Final Report of EE 260 (Deep Learning): Deep User-Item Interaction Learning for Recommendation

Tianyu Chen 862150667 tchen228@ucr.edu
Jin Yang 862186782 jyang361@ucr.edu
Xiaojing Weng 862187370 xweng003@ucr.edu
Jianyi Yang 862121706 jyang239@ucr.edu

Abstract

User-item interaction learning is pretty important for recommendation systems. One challenge for user-item interaction learning is user-item feature embedding. Besides, the cold-start problem for online recommendation is another challenge. In this project, we focus on these two challenges and use deep learning tools to solve them. Concretely, we add additional features to basic neural collaborative filtering and train this model based on offline dataset. Then with the embedded features, we exploit a deep bandit algorithm– Neural UCB– for efficient online recommendation for a new user. We do the experiments on the movies dataset [1] and compare the testing error of our neural collaborative filtering with that of linear methods and basic neural collaborative filtering. Also, we evaluate the regret of Neural UCB for online recommendation.

1 Introduction

Nowadays, recommendation system plays an important role in improving user experience of many internet applications such as movie, music, advertisement. Usually, a recommendation algorithm chooses an item for a user by predicting the rating of the user on the item. Hence it is important to learn the user-item interaction for a recommendation system.

The user-item interaction can be obtained by linear methods such as Singular Value Decomposition (SVD) and Non-negative Matrix Factorization(NMF)[5], but this requires enough user-item data. Recently, the deep learning model – neural collaborative filtering[11] has been widely used in useritem interaction learning and can capture complex relationships between user rating and user-item features. Therefore, if enough rating data is available for offline deep user-item interaction training, recommendation systems can predict user ratings very well. Nevertheless, a challenge for deep recommendation systems is the cold start problem in online settings. This always happens when the recommendation system need to serve a new user or item and not enough data is available to train a deep model. To find the best item for a new user efficiently, bandit learning [6] which balance the exploitation and exploration of user-item interaction is widely used for recommendation.

In this work, we focus on user preference learning and aim to train a deep model with user and movie features as input and movie rating as output. Further, we exploit a deep bandit algorithm – Neural-UCB [15] for online recommendation. The dataset we will use is the movies dataset[1]. The large dataset consists of 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users, Including tag genome data with 12 million relevance scores across 1,100 tags. The small dataset comprises of 100,000 ratings and 1,300 tag applications applied to 9,000 movies by 700 users. We will use a subset of the dataset to test the accuracy of the model.

The contributions in this project are listed as follows:
i. Neural Collaborative Filtering is implemented for movie rating prediction. We add new features

including genres and keywords for better offline training. With our architecture, the testing RMSE is reduced compared with the linear method SVD and basic neural collaborative filtering.
ii. Neural-UCB is exploited for online recommendation. Instead of training an online model from raw features, we use the user-item features extracted from the dense layer of our offline-trained neural collaborative filtering model. Then an efficient shallow neural network can be used for online prediction. Simulations show that using our online model, the recommendation regret decreases with time.

2 Related Work

There are many literatures about predicting a personalize rating for recommendation systems. The famous traditional method is collaboritive filtering [11] including memory-based methods and model-based methods like Singular Value Decomposition (SVD) and Non-negative Matrix Factorization(NMF)[5].

Collaboritive filtering is effecient to use but has limited ability inference based on available content, so many recent works focus on deep item recommendation. [4] uses a multi-layer perception to learn the user-item interaction. [10] leverages neural networks to learn a user's pairwise preference between items.

Recently, many new deep learning techniques are used in recommendation. For example, to solve the problem of data spasity, deep generative models are applied to generate implicit feedback data[8, 14]. Besides, some side information such as knowledge graph [13]is used to tackle the data spasity and cold-start problem. Recommendation based on graph learning.

Bandit learning has been widely used in online recommendation. [7] exploits linear contextual UCB in recommendation and evaluate the performance on Yahoo user click dataset. Besides UCB, Thompson sampling [3] is also an important bandit policy which can be used in more cases. Kernel UCB [12] generalizes bandit algorithm to non-linear case but it requires the knowledge of kernel functions. We mainly use Neural-UCB [15]in this project because it is a contextual bandit algorithm which can capture complex non-linear relationships.

3 Problem Formulation

In this section, we describe the mathematical formulation of the movie recommendation problem. The information provided to the recommendation algorithm includes user ID: $u_1, u_2, \cdots, u_N$, movie ID: $m_1, m_2, \cdots, m_M$ and other movie features like genres: $g_1, g_2, \cdots, g_M$. The ratings with respect to user i and item j can be represented as $r(u_i, m_j, g_j)$. In this project, we use a fully connected neural network $f(x_{i,j}, \theta)$, where $x_{i,j} = [u_i, m_j, g_j]$ is the input and θ is the parameters of the neural net, to approximate the rating function. The recommendation system only has access to the ratings in the training dataset $\{r(x_{i,j}), (i,j) \in \mathcal{D}_{\text{train}}\}$. With the trained network $f(x_{i,j}, \hat{\theta})$, the system can predict the ratings of user-item interaction in the test dataset $\{r(x_{i,j}), (i,j) \in \mathcal{D}_{\text{test}}\}$. The performance metric for prediction is the Root Mean Square Error (RMSE) which is expressed as

$$RMSE = \sqrt{\sum_{(i,j)\in\mathcal{D}_{\text{test}}} \Big(r(x_{i,j}) - f(x_{i,j}, \hat{\theta})\Big)^2 \Big/ \sum_{(i,j)\in\mathcal{D}_{\text{test}}} r(x_{i,j})^2}. \tag{1}$$

In the offline training of this project, we focus on minimize RMSE metric because an accurate rating prediction is crucial for recommendation.

Additionally, we consider the online recommendation problem where the rating predictor is trained online and the best item are selected at each round. In order to update the predictor efficiently, we extract the embedded user-item features $\phi(x_{i,j})$ from the neural net of offline training and use a relatively shallow network $f'(\phi(x_{i,j}), \theta)$ to train the online predictor. The performance metric of online recommendation is regret. If item j_t is selected for user i_t at round t, then the regret is expressed as

$$reg_t = \max_j f'(\phi(x_{i_t,j}), \theta^*) - f'(\phi(x_{i_t,j_t}), \theta^*), \tag{2}$$

where θ^* is the optimal neural net parameter.

4 Algorithms

4.1 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is one of the popular algorithms used for collaborative filtering.This type of algorithm finds the features of users and objects, and makes predictions based on these factors[9].

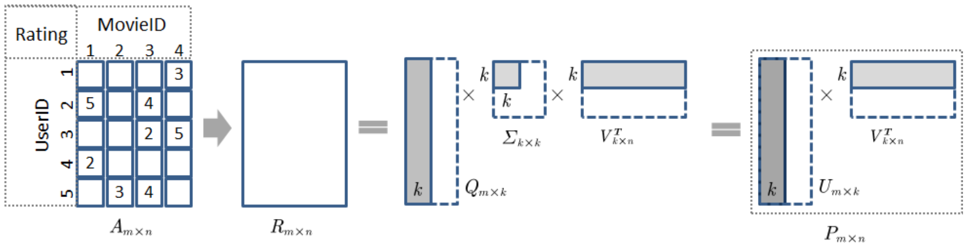


Figure 1: SVD algorithm.

Suppose $A \in \mathbb{R}^{m\times n}$ is the score matrix of m moive ID and n users, shows as Fig. 1.Each row represents one user, and each column represent one movie, score value ranges from 0 to 5. Usually, a user does not score all objects in the database.
Consider the case in which the ratings matrix is fully specified, or initial the missing value to make it a full score matrix R. One can approximately factorize the ratings matrix R by using truncated SVD of rank $k \ll min\{m, n\}$ Truncated SVD is computed as follows[2]:

$$R = Q\Sigma P^T \tag{3}$$

The SVD algorithm finds three matrices. Here, Q, Σ, and P are matrices of size $m \times k$, $k \times k$, and $n \times k$, respectively. The diagonal matrix Σ can be absorbed in either the user factors Q or the item factors P. By convention, the user factors and item factors are defined as follows:

$$U = Q\Sigma \tag{4}$$

$$V = P \tag{5}$$

As before, the factorization of the ratings matrix R is defined as $R = UV^T$. Therefore, the goal of the factorization process is to discover matrices U and V with orthogonal columns. Therefore, SVD can be formulated as the following optimization problem over the matrices U and V :

$$\begin{aligned} &Minimize\ J = \frac{1}{2}\|A - UV^T\|_2 \\ &subject\ to\ : \\ &\qquad Columns\ of\ U\ are\ mutually\ orthogonal \\ &\qquad Columns\ of\ V\ are\ mutually\ orthogonal \end{aligned} \tag{6}$$

4.2 Multi-layer Perceptron (MLP)

Since our input are userId, movieId and items like keywords and gerens, it is intuitive to combine the features by concatenating them. However, simply a vector concatenation does not account for any interactions between these features, which is insuffcient for modelling the collaborative filtering effect. To address this issue, we adapt the methods in paper[4] to add hidden layers on the concatenated vector, using MLP to learn the interaction between features. By doing this, the model will show large level of flexibility and non-linearity to learn the interactions. A simply MLP figures is as Fig. 2. More precisely, the MLP model under our NCF framework is defined as

$$\begin{aligned} z_1 &= \phi_1(P_u, Q_i) = \begin{bmatrix} P_u \\ Q_i \end{bmatrix}, \\ \phi_2(z_1) &= a_2(W_2^T z_1 + b_2), \\ &\cdots\cdots \\ \phi_L(z_{L-1}) &= a_L(W_2^T z_{L-1} + b_L), \\ \hat{y_{ui}} &= \sigma(h^T \phi_L z_{L-1}), \end{aligned} \tag{7}$$

where W_x, b_x, and a_x denote the weight matrix, bias vector, and activation function for the $x-th$ layer's perceptron, respectively. p_u and q_i denote the latent vector for user u and item i, respectively; y_{ui} is our estimation, which is our output.

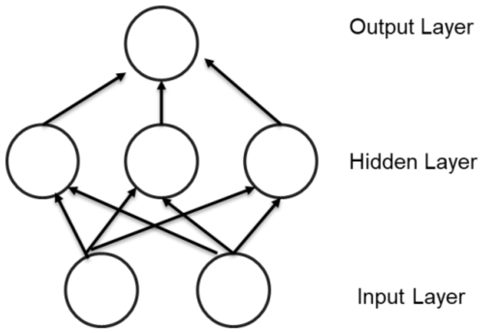


Figure 2: Model of a Simply Network.

4.3 Neural Collaborative Filtering(NCF)

Neural collaborative filtering is using a multi-layer perceptron (MLP) to learn the user-item interaction function. To learn model parameters, existing pointwise methods largely perform a regression with squared loss:

$$L_{sqr} \;=\; \sum w_{ui}\left(y_{ui}-\hat{y}_{ui}\right)^2 \tag{8}$$

Where w_{ui} is a hyper-parameter denoting the weight of training instance (u, i).

To construct a full neural treatment of collaborative filtering, we use multi-layer representation to model a userId, movieId, keywords and genres as shown in Fig. 3, where the output of one layer serves as the input of the next one. The bottom input layer consists of three feature vectors userId, movieId, keywords or genres. Above the input layer is the embedding layer; it is a fully connected layer that projects the sparse representation to a dense vector. After this we concatenate the dense vectors and let them go through 4 hidden layers.

To make the raw data as proper input, we used pandas. First, using lambda function, pandas can transform JSON data into a list object. Second, we removed data that contains empty or null values. The next step is to transform the list into a NumPy array. After many attempts, it turns out that keeping genres and keywords as three-dimension matrix would give the best performance. Now, with userId and movieId transformed into two one-dimensional vectors, genres transformed into a three-dimensional matrix, the data is ready to be used as model inputs.

After pre-processing the input, the general model needs to be modified to achieve regression work. The first layer of the model is the embedding layer. In this layer, each input is embedded into a 32-dimensional vector. Then after concatenation, comes the MLP layer which consists of 4 hidden layers. Each layer has weight regularizer and dropout to prevent overfitting because regression is easier to overfit than classifier. The output layer uses relu activation function rather than sigmoid.

4.4 Neural-UCB

Neural-UCB [15] aims to solve the cold-start problem in online recommendation. The key idea of Neural-UCB is to balance the exploitation and exploration when selecting items based on the prediction of neural network. At round t, for user i_t and item $j = 1, 2, \cdots, K$ with the joint feature $\phi_{i,j}$, the neural network predict the rating as $f(\phi_{i,j}, \hat{\theta}_t)$. Instead of directly selecting an item based on the prediction result, Neural-UCB select the item as

$$j_t = \max_j \left\{ f(\phi_{i,j}, \hat{\theta}_t) + \alpha\sqrt{g^\top(\phi_{i,j}, \hat{\theta}_t)\mathbf{V}_t^{-1}g(\phi_{i,j}, \hat{\theta}_t)} \right\}, \tag{9}$$

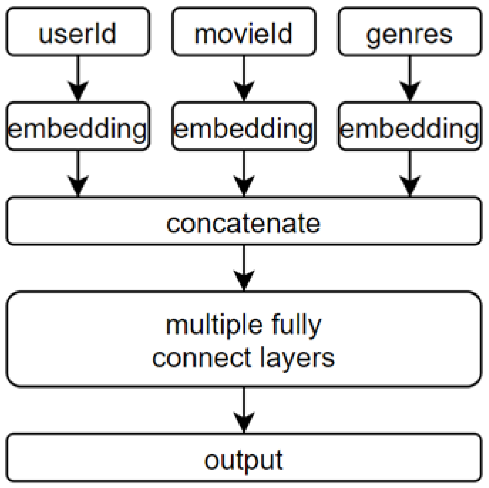


Figure 3: Learning Architecture of Recommendation.

where $g(\phi_{i,j}, \hat{\theta}_t)$ is the gradient of the neural network with input $\phi_{i,j}$ and $\mathbf{V}_t = \lambda\mathbf{I} + \sum_{s=1}^{t-1} g^\top(\phi_{i,j}, \hat{\theta}_t) g(\phi_{i,j}, \hat{\theta}_t)$ is the information matrix. In arm selection (9), $f(\phi_{i,j}, \hat{\theta}_t)$ is the exploitation term, $\sqrt{g^\top(\phi_{i,j}, \hat{\theta}_t)\mathbf{V}_t^{-1} g(\phi_{i,j}, \hat{\theta}_t)}$ is the exploration term and α is a hyper-parameter to balance the two terms.

In our experiments, we extract user-item features from the dense layer with 16 dimensions and use them as the input to the online neural network. The online neural net has only one hidden layer with 8 neurons. Relu function is used as the activation function.

5 Results

SVD The data we use to perform SVD to minimize the RMSE is 'userId', 'movieId', 'rating'. Recommendation is given based on the predicted rating value. After running 5-fold cross validation, we get a mean Root Mean Sqaure Error of 0.8972, see Fig. 4, This result can be use to compare with Neural Collaborative Filtering (NCF).

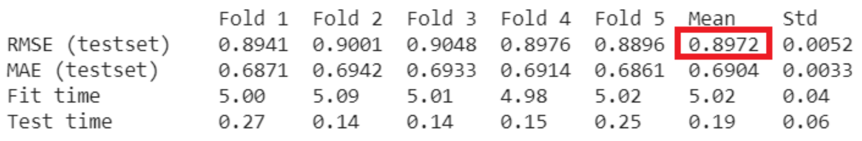

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8941  0.9001  0.9048  0.8976  0.8896  0.8972  0.0052
MAE (testset)     0.6871  0.6942  0.6933  0.6914  0.6861  0.6904  0.0033
Fit time          5.00    5.09    5.01    4.98    5.02    5.02    0.04
Test time         0.27    0.14    0.14    0.15    0.25    0.19    0.06
```

Figure 4: Training result of SVD using 5-fold cross- validation.

NCF Table 1 shows RMSE with respect to SVD and Neural collaborative filter. For the Neural method, we tested RMSE while adding different movie features into the input. And Fig. 5 is the detail records of RMSE in training using different inputs. We only add userId and movieId as our error baseline. Compared with baseline, we add genres and keywords respectively to the baseline items, and we find that the error is been reduced respectively. Besides, we combine genres and keywords with baseline and find that the error is become smaller.

We can see that Neural CF significantly outperform SVD CF. Also, putting more information into inputs, Neural CF can achieve even better performance.

Neural-UCB Assume the user with ID 15 is a new user. We use Neural-UCB to recommend items to this user online. First, we extract the 16-dimension features regarding the user and 1000 items interacting with the user from the offline trained neural network. Then a neural network with a hidden layer with 8 neurons is used for online training. The regret and rating with respect to the

Table 1: RMSE of SVD and Neural CF.

RMSE	baseline	genres	keywords	both
SVD CF	0.8972	NA	NA	NA
Neural CF	0.2468	0.2426	0.2425	0.2399

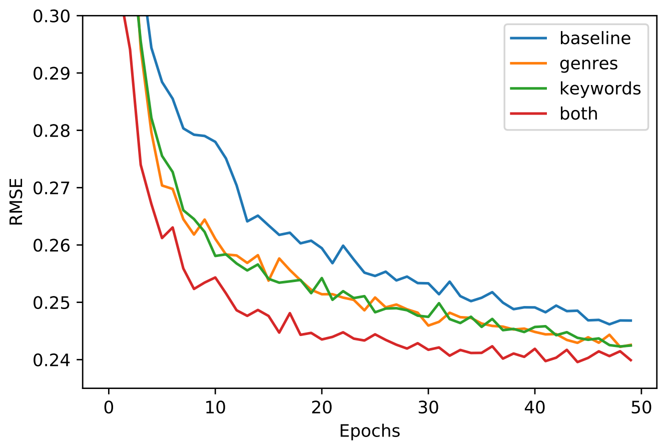
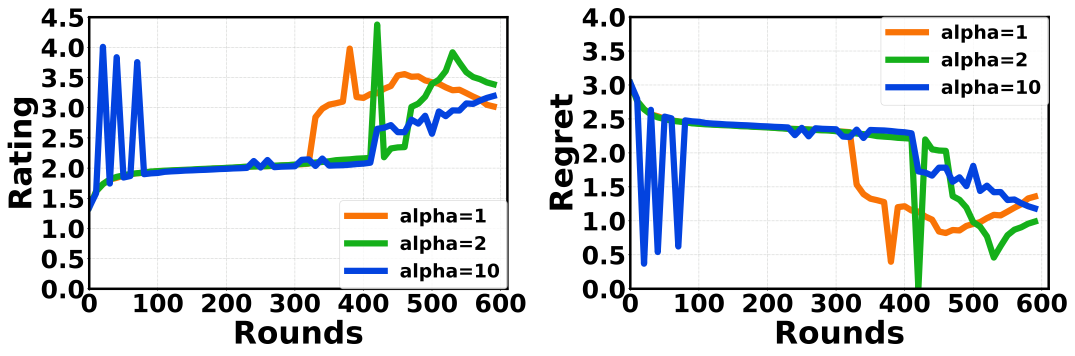


Figure 5: RMSE of Neural CF using different inputs.



(a) Ratings of user 15.

(b) Recommendation regrets of user 15.

Figure 6: Ratings and Recommendation regrets of user 15 with rounds under different exploration rate α.

recommended item at each round is given in Fig. 6. The simulation results show that the rating of user 15 increase and the corresponding regret decreases with rounds though with some fluctuations. Also with larger exploration rate α, the regret decreases slower but is lower for larger rounds. This is because more exploration is done.

6 Conclusion

In this project, we study the user-item interaction learning for recommendation systems based on Movies dataset [1]. We built a neural network with embedding layers to learn the user preference for items. By adding additional features like genres and keywords in neural collaborative filtering, we reduce the rating prediction RMSE compared to basic SVD and neural collaborative filtering. Based on the extracted user-item feature by offline training, we employed Neural-UCB into online recommendation for a new user. The simulation results show that the recommendation regret increases with round and user rating is improved.

References

[1] The movies dataset. `https://www.kaggle.com/rounakbanik/the-movies-dataset`.

[2] C. C. Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[3] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[5] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, pages 556–562, 2001.

[6] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[7] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th international conference on World wide web*, 2010.

[8] H. Liu, J. Wen, L. Jing, and J. Yu. Deep generative ranking for personalized recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 34–42, 2019.

[9] C.-C. Ma. A guide to singular value decomposition for collaborative filtering. *Computer (Long Beach, CA)*, 2008:1–14, 2008.

[10] B. Song, X. Yang, Y. Cao, and C. Xu. Neural collaborative ranking. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1353–1362, 2018.

[11] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009, 2009.

[12] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini. Finite-time analysis of kernelised contextual bandits. *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, 2013.

[13] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*, pages 2000–2010, 2019.

[14] Q. Wang, H. Yin, H. Wang, Q. V. H. Nguyen, Z. Huang, and L. Cui. Enhancing collaborative filtering with generative augmentation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 548–556, 2019.

[15] D. Zhou, L. Li, and Q. Gu. Neural contextual bandits with upper confidence bound-based exploration. *arXiv preprint arXiv:1911.04462*, 2019.