediction of *implicit user feedback*, that is, they are predicting the values of y_{ui} where $y_{ui} = 1$ when the obervation (user u, item i) is observed, and 0 when the observation (user u, item i) is not observed. Thus, at the final stage of their proposed model, the output of the fully-connected layer is passed into a sigmoid function, such that the model only outputs values within $[0, 1]$.

The problem of predicting implicit feedback is different from the problem we wish to solve in the Netflix dataset, because in the Netflix problem we wish to predict actual ratings from $[1, 5]$. Nevertheless, we still think that He's paper provides a good starting point - We modify He's model by removing the final sigmoid function and changing the loss functions used directly, such that the neural network solves a regression problem instead of the original binary classification problem.

The model proposed by He is split into 3 parts - Generalized Matrix Factorization, Multi-Layer Perceptron, and Neural matrix factorization. In fact, the third part is a combination of the previous two parts. For this part of our project, we will follow the same workflow as He and implement the 3 parts separately.

2.1. Model - Input Layer

The input of the model are the user ID and item ID transformed to a binarized vector with one-hot encoding.

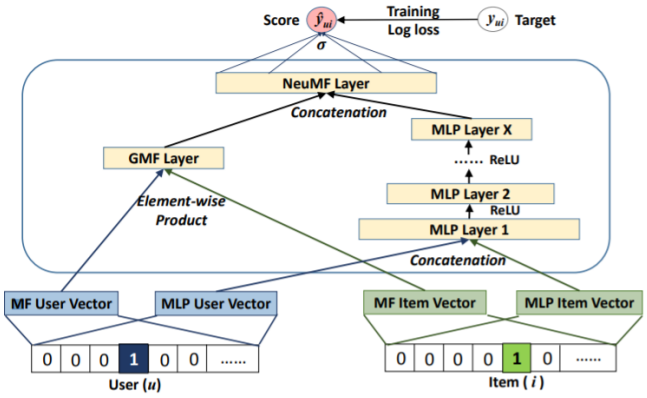

Figure 1. Neural Collaborative Filtering Model Design [1]

2.2. Model - Embedding Layer

For each user and each movie, we train two sets of embedding latent vectors. This is because one set of latent vectors will be passed into the itemwise multiplication part of the model (in He's paper, it is called Generalized Matrix Factorization (GMF)), and the other set of latent vectors will be passed into the neural network part of the model (in He's paper, it is called Multi-Layer Perceptron (MLP)).

2.3. Model - Generalized Matrix Factorization (GMF)

In this part of the model, the user latent vector and item latent vector are combined via itemwise multiplication, similar to that in the SVD matrix factorization model. The output of this part is another vector with the same size at the latent vectors and is passed on to the final output layer of the model.

2.4. Model - Multi-Layer Perceptron (MLP)

In this part of the model, the user latent vector and item latent vector are first concatenated. Afterwards, the whole concatenated vector is passed through several layers of fully connected hidden linear layers, using the ReLU activation function. the output of this part is a vector dependng on the output size of the last hidden layer and is passed on to the final output layer of the model.

2.5. Model - Output Layer

The final output layer takes in the vectors produced by the GMF and MLP parts of the model and concatenates them. The output of this part, which is the final output of the whole model, is a weighted sum of the vector elements, where the weights are model parameters to be learned. The final output layer produces a predicted score $\hat{y_{ui}}$.

2.6. Model - Formula

The formula of the GMF 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 MLP f, it can be formulated as:

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

The mapping functions are as follow:

- ϕ_{output}: the mapping function for output layer
- ϕ_x the activation function for the xth hidden neural FC layer. x is the number of the layers

3. Technical Details

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

3.1. 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.

3.2. 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

3.3. Hyperparameters selection for scheduler

There are two hyperparameters, namely

- step size
- γ

3.4. 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)

3.4.1 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.

3.4.2 Regularization (λ)

3.4.3 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.

3.4.4 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.

3.4.5 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.

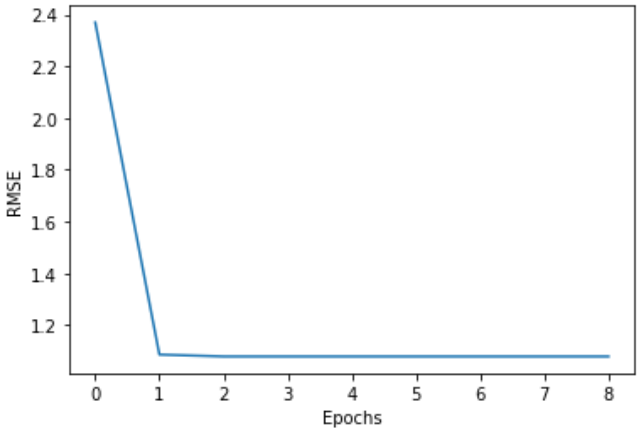

Figure 2. RMSE graph for model 1

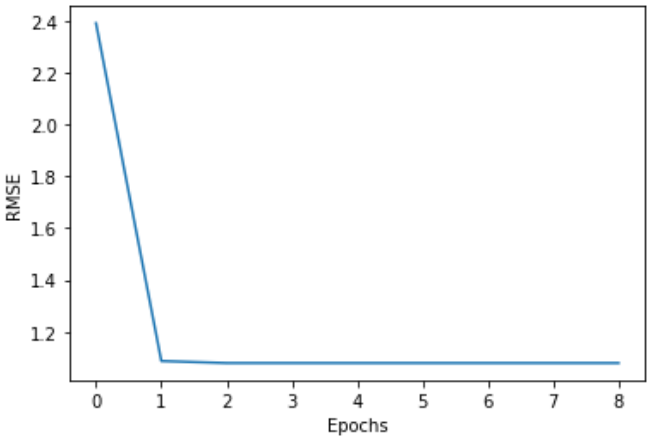

Figure 3. RMSE graph for model 2

3.5. 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.

4. Model performance

4.1. Findings

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.

No.	k	α	γ	N	Epochs	Train RMSE
1	100	1	0.7	65535	9	1.16384
2	300	1	0.7	65535	9	1.16491
3	100	0.1	0.9	65535	9	1.16536
4	100	1	0.7	4096	9	1.16368
5	100	1	0.7	1048576	8	5.14467

4.2. Discussion

When comparing the results of tuning the value of rank from 100 to 300, we can see the RMSE curves have a similar shape in general. After epoch 0, both MSE values drop significantly by around 4.5 units and then the values remain constant. The shape becomes smoother than previous results by tuning learning rate to 0.1 and γ to 0.9. The curve of decreasing the batch size to 4096 has a similar shape with the first and second one but the extent of drop is much smaller (by 0.3). By increasing the batch size to 1048576, the RMSE drops constantly (remains a straight line) from mse $= 13.17$ to mse $= 5.144$.

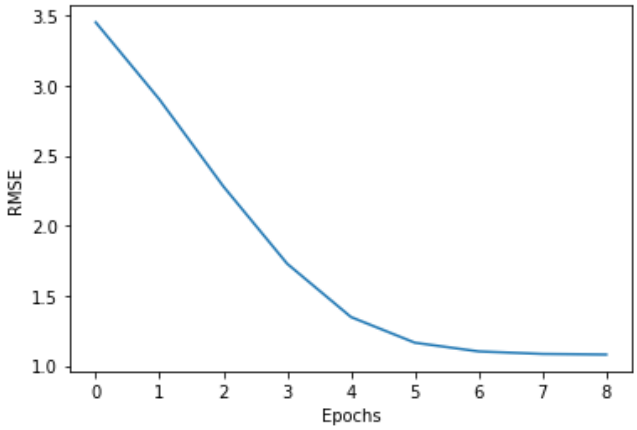

Figure 4. RMSE graph for model 3

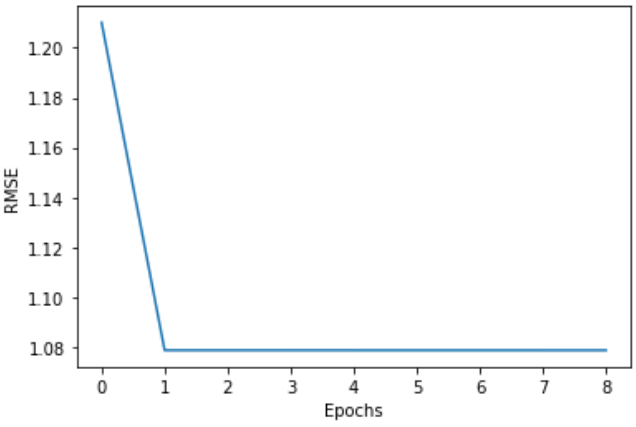

Figure 5. RMSE graph for model 4

Overall, the first one with (rank = 100, learning rate = 1.0, gamma = 0.7, batch size = 65535) has the lowest mse (mse = 1.163) after training. The highest belongs to the (rank = 100; learning rate = 1.0, gamma = 0.7, batch size = 1048576)

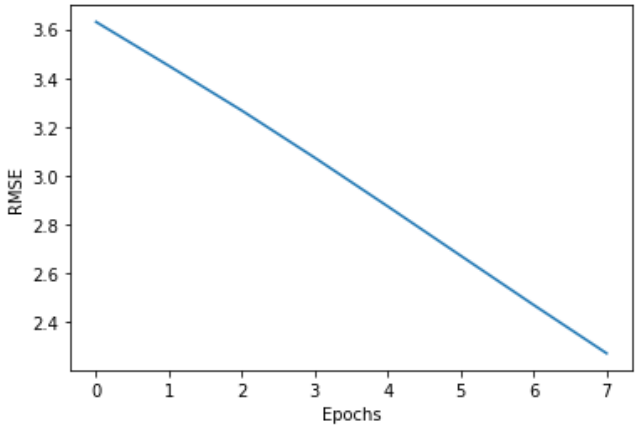

Figure 6. RMSE graph for model 5

References

[1] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017. 2

[2] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval*, SIGIR '11, pages 315–324. ACM, 2011. 2