Improving DMF with Hybrid Loss Function and Applying CF-NADE to the MOOC Recommendation System

Thanh Le
lnthanh@fit.hcmus.edu.com
Faculty of Information Technology
University of Science, VNU-HCM
Ho Chi Minh City, Vietnam

Vinh Vo
1612815@student.hcmus.edu.vn
Faculty of Information Technology
University of Science, VNU-HCM
Ho Chi Minh City, Vietnam

Khai Nguyen
1612909@student.hcmus.edu.com
Faculty of Information Technology
University of Science, VNU-HCM
Ho Chi Minh City, Vietnam

Bac Le
lhbac@fit.hcmus.edu.com
Faculty of Information Technology
University of Science, VNU-HCM
Ho Chi Minh City, Vietnam

***Abstract*—Nowadays, with the strong development of platforms like Coursera, Edx, etc., Massive Open Online Course (MOOC) is not too strange for most people. The number of online courses also increases day by day. One of the problems raised is how to recommend users to choose the appropriate course. To address the problem, we applied the Deep Matrix Factorization (DMF) model to build a user-item interaction matrix with explicit rating and zero implicit feedback. We then improved the loss function to yield more accurate results. In addition, we also used the Collaborative Filtering Neural Autoregressive Distribution Estimator (CF-NADE) model to MOOC Recommendation system. Our experiment shows that two proposed approaches achieve better results than the other methods.**

Keywords: MOOCs; Recommendation; CF-NADE; DMF.

I. INTRODUCTION

The recommendation system has gone through three decades of development across many fields, and today the most successful models are using deep learning because it provides high accuracy. In the Fourth Industrial Revolution, online learning became an indispensable need for learners who yearn for knowledge, so MOOC platforms, such as Coursera [36] Edx [37] Khan Academy [38] etc. contain thousands of courses and millions of learners. The rapid increase in the number of courses on MOOCs poses a problem of how to choose the right course for learners themselves.

According to our survey, there is not much current research on MOOCs recommendation. These lead to the complexity of assessing user knowledge. A good course recommendation system will help learners not have to spend time trying out courses or reading feedback from other learners to find a suitable course. If a learner has to take 5 to 6 pre-trial courses and read hundreds of feedbacks from other learners before finding a suitable course for themself, they will be discouraged from looking for more courses. Jdidou et al. [11][12] showed that the use of the course recommendation system will optimize learners' profitability. Yanhui et al. [13] showed that 87.3% of the respondents rated highly in the effectiveness of a course recommendation system. Course recommendation systems have also been developed to increase the completion rates of learners [13] [14].

Data from MOOCs is usually the course name, content, as well as learner information, etc. One person can study many different courses. From there, we can simulate in the form of knowledge graph in which users, courses, etc. will be the vertices of the graph and the edge will be a relationship such as learning, voting, etc. This graph will get bigger and more complex as more courses are created and the number of learners increases. Recommending a suitable course for a specific user will be complicated both in time and accuracy.

Because the data can be represented in the form of knowledge graphs, we conducted a survey of approaches in knowledge graph mining and combined them with collaborative filtering methods in the recommendation system. In the results, we chose two models, improved and evaluated them on the Travel-well dataset [24] which is the MOOC data in Learning Resource Exchange (LRE) portal in Europe. Specifically, we used the DMF model and [16] CF-NADE model [10]. The DMF model uses the Matrix Factorization and the Deep Structured Semantic to construct a user-item matrix with explicit ratings and non-preference implicit feedback. The loss function improved to achieve better results than conventional loss functions. The CF-NADE model is used to predict the probability of rating and use the rating to recommend items to users.

The remainder of this paper is structured as follows. Section II describes the related works used in the recommendation. Section III, we present the theoretical principle of CF-NADE and DMF model and our improvements. In section IV, we conduct experiments on the Travel-well dataset. Section V is the conclusion and our future works.

II. RELATED WORKS

In recent years, deep learning has been widely applied in many different fields. Applying deep learning into the recommendation system also brings many good results [15] Salakhutdinov et al. [17] proposed the Restricted Boltzmann Machine (RBM) model with only one hidden layer, which is the first deep learning model used in the recommendation system. Zheng et al. [10] proposed a CF-NADE model based on the Neural Autoregressive Distribution Estimator (NADE) model and Restricted Boltzmann Machine for Collaborative Filtering (RBM-CF) model [17], which showed better results than RBM-CF in Netflix and Movielens dataset. Sedhain et al. [19] proposed an AutoRec model based on an autoencoder network to fill the missing rating from the input layer. These above models only use rating explicit feedback, and do not solve the cold-start problem. Wang et al. [20] presented a new model called Collaborative Deep Learning (CDL) using hierarchical Bayesian and Stack Denoising Autoencoders (SDAE) to solve the sparsity problem of data. Wu et al. [22] proposed a Collaborative Denoising Autoencoder (CDAE) model improved from the Denoising AutoEncoder (DAE) model. Pan et al. [23] improved the CDAE model by using three small CDAE models combined to form a new model called Correlation Denoising Autoencoder (CoDAE). Elkahky et al. [18] proposed the Multi-View Deep Neural Network (MV-DNN) model by combining multiple Deep Structured Semantic Model (DSSM) models to take advantage of common information between different regions. Later, Xue et al. [16] proposed the DMF method by combining matrix factorization and DSSM model, while improving the loss function to achieve better effectiveness and having less running time.

In the MOOCs recommender system, there are many research works apply deep learning methods. Yang et al. [2] use matrix factorization and context information forum apply on MOOC Forum thread. Kardan et al. [25] use social network analysis and association rule mining for MOOC forums. Raghuveer et al. [5] propose a reinforcement learning model to generate learning context and analyze learner information. Mi et al. [26] use the context tree applied to an online sequential recommendation. Pardos [6] uses Recurrent Neural Networks to handle learner's time on each page for predicted courses. Jing et al. [1] construct a content-awareness framework using users' access information to represent learners' interests and behavior features. Zhang et al. [3] used a deep belief network for the first time in MOOC recommendation. Then, Zhang et al. [35] improved a better accurate recommendation model using learner's information features, course content features, and learner-course feature vectors as inputs. Zhang et al. [4] also improved the Deep Belief Network (DBN) model and analyzed learner's style features. Pardos et al. [7] propose a Multifactor2vec model to improve the semantics of token embedding. While effective, these methods need data that has a lot of information about courses and very high computation. In 2020, Troussas et al. [39] has proposed a recommendation system that performs a learning analysis of a given user to suggest relevant courses for that user. In addition, this system can also predict user behavior, meaning that it predicts courses that users continue to study in the future; the system has received very positive feedback. In terms of predicting the course, 62.5% of the participants thought that the system suggested exactly the courses they wanted to study and only 9.375% of the participants surveyed thought that the system was unsuccessful. Regarding the prediction of user behavior, 68.75% of participants thought that it was very accurate and only 6.25% thought that this system predicted incorrectly.

Among the above methods, we found that Xue et al.'s DMF model [16] and Zheng's CF-NADE model [10] are suitable for the MOOC recommendation system. Therefore, in the next section, we present the basic theoretical principles of these methods, and improvements to increase the accuracy of the system.

III. MOOC Recommendation System

Initially, user and course data will be stored as a matrix. To turn this space into an appropriate form for later problems, we often use methods in graph embeddings such as factorization approaches, random walk approaches, and deep approaches. Among them, the deep approaches are being used quite a lot. In section A, we use the matrix factorization using deep learning because of its effectiveness in our problem. Section B will cover the foundation of the CF-NADE model.

A. Deep Matrix Factorization Model

DMF is a technique that combines the Matrix Factorization technique (MF) and DSSM.

DSSM is proposed by Huang et al. [28] used in web search. DSSM uses a deep learning model to rank documents for a query. In the beginning, DSSM maps query and documents into lower semantic space with a multi-layer non-linear projection. Then, for ranking web search, cosine similarity between the query vector and a document vector is applied.

Given a set include M users: $\mathbf{U=\{u_1,u_2,\dots,u_M\}}$, and a set include N items: $\mathbf{I=\{i_1,i_2,\dots,i_N\}}$. $\mathbf{R}\in\mathbb{R}^{\mathbf{MxN}}$is the rating interaction matrix with $\mathbf{R_{ij}}$ is the rating of user **i** for item **j**, **unk** is unknown rating. Equation (1) presents user-item interaction matrix.

$$Y_{ij}=\begin{cases}0,\text{if } R_{ij}=\text{unk}\\ R_{ij},\text{ otherwise}\end{cases} \tag{1}$$

where **u** is user, **v** is item; **i**, **j** is index of **u**, **v**. **Y** is user-item interaction matrix, $\mathbf{Y^+}$is observed interactions, $\mathbf{Y^-}$ is zero elements in **Y**, $\mathbf{Y^-_{sampled}}$ is a set of negative instances from **Y** (in part or in whole). Then $\mathbf{Y^-\cup Y_{sampled}}$ is a set of training interactions. Row **i** of matrix **Y** is $\mathbf{Y_{i*}}$, column **j** of the matrix is $\mathbf{Y_{*j}}$.

1) Deep matrix factorization model (DMF)

DMF is proposed by Xue et al. [16] and used in the recommender system. This model uses explicit rating and zero implicit feedback to predict items based on rated items that users. The DMF model has input is an interaction matrix, like DSSM, this matrix is split into two multi-layer perceptrons (MLPs). Then, the output of these MLPs is latent representations. Finally, for calculating the correlation between two latent representations, we use cosine similarity. Figure 1 illustrates DMF model.

Suppose the input is vector **x**, the output is vector **y**, $\mathbf{l_i}$ is ith hidden layer, $\mathbf{W_i}$ is ith weight matrix, $\mathbf{b_i}$ is ith biased. From

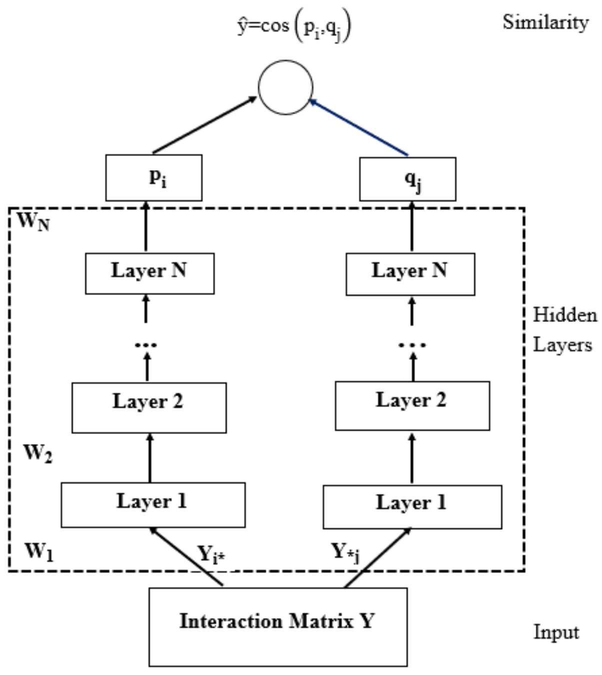

Figure 1. DMF architecture

interaction matrix **Y**, each user $\mathbf{u_i}$ is a vector of $\mathbf{Y_{i*}}$meaning i^{th} user rates for all items. Each item $\mathbf{v_j}$ is a vector $\mathbf{Y_{*j}}$, meaning j^{th} item rated by all users. MLPs use (2).

$$l_1=W_1x$$
$$l_i = f(W_{i-1}l_{i-1}+b_i),i=2,\ldots N-1 \qquad (2)$$
$$y = f(W_N l_{N-1}+b_N)$$

We use ReLU activation function in (3).

$$f(x)=\max(0,x) \qquad (3)$$

In other words, this model has two MLPs, one for users and one for items, and outputs are mapped into low dimensional vectors in latent space in (4).

$$p_i= f_{\theta_N^U}\left(\ldots f_{\theta_3^U}\left(W_{U2}f_{\theta_2^U}(Y_{i*}W_{U1})\right)\ldots\right)$$

$$q_j= f_{\theta_N^I}\left(\ldots f_{\theta_3^I}\left(W_{V2}f_{\theta_2^I}\left(Y_{*j}^T W_{V1}\right)\right)\ldots\right) \qquad (4)$$

Then, we calculate the cosine similarity of two latent representations p_i and q_j with (5).

$$\hat{Y}_{ij} =F^{DMF}(\theta)=\text{cosine}\left(p_i,q_j\right)=\frac{p_i^T q_j}{\|p_i\|.\|q_j\|} \qquad (5)$$

This is the first model that uses direct input as an interaction matrix and very useful in representing the final low dimension.

In the next section, we will represent an improved loss function, which increases the accuracy of the model.

2) Loss function

The general objective function in (6).

$$L=\sum_{u\in Y^+\cup Y^-} l(y,\hat{y})+\lambda\Omega(\theta) \qquad (6)$$

where $\Omega(\theta)$ is regularizer and l(.) is loss function.

The loss function is an important part of the objective function. The better the loss function, the better the objective function. We optimize the objective function.

Some papers using Binary cross-entropy were proposed (7) [30]

$$L_{BCE} = -\sum_{(i,j)\in Y^+\cup Y^-} Y_{ij}\log\hat{Y}_{ij}+\left(1-Y_{ij}\right)\log\left(1-\hat{Y}_{ij}\right) \qquad (7)$$

This function works effectively with implicit feedback because it addresses implicit feedback classification as binary classification.

Because we use both zero implicit feedback and explicit rating so that we use a new loss function that combines (7) and max rating. It is called Normalized cross-entropy loss in (8).

$$L_{NCE} = -\sum_{(i,j)\in Y^+\cup Y^-} \left(\frac{Y_{ij}}{\max(R)}\log\hat{Y}_{ij} + (1-\frac{Y_{ij}}{\max(R)})\log\left(1-\hat{Y}_{ij}\right)\right) \qquad (8)$$

where max(R) is max(Rating). In our data, we use max(R)=5 because 5 is the max rating.

Normalized cross-entropy can make Y_{ij} negative so that we use (9) to solve this problem.

$$\hat{Y}_{ij}^O = \max\left(\mu,\hat{Y}_{ij}\right) \qquad (9)$$

where $\mu=10^{-6}$.

L2 loss function fits in solving the overfitting problem and is used in many papers [33] [32]. Equation (10).

$$L2 = \frac{\sum_i^m w_i^2}{2} \qquad (10)$$

where $w_i^2=\sum_j^N w_{ij}^2$, and w_{ij} is the weight of the training instance (i,j); N is the dimension of w_{ij}.

We improve loss function by combining (8) and (10) and call it is Hybrid loss (11)

$$L_{Hybrid} = L_{NCE} + \alpha.L2 \qquad (11)$$

This loss function that we improve not only fit rating data but it also avoids overfitting. When we use this loss function, we gain better results in experiments. In our experiment, we give $\alpha = 10^{-3}$. We represent the DMF with Hybrid loss function in Figure 2 (named Hybrid-DMF).

```
Algorithm 1: Hybrid-DMF (Iter, neg-ratio, R)
Inputs:
    Iter # The number of iterations
    neg-ratio #negative ratio
    R # Interaction matrix
Outputs:
    $W_{Ui}$(i=1…N-1)# weight matrix for user
    $W_{Vi}$(i=1…N-1)# weight matrix for item
1. Initialize:
    a. Initialize randomly $W_U$ and $W_V$
    b. Y := Use (3)
    c. $Y^+$:= All non-zero interactions in Y;
    d. $Y^-$ :=All zero interactions in Y;
    e. $Y^-_{sampled}$ := sample neg _ ratio* $||Y^+||$
       (interactions from $Y^-$)
    f. T:=$Y^+ \cup Y^-_{sampled}$
2. Loop it from 1 to Iter:
       Loop each interaction of user i and item j in T:
           $p_i$,$q_j$ := Use (4)
           $\hat{Y}^O_{ij}$ := Use (5) and (9)
           L := Use (11).
       End for
   End for
```

Figure 3. Hybrid-DMF algorithm

B. Collaborative Filtering Neural Autoregressive Distribution Estimator Model (CF-NADE)

Besides DMF, we found that the CF-NADE model is also suitable for the MOOC recommendation system.

Assuming there are M courses and N users, the user ratings are from 1 to K. We assume each user rated D courses and D≪M. With any user u, we will have the rating vector $\mathbf{r^u}=\left(r^u_{m_{o_1}}, r^u_{m_{o_2}},..., r^u_{m_{o_D}}\right)$, where o is the permutation of (1, 2, …, D), $\mathbf{r^u_{m_{o_i}}} \in \{1,2,…,K\}$ is present for the rating of user u and item $\mathbf{m_{o_i}}$. Section 2 will present the basic CF-NADE model.

1) Basic CF-NADE model

The CF-NADE model was proposed by Zheng et al. [10] developed from the NADE model, and can be used in the recommendation system. Basic CF-NADE has 3 layers (input layer, hidden layer, and output layer), and the basic CF-NADE can be extended to have one more hidden layer and become a deep neural network. Equation (13) shows the probability of the rating vector r. Where $\mathbf{r_{m_{o<i}}}=\left(r_{m_{o_1}},r_{m_{o_2}},...,r_{m_{o_{i-1}}}\right)$ is the first i - 1 elements by index **o** of **r**.

$$p(r)=\prod_{i=1}^{D} p\left(r_{m_{o_i}}\middle|r_{m_{o<i}}\right) \tag{12}$$

Like NADE, CF-NADE models (12) with an autoencoder. Firstly, the CF-NADE model computes the hidden presentation in a hidden layer (13).

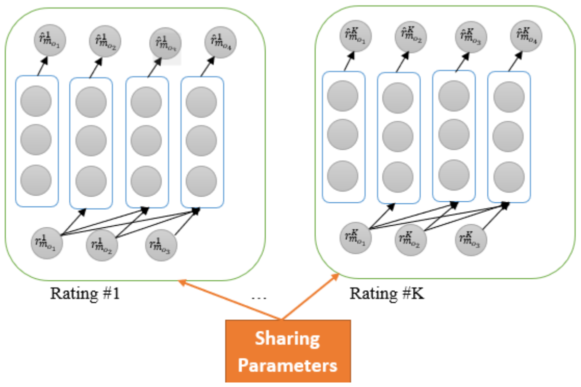

Figure 2. CF-NADE with sharing parameters

$$h\left(r_{m_{o<i}}\right)=g\left(c+\sum_{j<i}\mathbf{W}^{r_{m_{o_j}}}_{:,\,m_{o_j}}\right) \tag{13}$$

With g(.) is an activation function, such as tanh function. $\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$, $W^k \in \mathbb{R}^{H\times M}$ is a connective matrix between input layer and hidden layer. We can rewrite (13) as (14).

$$p\left(r_{m_{o_i}}=k\middle|r_{m_{o<i}}\right)=\frac{\exp\left(s^k_{m_{o_i}}\left(r_{m_{o_i}}\right)\right)}{\sum_{k'=1}^{K}\exp\left(s^{k'}_{m_{o_i}}\left(r_{m_{o<i}}\right)\right)} \tag{14}$$

where $\mathbf{s^k_{m_{o_i}}}\left(\mathbf{r_{m_{o<i}}}\right)$ is the rating of course $\mathbf{m_{o_i}}$ by user **k** when knew the previous ratings $\mathbf{r_{m_{o<i}}}$, has computed by (15)

$$s^k_{m_{o_i}}\left(r_{m_{o<i}}\right)=b^k_{m_{o_i}}+V^k_{m_{o_j},:}h\left(r_{m_{o<i}}\right) \tag{15}$$

where V is connection matrix and b is bias with rating **k.**

2) CF-NADE model with sharing parameters.

Because if we use the basic CF-NADE model, we will have many models that depend on the number of users and rating range. Specifically, if we have **N** users and **K** ratings from 1 to K, we must use **N**x**K** basic CF-NADE model. Figure 3 shows how the CF-NADE model shared parameters. However, in reality, with each user, the number of items that user rates is very small, making the model difficult to optimize, so it is necessary to share parameters between the models, use only one model for N users and K ratings So that, (13) and (15) will change to (16) and (17) respectively.

$$h\left(r_{m_{o<i}}\right)=g\left(c+\sum_{j<i}\sum_{k=1}^{r_{m_{o_j}}}W^k_{:,\,m_{o_j}}\right) \tag{16}$$

$$s^k_{m_{o_i}}\left(r_{m_{o<i}}\right)=\sum_{j\le k}\left(b^j_{m_{o_i}}+V^j_{m_{o_i},:}h\left(r_{m_{o<i}}\right)\right) \tag{17}$$

where V^j, b^j is shared by k ratings.

3) Training model with ratings' ordinal information.

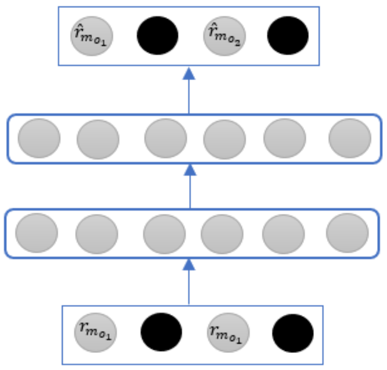

Figure 4. CF-NADE deep model

The model can be trained by minimizing the negative log-likelihood function (18).

$$-\log p(r)=-\sum_{i=1}^{D}\log p\left(r_{m_{o_i}}\middle|r_{m_{o_{<i}}}\right) \tag{18}$$

The current cost function treats different ratings as different labels. The labels with the highest probability will be the true label and not capture the ordinal information. Zheng et al. improved the cost function by adding label ordering information. When the realistic rating is k, this improvement makes the probability from 1 to k ascending and from k to K descending. So that it helps predicted ratings more accuracy. Assume that user rates kth, then the rating from 1 to k has priority increase, and the value from k to K has priority decrease. Equation (19) is the improved function to compute the conditional probability of $r_{m_{o_i}}=k$, with given previous ratings $\mathbf{r}_{\mathbf{m}_{\mathbf{o}_{<i}}}$.

$$p\left(r_{m_{o_i}}=k\middle|r_{m_{o_{<i}}}\right)=\prod_{j=k}^{1}\frac{\exp\left(s_{m_{o_i}}^{j}\right)}{\sum_{t=1}^{j}\exp\left(s_{m_{o_i}}^{t}\right)}\prod_{j=k}^{K}\frac{\exp\left(s_{m_{o_i}}^{j}\right)}{\sum_{t=j}^{K}\exp\left(s_{m_{o_i}}^{t}\right)} \tag{19}$$

Cost function will be change to (20):

$$C_{hybrid}=(1-\lambda)C_{reg}+\lambda C_{ord} \tag{20}$$

where C_{ord} is an ordinal cost (19) and C_{reg} is a regular cost or negative log-likelihood (18) and λ is a hyper-parameter to determine the weight of C_{ord}.

4) Extend CF-NADE to a deep neural network.

According to Zheng et al., the CF-NADE model can have one additional hidden layer. Figure 4 describes CF-NADE when adding one hidden layer. When added a hidden layer to the model, computational complexity will increase, and the calculation formula of that layer is as (21).

$$h^{(l)}\left(r_{m_{o_{<i}}}\right)=g\left(c^{(l)}+W^{(l)}h^{(l-1)}\left(r_{m_{o_{<i}}}\right)\right) \tag{21}$$

where $l = 2, \ldots, L$ correspond to the hidden layers and the conditional probability $p\left(r_{m_{o_{<i}}}\right)$ is computed based on $h^{(L)}\left(r_{m_{o_{<i}}}\right)$. The computational complexity is $O(K\widehat{D}H+H^2L)$. Where $\widehat{\mathbf{D}}$ is the average of the number of courses that user rated, **H** is the nodes of the hidden layer. The Cost function is optimized using nonlinear optimization methods such as gradient descent.

Two suggested models above have effective results. Therefore, we will apply these models to the MOOC recommendation system. For comparison, in section IV, experience and results, we will evaluate two models with others to indicate the outperformance of two models compared to other methods.

IV. EXPERIENCE AND RESULTS

In this section, we will compare our improve models with other models with the MOOC dataset. Then, we show the results and comments.

A. Dataset

We use only the Travel-well dataset in [24] for our experiment because some other datasets are not fit for our model or not public. The Travel-well dataset was collected from the LRE portal includes 20 content providers. It contains information about the ratings and tagging behaviors of 98 learners in over six months (August 2008-February 2009), but there are only 75 learners rated for courses. In our experiment, we only use rating information.

TABLE I. TRAVEL-WELL DATASET

#learners (#users)	**#courses (#items)**	**#ratings**	**density**
75	1608	2156	0.0178

B. Parameter Settings

1) Parameter settings for Hybrid-DMF

We run the experiment with the following configurations: Python = 3.7.6, libraries such as Tensorflow-gpu = 1.5.0, numpy = 2.1.0.

Hyperparameter settings: learning rates = 10^{-4}, max epoch = 30 because our improved model converges less than 30 epochs, batch size = 256, early stopping = 5, the latent factor = 64 because with the number of latent factor < 64 we get wrong result and when the number of latent factor > 64, the result < when the number of latent factor = 64. The dimension of input user = 1024, and the item =512. The hyperparameters are tested from multiple hyperparameters and then choose the most optimal hyper-parameters.

2) Parameter settings for CF-NADE

The code runs with requirements: python 3.6.8. Dependence packages: Tensorflow (2.1.0), Tensorflow-gpu (2.1.0), Matplotlib, Keras (2.0.8), Pyspark (2.4.1).

Hyper-parameter settings: Learning rate = 10^{-3}, Hidden unit = 500, epochs = 20. The hyper-parameters are tested from multiple hyper-parameters and then choose the most optimal hyper-parameters. Specifically, the learning rate is chosen from 10^{-3}, $5*10^{-4}$ and 10^{-4}, epoch is chosen from 20 and 50, hidden units are 500 and 1000.

C. Metrics

To evaluate performance, we adopted the *leave-one-out* evaluation. We use two metrics: *Normalized Discounted Cumulative Gain* (NDCG) in [31] to evaluate the ranking performance of the relevance courses. NDCG is a metric, which assigns the results at top ranks, scoring successively lower ranks (22) and *Root mean square error* (RMSE) (23).

$$NDCG@K = Z_K \sum_{k=1}^{K} \frac{2^{r_i}-1}{\log_2(i+1)} \quad (22)$$

where Z_K is the ideal ranking and has a value of 1; r_i is the graded relevance of item at position i. We use the simple binary relevance for our work: r_i=1 if the item is in the test set, and 0 otherwise.

$$RMSE = \sqrt{\frac{\sum_{i,j}^{M,N}(r_{ij}-\hat{r}_{ij})}{\#ratings}} \quad (23)$$

For the NDCG metrics, larger values indicate better performance and RMSE, smaller values indicate better performance.

D. Results

To validate the effectiveness of 2 above models, we have selected five algorithms for evaluations (3 classical algorithms and 2 deep learning models): neighborhood-based collaborative filtering methods on item-based (IBCF) [34], neighborhood-based collaborative filtering methods on user-based (Pearson correlation) [34], single value decomposition (SVD) [9], Probabilistic Matrix Factorization (PMF) [8], AutoEncoder based on Collaborative Filtering (AutoRec) [19].

Because SVD, item-based CF and user-based CF are classical algorithms and do not use nonlinear optimization methods like neural networks. Therefore, we only compare RMSE of PMF, CF-NADE and Hybrid-DMF by epoch.

Table II shows RMSE on the test set between comparison methods. As we can see, two models we used have much better results than other methods. Hybrid-DMF has achieved the best RMSE (0.7916), and then CF-NADE (0.8283) for the second.

TABLE II. COMPARISON RMSE OF HYBRID-DMF, CF-NADE, AND OTHER MODELS ON TRAVEL-WELL DATASET

Models and Algorithms	RMSE
AutoRec	2.50037
IBCF	1.3366
UBCF	1.1016
SVD	0.9063
PMF	0.8651
CF-NADE	**0.8283**
Hybrid-DMF	**0.7916**

TABLE III. NDCG@K WITH DIFFERENT TOPKS

TopK	Hybrid-DMF	CF-NADE	AutoRec	SVD
1	0.3467	0.3554	0.0019	0.4927
5	0.3945	0.5505	0.0039	0.4875
10	0.4701	0.6606	0.0040	0.4833
20	0.5000	0.7694	0.0043	0.4800
30	0.5493	0.8225	0.0059	0.4789
50	0.5762	0.8665	0.0070	0.4770

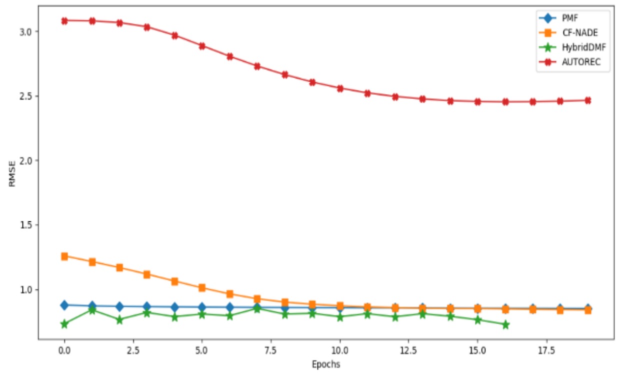

Figure 5. Comparison with RMSE of four models on Travel-Well dataset.

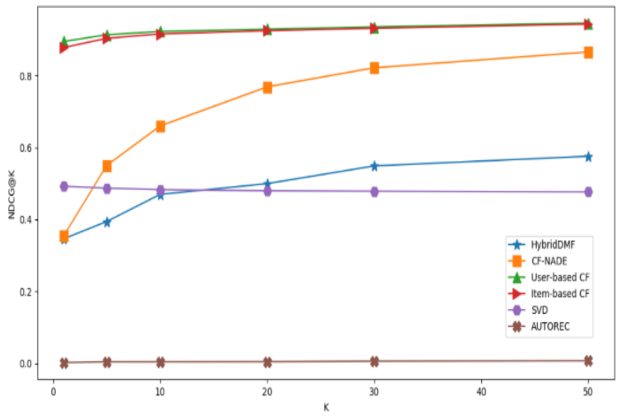

Figure 6. Comparison with NDCG@K on Travel-Well dataset.

Figure 5 shows the RMSE on the test set between PMF, CF-NADE and Hybrid-DMF, AutoRec by epoch. DMF has achieved the best RMSE. CF-NADE initially had a better RMSE than the PMF. However, later the CF-NADE has converged resulting in a better RMSE than the PMF. Finally, AutoRec has the highest RMSE error.

Figure 6 shows the NDCG@K metrics with K = {1, 5, 10, 20, 30, 50} of DMF, CF-NADE, UBCF, IBCF, AutoRec and SVD models. Table III is the detailed result with NDCG@K metric, respectively. In general, user-based CF and item-based CF algorithms have a better NDCG than deep learning algorithms but predict course rating is much worse than deep learning. One of the reasons for deep learning algorithms shown with NDCG measurements is not high is that data is not sufficient.

V. CONCLUSION AND FUTURE WORKS

In this paper, we improved the DMF model with a new loss function (Hybrid-DMF) and combined with the CF-NADE model for the MOOC recommendation system. We evaluated the models on MOOC dataset in Europe. The results showed that the proposed approach is better than the other models with RMSE and NDCG@K measurements when evaluated on the Travel-well dataset. In the future, we will continue to improve DMF with some other loss functions. In addition, this model can be expanded from zero of implicit feedback to implicit feedback containing user feedback such as the click information. This model can also be improved by adding side information from users and items to get better accuracy. Improving CF-NADE can be done by implicit feedback information, such as user tagging for each course to improve the accuracy of the model.

ACKNOWLEDGMENT

This research is funded by University of Science, VNU-HCM, under grant number CNTT 2020-05.

REFERENCES

[1] Jing, X. and Tang, J., 2017, August, “Guess you like: course recommendation in MOOCs,” Proceedings of the International Conference on Web Intelligence (pp. 783-789).

[2] D. Yang, M. Piergallini and P. Rosé, “Forum Thread Recommendation for Massive Open Online Courses,” Education Data Mining, 2014.

[3] Zhang, H., Yang, H., Huang, T. and Zhan, G., 2017, June, “DBNCF: Personalized courses recommendation system based on DBN in MOOC environment,” 2017 International Symposium on Educational Technology (ISET) (pp. 106-108). IEEE.

[4] Zhang, H., et al, “A learning style classification approach based on deep belief network for large-scale online education,” Journal of Cloud Computing 9 (2020): 1-17.

[5] R. Raghuveer, B. K. Tripathy, T. Singh and S. Khanna, “Reinforcement learning approach towards effective content recommendation in MOOC environments,” 2014 IEEE International Conference on MOOC, Innovation and Technology in Education (MITE), Patiala, 2014, pp. 285-289, doi: 10.1109/MITE.2014.7020289.

[6] Pardos, Z.A., Fan, Z. and Jiang, W, “Connectionist recommendation in the wild: on the utility and scrutability of neural networks for personalized course guidance,” User Model User-Adap Inter 29, pp. 487–525, 2019, doi:10.1007/s11257-019-09218-7.

[7] Pardos, Z.A. and Jiang, W., 2020, March, “Designing for serendipity in a university course recommendation system,” Proceedings of the Tenth International Conference on Learning Analytics & Knowledge (pp. 350-359).

[8] Mnih, A. and Salakhutdinov, R.R., 2008, “Probabilistic matrix factorization,” Advances in neural information processing systems (pp. 1257-1264).

[9] Zhang, S., et al, “Using singular value decomposition approximation for collaborative filtering,” Seventh IEEE International Conference on E-Commerce Technology (CEC'05), IEEE, 2005.

[10] Zheng, Y., Tang, B., Ding, W. and Zhou, H., 2016, “A neural autoregressive approach to collaborative filtering,” arXiv preprint arXiv:1605.09477.

[11] Jdidou, Y. and Khaldi, M., 2016, “Increasing the Profitability of Students in MOOCs using Recommendation System,” International Journal of Knowledge Society Research (IJKSR), 7(4), pp.75-85.

[12] Jdidou, Y. and Khaldi, M., 2018, “Using Recommendation Systems in MOOC: An Innovation in Education That Increases the Profitability of Students,” In Enhancing Knowledge Discovery and Innovation in the Digital Era (pp. 176-190), IGI Global.

[13] Yanhui, D., Dequan, W., Yongxin, Z. and Lin, L., 2015, November, “A group recommender system for online course study,” 2015 7th International Conference on Information Technology in Medicine and Education (ITME) (pp. 318-320). IEEE.

[14] Labarthe, H., Bachelet, R., Bouchet, F. and Yacef, K., 2016, “Increasing MOOC completion rates through social interactions: a recommendation system,” Research Track, p.471.

[15] Zhang, S., Yao, L., Sun, A. and Tay, Y., 2019, “Deep learning based recommender system: A survey and new perspectives,” ACM Computing Surveys (CSUR), 52(1), pp.1-38.

[16] Xue, H.J., Dai, X., Zhang, J., Huang, S. and Chen, J., 2017, August, “Deep Matrix Factorization Models for Recommender Systems,” In IJCAI (Vol. 17, pp. 3203-3209).

[17] Salakhutdinov, R., Mnih, A. and Hinton, G., 2007, June, “Restricted Boltzmann machines for collaborative filtering,” Proceedings of the 24th international conference on Machine learning (pp. 791-798).

[18] Elkahky, A.M., Song, Y. and He, X., 2015, May, “A multi-view deep learning approach for cross domain user modeling in recommendation systems,” In Proceedings of the 24th International Conference on World Wide Web (pp. 278-288).

[19] Sedhain, S., Menon, A.K., Sanner, S. and Xie, L., 2015, May, “Autorec: Autoencoders meet collaborative filtering,” In Proceedings of the 24th international conference on World Wide Web (pp. 111-112).

[20] Wang, H., Wang, N. and Yeung, D.Y., 2015, August, “Collaborative deep learning for recommender systems,” Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1235-1244).

[21] Covington, P., Adams, J. and Sargin, E., 2016, September, “Deep neural networks for youtube recommendations,” Proceedings of the 10th ACM conference on recommender systems (pp. 191-198).

[22] Wu, Y., DuBois, C., Zheng, A.X. and Ester, M., 2016, February, “Collaborative denoising auto-encoders for top-n recommender systems,” Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (pp. 153-162).

[23] Pan, Y., He, F. and Yu, H, “A correlative denoising autoencoder to model social influence for top-N recommender system,” Frontiers of Computer Science volume 14, 2020.

[24] Verbert, K., et al, “Dataset-driven research for improving recommender systems for learning,” Proceedings of the 1st International Conference on Learning Analytics and Knowledge. 2011.

[25] Kardan, A., Narimani, A. and Ataiefard, F., 2017, “A hybrid approach for thread recommendation in MOOC forums,” International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering, 11(10), pp.2195-2201.

[26] Mi, F. and Faltings, B., 2016, July, “Adaptive Sequential Recommendation Using Context Trees,” IJCAI (pp. 4018-4019).

[27] Deshpande, M. and Karypis, G., 2004, “Item-based top-n recommendation algorithms,” ACM Transactions on Information Systems (TOIS), 22(1), pp.143-177.

[28] Huang, P. S., et al, “Learning deep structured semantic models for web search using clickthrough data,” Proceedings of the 22nd ACM international conference on Information & Knowledge Management, 2013.

[29] He, X., Zhang, H., Kan, M.Y. and Chua, T.S., 2016, July, “Fast matrix factorization for online recommendation with implicit feedback,” Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (pp. 549-558).

[30] He, X., Liao, L., Zhang, H., Nie, L., Hu, X. and Chua, T.S., 2017, April, “Neural collaborative filtering,” Proceedings of the 26th international conference on world wide web (pp. 173-182).

[31] He, X., Chen, T., Kan, M.Y. and Chen, X., 2015, October, “Trirank: Review-aware explainable recommendation by modeling aspects,” Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (pp. 1661-1670).

[32] Nie, F., Huang, H., Cai, X. and Ding, C.H., 2010, “Efficient and robust feature selection via joint ℓ2, 1-norms minimization,” In Advances in neural information processing systems (pp. 1813-1821).

[33] Chang, K.W., Hsieh, C.J. and Lin, C.J., 2008, “Coordinate descent method for large-scale l2-loss linear support vector machines,” Journal of Machine Learning Research, 9(Jul), pp.1369-1398.

[34] Aggarwal C.C., 2016, “Neighborhood-based collaborative filtering,” Recommender systems (pp. 29-70). Springer, Cham.

[35] Zhang, H., Huang, T., Lv, Z., Liu, S. and Yang, H., 2019. “MOOCRC: A highly accurate resource recommendation model for use in MOOC environments,” Mobile Networks and Applications, 24(1), pp.34-46.

[36] (2020, July 1). Retrieved from https://www.coursera.org/

[37] (2020, July 1). Retrieved from https://www.edx.org/

[38] (2020, July 1). Retrieved from https://www.khanacademy.org/

[39] Troussas, C., Krouska A., Virvou M, 2020, “Using a Multi Module Model for Learning Analytics to Predict Learners’ Cognitive States and Provide Tailored Learning Pathways and Assessment,” Virvou M., Alepis E., Tsihrintzis G., Jain L. (eds) Machine Learning Paradigms. Intelligent Systems Reference Library, vol 158. Springer, Cham.