International Journal of Machine Learning and Cybernetics (2022) 13:1355–1368
https://doi.org/10.1007/s13042-021-01451-0

ORIGINAL ARTICLE

Multi-task learning for collaborative filtering

Lianjie Long[1] · **Faliang Huang**[1,2] · **Yunfei Yin**[1] · **Youquan Xu**[1]

Received: 26 September 2020 / Accepted: 9 October 2021 / Published online: 13 November 2021

Abstract
In the recommender system, the user's historical behavior data is one of the most important sources of the system's input data. According to the user's feedback mechanism, behavior data can be divided into explicit feedback data and implicit feedback data. However, most recommendation algorithms focus separately on explicit feedback or implicit feedback. How to combine explicit and implicit feedback for recommendation tasks has always been a research problem. In recent years, deep learning technology has dominated the research on recommendation algorithms. But even the latest neural network-based recommendation algorithm cannot exceed classic methods (such as matrix factorization) in most cases. In this work, we propose a new collaborative filtering framework with neural network architecture. On the one hand, we use both explicit feedback data and implicit feedback data as input to learn multiple representations of users and items. On the other hand, we use multi-task learning to optimize our framework and use two relatively simple auxiliary tasks to enhance the generalization ability of our framework. Extensive experiments on five real-world datasets show significant improvements in our proposed framework over the state-of-the-art methods and vanilla matrix factorization.

Keywords Recommender system · Explicit feedback · Implicit feedback · Multi-task learning

1 Introduction

In the era of information overload, the recommender system exists as an indispensable tool and is widely used in a variety of online services, including e-commerce, short videos, and social networking sites. The key to personalized recommendation is to screen out items that the user may like according to their past interactions, which is called collaborative filtering [1–4]. Collaborative filtering is the most influential and widely used model in the field of recommender systems. Among many collaborative filtering models, matrix factorization has the most generalization ability and can deal with sparse matrices and other characteristics. As shown in Fig. 1, taking movie recommendation as an example, matrix factorization [5, 6] projects users and items into a shared latent space, in which the recommender system predicts a personalized ranking over a set of items for each user with the similarities among the users and items. The recommendation task here is to use the user's history to predict the ratings of unwatched movies. During the history of the interactions between the users and the items, there are two acts, explicit feedback, and implicit feedback. Explicit feedback includes the user's ratings or views on the item, which can directly show the user's preferences and can also reflect the user's preferences. While implicit feedback includes the user's purchase, click, collection, etc., which cannot reflect the user's preference directly but can be used to mine the user's preferences.

In the past few years, since Deep Neural Networks (DNNs) are extremely good at representation learning, deep learning methods have been widely explored and have shown promising results in various areas such as computer vision and natural language processing [7–9]. Xue et al. [6]

✉ Faliang Huang
faliang.huang@gmail.com

✉ Yunfei Yin
yinyunfei@cqu.edu.cn

Lianjie Long
longlianjie@cqu.edu.cn

Youquan Xu
xuyouquan@cqu.edu.cn

[1] The College of Computer Science, Chongqing University, Chongqing 400044, China

[2] The College of Computer and Information Engineering, Guangxi Key Lab of Human-machine Interaction and Intelligent Decision, Nanning Normal University, Nanning 530001, China

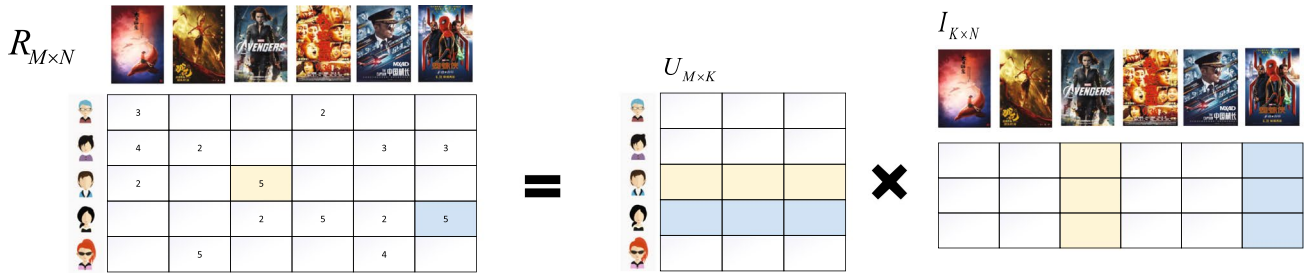

Fig. 1 Collaborative filtering for recommender sytem. $R_{M\times N}$ represents M users' rating matrix for N movies, and $U_{M\times K}$ and $I_{K\times N}$ are low-dimensional representations of users and movies, respectively

proposed a Deep Matrix Factorization (DMF), which uses a neural network architecture to replace the linear embedding operation used in vanilla matrix factorization. It uses the rows and columns in the user-item rating matrix as high-dimensional vector representations of users and items, and maps them to low-dimensional space through DNNs. In addition to learning better representation for users and items, DNNs are very suitable for learning complex interaction functions because they can approximate any continuous function [10]. NCF [11] was proposed to model the user-item interactions with a multi-layer feedforward neural network. it uses the concatenated vectors of user ID embedding and item ID embedding as input to the multi-layer perceptron (MLP) model for prediction. Using the high capacity and nonlinear characteristics of DNNs, we can learn the complex mapping relationship between user-item representations and predict scores. Recently, Rendle et al. [12] revisited the experiments in the NCF paper, proving that under the same experimental settings, the vanilla matrix factorization model after tuning can be significantly better than MLP in simulating the interaction between users and items. Obviously, the above two methods feed the neural network data differently. In this paper, we call them explicit data and implicit data respectively. Although some recent advances [13–15] have applied DNNs to recommendation tasks and shown promising results, they mostly used DNNs to model auxiliary information, such as a textual description of items, audio features of pieces of music, and visual content of images. However, they all use auxiliary information to learn the representation vectors of users and items, and do not consider the difference between explicit feedback and implicit feedback.

According to the above discussion, we can see that it seems feasible to learn to represent users and items by considering explicit data and implicit data. With this assumption, we propose a collaborative filtering framework that combines two types of feedback and uses multi-task learning optimization called Multi-task learning for collaborative filtering (MTCF). Our proposed framework has three optimization tasks. We first use these two types of feedback data to perform vanilla matrix factorization tasks to obtain predicting scores of items and then cross the low-dimensional representations of users and items produced in the two matrix factorization tasks. The specific method is to cross the explicit user representation and the implicit user representation to obtain a higher-order integrated user representation, and we use the same method for items. After obtaining the comprehensive representation, we then use DNNs to learn the complex mapping relationship between the two types of feedback features and the integrated features and finally obtain the predicting score through user-item interaction. It is worth mentioning that we are not only outputting the predicting score after comprehensive crossover but outputting all three modules' predicting scores as our final main task's predicting score.

Figure 2 illustrates our key ideas. We take user representation as an example. For the representation learning of single feedback data, there is only one user representation at the end, but for our model, considering explicit feedback data and implicit feedback data, there will be at least three user representations. Multiple user representations reflect different features of users, which provides more benefits for our next recommendation task.

The main contributions of this work are as follows.

- We propose a new collaborative filtering framework that can utilize explicit feedback information and implicit feedback information at the same time, and mine the cross-features of the two types of feedback information. It is worth mentioning that the model has good generalization performance and can be widely used in most scenarios with explicit feedback and implicit feedback.
- We use multi-task learning to train our model, give full play to the generalization ability of the model, accelerate the convergence of the model, and improve the effectiveness of the model. As far as we know, we are the first to use multi-task learning for collaborative filtering that combines implicit and explicit feedback.
- We perform extensive experiments on 5 real-world datasets to demonstrate the effectiveness and rationality of the proposed MTCF framework.

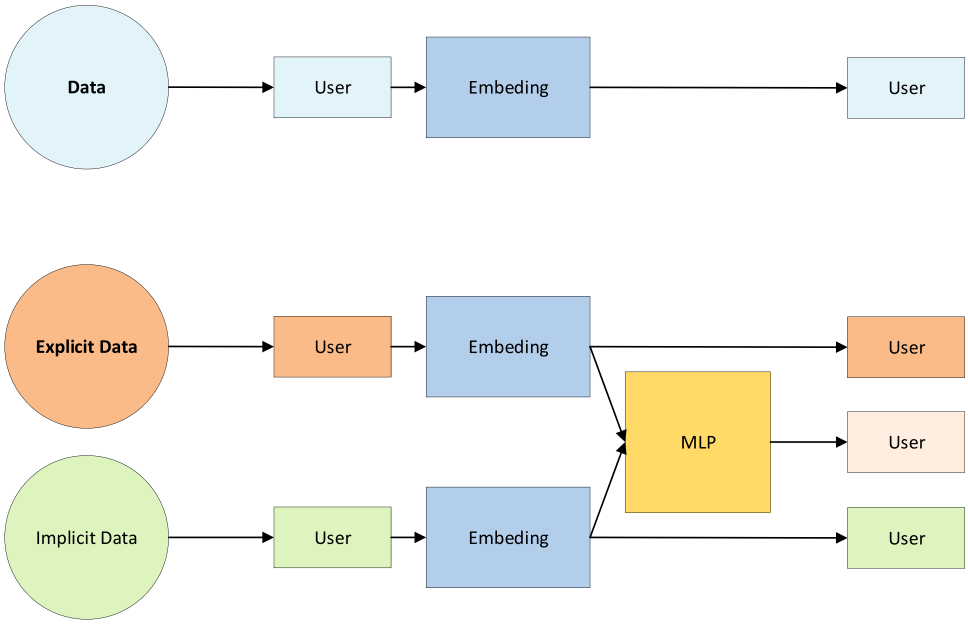

Fig. 2 User representation learning with single feedback data and user representation of our model

2 Related work

2.1 Collaborative filtering

In the recommender system, the historical interaction information between the user and the item is the key to collaborative filtering, and the user's explicit feedback just reflects the user's preference. Collaborative filtering with explicit data uses the users' direct ratings or comments on items to perform scoring prediction tasks. Singular Value Decomposition (SVD) [16] is an early model for the matrix factorization method, which predicts user ratings on items by decomposing the rating matrix into two small matrices. After, Lee et al. [17] proposed a non-negative matrix method, which enhanced the interpretability after matrix factorization. The success of the Netflix Prize has set off a wave of research on recommendation algorithms. Salakhutdinov et al. [18] applied the restricted Boltzmann machine to the Netflix dataset with great success, and then the model was extended to the item order scoring task. Immediately after, Georgiev et al. [19] proposed a hybrid RBM framework that used both user-based and item-based RBM frameworks, used the rating matrix as input to learn the hidden layer distribution, and tried to reconstruct the rating matrix. Recently, DMF was [6] proposed to use a bidirectional path neural network architecture to replace the linear embedding used in matrix factorization, and design a new loss function to optimize the model. This direct use of the rating matrix as an input only makes use of explicit feedback data.

Since most users do not tend to rate items, it is often difficult to collect explicit feedback data. ALS [8] and SVD++ [20] are two early effective tasks that use implicit feedback for recommendation tasks. Both models ignore the rating value and use binarized implicit feedback for the recommendation. NCF [11] was proposed to use element product to replace the inner product operation in traditional matrix factorization, and interprets the matrix factorization method as a special case of the NCF method. Further, NCF has two modules Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP) to learn the relationship between linear and non-linear. Using linear fusion to fuse the two models to improve model performance. Bai et al. [21] proposed a new Neighbor-based Neural Collaborative Filtering (NNCF) model. For the first time, the neighborhood model was integrated into neural collaborative filtering. This method improves the NCF model performance by constructing user-item neighborhoods as input. In the collaborative filtering model based on deep learning, Zhang et al. [18, 22] introduced the attention mechanism into the recommender system to learn the relative weight of each user-item interaction to better learn the user's instantaneous interests. However, they only resort to implicit data when building a model.

Some studies have found that there is a complementary relationship between explicit feedback and implicit feedback [23–25], and applying both of them at the same time is likely to improve the recommendation effect. Robert M. Bell and Yehuda Koren [26] cut in from the perspective of combining explicit feedback and implicit feedback, mining explicit implicit feedback data from movie recommendations, using score data as explicit feedback, and factoring and based on the neighborhood model fuse these two types of feedback data for recommendation tasks. Weike Pan [27] first clusters the user set and item set through K-means and proposes a factorization machine model that incorporates

explicit implicit feedback based on transfer learning. Gai Li [28] combines the advantages of xCLiMF [25] and SVD++ [20] for recommendation tasks while combining explicit and implicit feedback, and proposes a new evaluation method ERR (Expected Reciprocal Rank) to evaluate the recommendation quality of the algorithm. Chen et al [29]. performed weighted low-rank processing on implicit feedback data to better leverage the ability of implicit feedback data to reflect users' hidden preferences. Liu et al. [24] considered the heterogeneity of explicit implicit feedback, that is, explicit feedback is mostly numerical, and implicit feedback is mostly binary, to eliminate the numerical difference between the two, both Convert to a value between 0-1 and set different weights for them respectively. However, through this approach, the ability of explicit feedback data to reflect the user's preference is weakened, and the important characteristics of explicit feedback data have not been considered. What is different from the above work is that we retain the respective characteristics of explicit data and implicit data, and mine the deep data comprehensive characteristics through neural networks, and use appropriate methods for training. We make full use of implicit feedback data to reflect users' hidden preferences and explicit feedback data to reflect user degree of preference. In Table 1, we summarize all the pertinent characteristics of implicit and explicit feedback.

2.2 Multi-task learning

Multi-task learning [30] is a derivation transfer learning method. The main task uses the domain-related information possessed by the training signals of the related tasks as a derivation bias to improve the generalization effect of the main task. In recent years, multi-task learning has become more and more popular, because the success of machine learning and deep learning is mainly due to the model's better access to data representation and the ability to mine the required information from the data. Multi-task learning can obtain more comprehensive and changeable information from the data. The features extracted by the single task model are only valid for the single task, and a single feature cannot describe a sample well. When the amount of tasks is large and the learned features are required to serve each task, that is, the features are required to have a certain generality, multi-task learning is more suitable. Multi-task learning is generally divided into two types, one is divided into one main task and auxiliary tasks, the auxiliary tasks are to help the main task to train. The other is multiple Equal tasks, there is no major or minor. The former is used in our work. Choosing appropriate auxiliary tasks is the key to the success of the multi-task learning framework [31].

Multi-task learning has many advantages in recommending tasks [32]. For example, multiple tasks can share a part of the network structure as Fig. 3, and the learned user and item vector representations can be easily migrated to other tasks. Besides, the correlation between different tasks has a greater impact on the multi-task learning effect. In our work, the auxiliary task we use is part of our model, so it has a high correlation with the main task. Recently, multi-task learning is used to solve multiple problems simultaneously in the recommender system. Based on the user's decision-making process, Hadash et al [33]. divided the recommendation task into a ranking task and a scoring task, and proposed a multi-task framework to jointly train these two tasks. This is the first work to apply multi-task learning to collaborative filtering. Lu et al [34]. proposed a multi-task learning framework that combines probabilistic matrix factorization (PMF) and adversarial Seq2Seq model. The matrix factorization model can be used to obtain user ratings for items. The Seq2Seq model can generate user comments on items and improve recommendations. While predicting the accuracy, it can solve the difficulty of providing interpretable recommendation results in the recommendation system to a certain extent. Based on the idea of Multi-Task Learning, Ma et al [35]. proposed a new CVR estimation model—ESMM, which effectively solved the two key problems of data sparseness and sample selection bias faced by CVR estimation in real scenes. With the background of Taobao search and recommendation scenarios, Ni et al [36]. used a multi-task model

Table 1 Characteristics of explicit and implicit feedback

	Explicit feedback	Implicit feedback
Accuracy	High	Low
Abundance	Low	High
Context-sensitive	Yes	Yes
Expressivity of user preference	Positive and negative	Positive
Measurement reference	Absolute	Relative

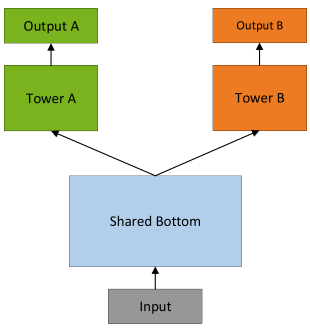

Fig. 3 General multi-task learning model framework. The Shared-Bottom network is usually at the bottom, denoted as f, and multiple tasks share this layer. Up, the K subtasks correspond to a tower network, denoted as h_K, and the output of each subtask is $y_K = h_K(f(x))$

to learn the general representation of users, and compared some experimental effects of the multi-task model and the single-task model. Zhao et al [37]. used multi-task learning to predict the two key tasks of video recommendation scenarios, including whether the user would click on the video and the user's feedback after watching the video. The above two key tasks can be used as implicit feedback tasks and explicit feedback tasks respectively. Inspired by the video recommendation work, we further use multi-task learning for collaborative filtering, using two types of feedback data to construct different collaborative filtering tasks.

3 Preliminaries

Suppose there are M users $U = \{u_1, ..., u_M\}$, N items $I = \{i_1, ..., i_N\}$. Each item can be a book, a movie, or a web page. In the explicit feedback data, let $R \in R^{M \times N}$ denote the rating matrix, where R_{ui} is the rating of user u on item i.

$$y_{ui} = \begin{cases} R_{ui}, & \text{if } R_{ui} \text{ is observed} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In the implicit feedback data, let $A \in A^{M \times N}$ denote the interact matrix, where A_{ui} may be click, favorite, browse, etc.

$$y_{ui} = \begin{cases} 1, & \text{if interaction } (u, i) \text{ is observed} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

In particular, for two types of feedback information, $y_{ui} = 0$ does not mean user u does not like i. In fact, there are too many items in a system, and user u may have never observed item i. The recommendation problem with explicit feedback is usually formulated as a rating prediction problem that estimates the missing values in the rating matrix R. Finally, we select top-k items to recommend to users by sorting the predicted scores of the items. Similarly, to settle the recommendation problem with implicit feedback, we can formulate it as an interaction prediction problem that estimates the missing values in the interaction matrix [38]. It should be emphasized that in order to eliminate the difference between the predicted values of the two types of recommendation problems, we convert them into values between 0–1. Model-based approaches [20, 39] assume that there is an underlying model which can generate all ratings as follows.

$$\hat{y}_{ui} = f(u, i \mid \Theta) \tag{3}$$

Where $\hat{y}_{ui}$ denotes the predicted score of interaction matrix between user u and item i,Θ denotes the model parameters, and f denotes the function that maps the model parameters to the predicted scores. The key to the problem is how to define such a function f. Let p_u and q_i denote the latent representations of u and i, respectively. Latent Factor Model (LFM) [40] simply applied the dot product of p_u, q_i to predict the $\hat{y}_{ui}$ as follows.

$$\hat{y}_{ui} = f(u, i \mid \Theta) = p_u^T q_i = \sum_{k=1}^{K} p_{uk} q_{ik} \tag{4}$$

where K denotes the dimension of the latent space, $K \ll \min(M, N)$. In addition, based on the calculation of the similarity between the user and the item to reflect the predicted score, we can use cosine similarity to predict $\hat{y}_{ui}$ as follows.

$$\hat{y}_{ui} = f(u, i \mid \Theta) = \text{cosine}(p_u, q_i) = \frac{p_u^T q_i}{\|p_u\| \|q_i\|} \tag{5}$$

Both dot product and cosine similarity are used in our work. In general matrix factorization using implicit feedback data [6, 11, 12, 40], the dot product is often used to calculate the similarity between users and items due to its excellent performance. But unlike the binary implicit feedback, the explicit feedback is mostly numeric, and its value reflects the user's degree of interest. In order to choose a suitable similarity measure for explicit feedback data, we conduct some preliminary experiments and find that cosine similarity stands out among many similarity calculation methods. So we use dot product for implicit feedback and cosine similarity for explicit feedback. Neural collaborative filtering proposes to use MLP to automatically learn f. Their motivation is to learn the nonlinear interaction between users and items. We did not follow neural collaborative filtering, because we tried to learn the explicit and implicit higher-order features of users and items through a deep representation learning framework to obtain users' comprehensive interests.

Now, the next question is how to learn model parameters, and many of the existing works generally estimate parameters through optimizing an objective function. Recommender systems are often abstracted into learning to rank or predicting rating, often using two types of objective functions, pairwise loss, and point-wise loss. In this paper, we predict the user's rating of items based on the user's explicit feedback information. Point-wise loss is widely used in collaborative filtering regression models based on explicit feedback [41]. For our regression prediction object, we use the most commonly used point-wise loss which is the squared loss to learn the parameters by minimizing the squared error between y_{ui} and $\hat{y}_{ui}$.

$$L_{sqr} = \sum_{(u,i) \in y^+ \cup y^-} w_{ui} \left(y_{ui} - \hat{y}_{ui}\right)^2 + \lambda \|\theta\|_2^2 \tag{6}$$

Where y^+ denotes all the observed interactions and y^- denotes the sampled unobserved interactions, and w_{ui} denotes the weight of training instance (u, i). θ denotes all trainable model parameters and λ controls the L_2 regularization strength to prevent overfitting. we adopt the Adaptive Moment Estimation (Adam) [42], which adapts the learning rate for each parameter by performing smaller updates for frequent and larger updates for infrequent parameters. The Adam method yields faster convergence than Stochastic Gradient Descent (SGD) and gets out of trouble of tuning the learning rate. In summary, our recommendation task can be described as a problem of predicting scores, through the vector representation of users and items to interact with each other to obtain the predicting scores of items. Finally, the model is optimized by minimizing the squared loss.

4 The proposed framework

Our framework aims to make full use of explicit feedback data and implicit feedback data, and considering the generalization of the model, our framework has three tasks, two auxiliary tasks, and one main task. Figure 4 illustrates our proposed architecture. The green and orange parts are extracted separately as our two auxiliary tasks, the collaborative filtering task with implicit data (ICF) and the collaborative filtering task with explicit data (ECF). The entire framework is the main task of our training.

In particular, for the conversion of explicit feedback data to implicit feedback data, we take the MovieLens dataset as an example. Movie ratings include 1, 2, 3, 4, 5 (observed), and missing value (unobserved). There are three main ways to convert explicit feedback to implicit feedback:

(a) rating $\geq 3, r = 1$(observed, positive sample); otherwise, $r = 0$(unobserved, negative sample);
(b) rating $\neq \emptyset, r = 1$(observed, positive sample); otherwise, $r = 0$(unobserved, treat all missing items as negative samples);
(c) rating $\neq \emptyset, r = 1$(observed, positive sample); otherwise, $r = 0$(unobserved, sample all missing items and select some as negative samples);

Where $\emptyset$ means there are no ratings. The third processing method is adopted in our work.

In real scenarios, user behaviors often contain different implicit feedbacks such as click, favorite, and browse, and these different feedbacks may reflect user interests. Taking different implicit feedbacks into consideration may improve recommendation performance to some certain. Owing to the unavailability of data containing different implicit feedbacks and the limited computing resource, all the different feedbacks are not treated differently in

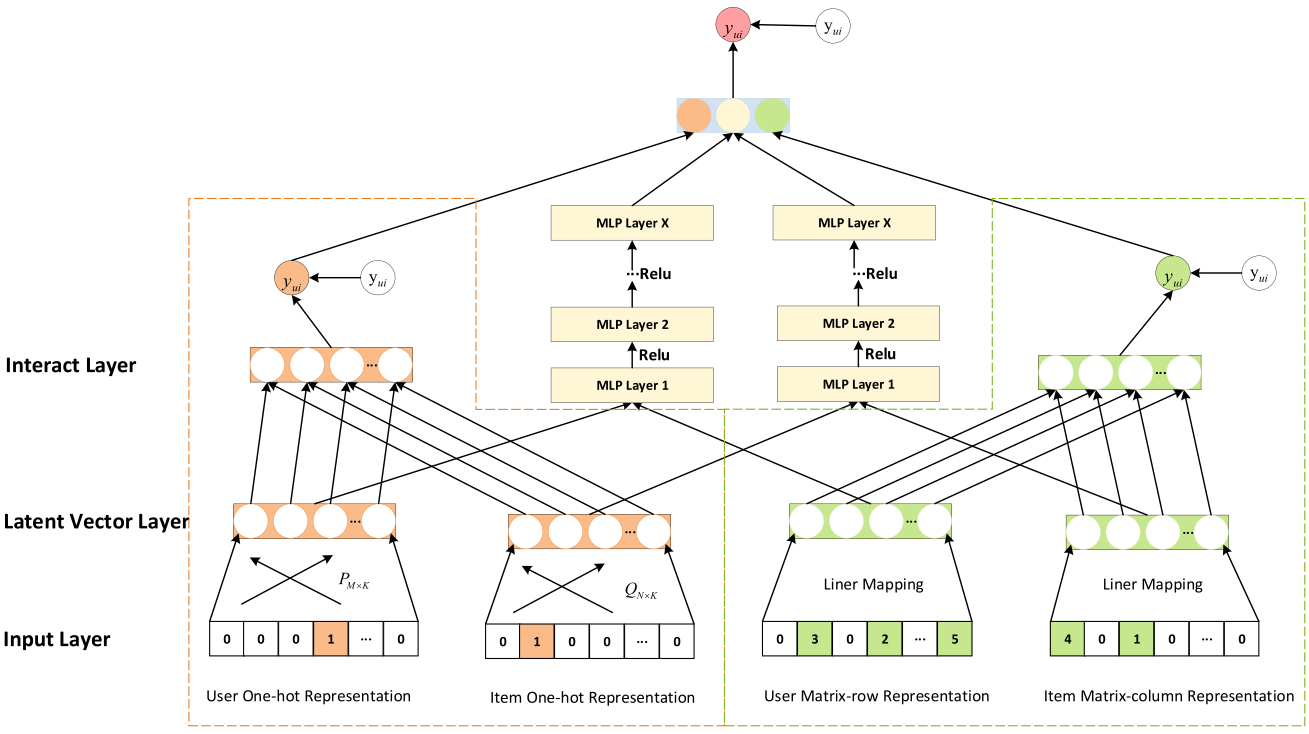

Fig. 4 The architecture of Multi-task learning for collaborative filtering Models. The orange and green parts are two auxiliary tasks that use implicit data and explicit data, respectively

this paper. In fact, the proposed model in the article is easily extended to utilizing different implicit feedbacks for the recommendation, and this is also the focus of our future work.

4.1 Collaborative filtering task with explicit data and implicit data

In the orange part, we use the one-hot encoding of the user (item) ID as the input and let $V_u^U(V_i^I)$ denotes the one-hot encoding of the user (item) ID. Let user latent vector p_u and item latent vector q_i be represented as follows.

$$\begin{aligned} p_u &= P^T V_u^U \\ q_i &= Q^T V_i^I \end{aligned} \tag{7}$$

We use Eq.(4) to predict item scores. From this, the output of collaborative filtering task with implicit data (ICF) can be defined as follows.

$$\hat{y}_{ui}^{ICF} = a_{\text{out}}\left(p_u^T q_i\right) \tag{8}$$

Where a_{out} denotes the activation function, using sigmoid $(x) = 1/(1+e^{-x})$ to map our output value between [0,1]

In the green part, we use the interact matrix $y^{M\times N}$ as input, and Each row y_{u*} and each column y_{*i} in y represent a user and an item, respectively. Since the initial input is a high-dimensional vector with two different dimensions, we must map it to a low-dimensional space of the same dimension to facilitate our subsequent operations. We simply use a linear regression function to complete this mapping, and the latent vectors of users and items can be defined as follows.

$$\begin{aligned} p_u &= y_{u*} W_u + b_u \\ q_i &= y_{*i} W_i + b_i \end{aligned} \tag{9}$$

where W_* and b_* denote the weight matrix and bias vector, respectively. Here we use Eq.(5) to predict item scores. From this, the output of collaborative filtering task with explicit data (ECF) can be defined as follows.

$$\hat{y}_{ui}^{ECF} = a_{\text{out}}\left(\text{cosine}\left(p_u, q_i\right)\right) \tag{10}$$

Similarly, where a_{out} denotes the activation function, using sigmoid $(x) = 1/(1+e^{-x})$ as a_{out} to map our output value between [0,1].

In the previous two parts, we got the user and item representations under two types of feedback data. Then we cross the two representations to obtain a more complex representation. Here we use the product of elements ($\odot$) to complete the cross-features. The latent vectors of new users and items are represented as follows.

$$\begin{aligned} p_u &= p_u^{ICF} \odot p_u^{ECF} \\ q_i &= q_i^{ICF} \odot q_i^{ECF} \end{aligned} \tag{11}$$

Next, we use MLP to further learn the comprehensive latent representation of users and items. Therefore, the user's representation learning part can be defined as follows.

$$\begin{aligned} a_0 &= W_0^T p_u \\ a_1 &= a\left(W_1^T a_0 + b_1\right) \\ &\cdots\cdots \\ p_u &= a_x = a\left(W_x^T a_{x-1} + b_x\right) \end{aligned} \tag{12}$$

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. In this paper, we use Rectifier (ReLU) as the activation function. The same method can be used to obtain the comprehensive potential representation of the item q_i. Finally, we also use cosine similarity to predict item scores. From this, the IECF output can be defined as follows.

$$\hat{y}_{ui}^{IECF} = \text{cosine}\left(p_u, q_i\right) \tag{13}$$

Finally, we can get the final output of the main task as follows.

$$\hat{y}_{ui}^{\text{main}} = \text{sigmoid}\left(\hat{y}_{ui}^{ICF} + \hat{y}_{ui}^{IECF} + \hat{y}_{ui}^{ECF}\right) \tag{14}$$

4.2 Multi-tasks

To effectively learn parameters for the recommendation, as well as preserve the generalization ability of the framework, we use ICF and ECF as independent auxiliary tasks and our entire model as the main task. Compared with the main task, the two auxiliary tasks are relatively simple, so the multi-task learning strategy we adopt is to combine simple tasks and complex tasks. When training on three tasks, our training set is the same, and all the parameters of the model are shared, which achieves the effect of knowledge transfer and can accelerate the model convergence. For the optimization of the three tasks, we use the square error loss function of Eq.(6).

$$\begin{aligned} L_{\text{main}} &= \sum_{(u,i)\in y^+\cup y^-} w_{ui}\left(y_{ui} - \hat{y}_{ui}^{\text{main}}\right)^2 \\ L_{ICF} &= \sum_{(u,i)\in y^+\cup y^-} w_{ui}\left(y_{ui} - \hat{y}_{ui}^{ICF}\right)^2 \\ L_{ECF} &= \sum_{(u,i)\in y^+\cup y^-} w_{ui}\left(y_{ui} - \hat{y}_{ui}^{ECF}\right)^2 \end{aligned} \tag{15}$$

For the convenience of calculation, we map our output value $\hat{y}_{ui}$ between [0,1]. So, here all y_{ui} comes from Eq.(2).

Generally, there are two training methods for multi-task learning, one is alternating training and the other is joint training. Alternate training means that in iterative training we alternately perform loss learning for each task, and joint training means that we train all tasks in each epoch and then integrate their respective losses. Wang et al. [43] also use multi-task learning when they use knowledge graphs to complete recommended tasks. They alternately train recommendation tasks and knowledge graph embedding tasks. As can be seen from their code, for every 5 epochs, 4 of them are in the training recommendation task, and the remaining one is in the training knowledge graph embedding task. Xin et al. [44] divided recommendation tasks into the user-item preference modeling task and the item-item relationship modeling task. They jointly trained the two tasks to obtain a total objective function. In our work, we adopt the latter.

The next key question is how to integrate multiple losses, and the first method we tried was to simply add up the different losses. Soon we found that the scale of the loss of different tasks is very different, resulting in the overall loss being dominated by a certain task, and ultimately leading to the loss of other tasks that cannot affect the learning process of the network shared layer. Moreover, when the loss of the main task is very small, we do not want the auxiliary task to change the model parameters significantly, so we designed a total objective function as follows.

$$\min_{\theta} L = L_{\text{main}} + L_{\text{main}}\left(\alpha L_{ICF} + \beta L_{ECF}\right) \tag{16}$$

Where α and β are relative weights of auxiliary tasks. Before the experiment, to explore the relative weights α and β, we conduct some preliminary experiments. From Fig. 5, taking ML100K dataset as an example, we can see that the ECF training doesn't take long before the loss is almost fixed, while the loss of ICF continues to decline. The convergence speed of ECF is faster than ICF, and we hope that ICF will be fully trained. When the ICF converges quickly, the loss of the two auxiliary tasks is about 4 times the relationship. In Eq.(15), so we try to increase the relative weight of the loss of ICF. In our work, for ML100K dataset, we simply set $\alpha = 4$, $\beta = 1$. In the future, we will deeply explore the relationship between auxiliary tasks. The training procedure for MTCF is illustrated in Algorithm 1.

Algorithm 1 Multi-task training for collaborative filtering

Input: Explicit interaction matrix R, Implicit interaction matrix A, learning rate η
Output: Prediction function $f(u, i \mid \Theta, R, A)$

initialize all parameters Θ
for number of training iteration **do**
  Sample minibatch from interactions
  Compute L_{ICF} according to Eq.(7), Eq.(8), Eq.(6)
  Compute L_{ECF} according to Eq.(9), Eq.(10), Eq.(6)
  Compute L_{main} according to Eq.(7)-Eq.(14), Eq.(6)
  $L \leftarrow L_{\text{main}} + L_{\text{main}}\left(\alpha L_{ICF} + \beta L_{ECF}\right)$
  for each parameter $\vartheta \in \Theta$ **do**
    Compute $\partial L / \partial \vartheta$ on the mini-batch by back-propagation
    Update $\vartheta \leftarrow \vartheta - \eta \cdot \partial \mathcal{L} / \partial \vartheta$
  end for
end for
Return all parameters in Θ

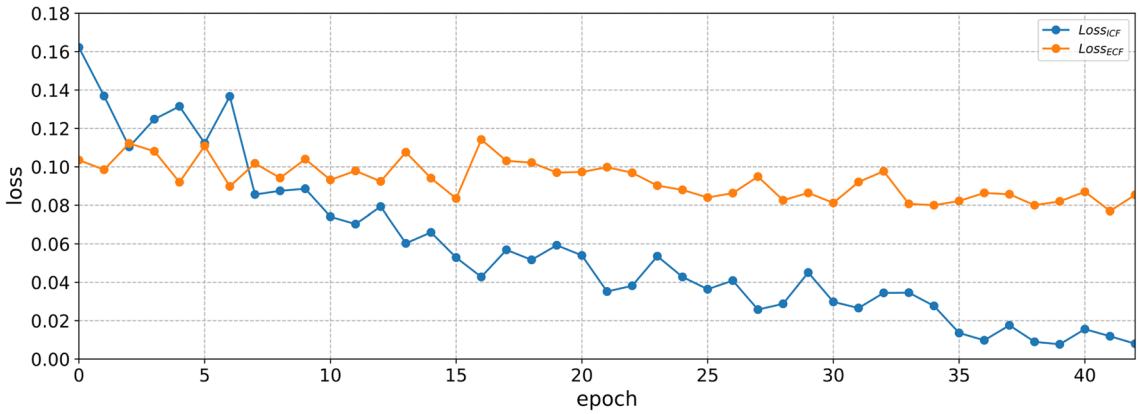

Fig. 5 The loss of two auxiliary tasks training alone

5 Experiments

In this section, we prove the effectiveness of our proposed framework through experiments and perform a series of extensive experiments to compare the performance of different experimental settings, such as the number of negative samples and the number of network layers.

5.1 Experimental settings

5.1.1 Datasets

We evaluate our proposed framework on five benchmark datasets: MovieLens 100K (ML100k), MovieLens 1M (ML1M), LastFM, Amazon music (AMusic), Amazon toy (AToy). The MovieLens datasets have been preprocessed by the provider. Each user has at least 20 ratings and each item has been rated by at least 5 users. For the LastFM dataset, we do not filter any users and ratings, and use this version of the dataset directly. For the other 3 datasets, we use the same method as MovieLens. The statistics of these five datasets are summarized in Table 2.

5.1.2 Evaluation for recommendation

To evaluate the performance of item recommendation, we adopted the leave-one-out evaluation, which has been widely used in the literature [5, 11, 12, 45]. The latest interaction of each user is used for testing and the remaining dataset for training. Since ranking all items is time-consuming, we randomly sample 99 unobserved interactions for each user. We then rank the 100 items according to the prediction. The performance of a ranked list is often measured by Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [46]. In our experiments, we truncated the ranked list at 10 for both metrics. Intuitively, the HR measures whether the test item is present on the top-10 list or not, and the NDCG measures the ranking quality which assigns higher scores to hit at top position ranks.

Table 2 Statistics of the evaluation datasets

Datasets	Interactions	Items	Users	Density (%)
ML100K	100000	1683	944	6.29
ML1M	1000209	3706	6040	4.47
LastFM	69149	2665	1741	1.49
Amusic	46087	12929	1776	0.20
Atoy	40926	3393	3317	0.07

5.1.3 Adapting to temporal changes

A key assumption of most machine learning models is that the input is independent and identically distributed. This is not strictly true in the field of recommendation since the user's behavioral preferences change with time, recent behaviors can better represent the current user's preferences. In addition, the recommender system task is to predict the user's next click. Therefore [47], learning a recommendation model on the entire dataset may lead to worse performance because the model ends up focusing on some out-of-date properties. One way to deal with this is to discard early data, but this will reduce the amount of our training data. We propose a simple solution to get the best of both worlds via pre-training. We first use the entire dataset to pre-train a model, and then further train the model using only a subset of recent data, e.g. the last week worth of data out of a month of interactions.

5.2 Performance comparison

We compared our proposed MTCF method with the following methods. Since the proposed models focus on modeling the relationship between users and items, we mainly compare with user-item models.

- **ItemPop** [19] is a non-personalized method that is often used as a benchmark for recommendation tasks. It ranked the items by their popularity judged by the number of interactions.
- **ItemKNN** [3] is the standard item-based collaborative filtering method.
- **eALS** [5] is a state-of-the-art MF method for recommendation with square loss. It used all unobserved interactions as negative instances and weighted them non-uniformly by the item popularity.
- **MF** is the standard matrix factorization that models the user preference with the inner product between user and item embeddings.
- **NeuMF** [11] is a state-of-the-art representation learning-based MF method which performs deep matrix factorization with normalized cross-entropy loss as the loss function.
- **DMF** [6] is a state-of-the-art representation learning-based MF method which performs deep matrix factorization with normalized cross-entropy loss as the loss function.
- **DeepCF** [38] is an improved algorithm based on representation learning and matching learning, which combines the advantages of the two models and uses a high-dimensional vector of user-item explicit feedback as input for users and items.

Table 3 Comparison results of different methods in terms of HR@10 and NDCG@10

	ML100K		ML1M		LastFM		AMusic		AToy	
	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
ItemPop	0.420	0.235	0.453	0.254	0.591	0.407	0.239	0.125	0.284	0.152
ItemKNN	0.600	0.334	0.638	0.373	0.846	0.766	0.356	0.222	0.343	0.215
eALS	0.623	0.356	0.709	0.426	0.854	0.731	0.371	0.235	0.372	0.243
MF	**0.688**	**0.401**	**0.714**	**0.432**	**0.874**	**0.790**	0.381	0.248	0.419	**0.263**
NeuMF	0.681	0.388	0.703	0.427	0.870	0.757	0.357	0.225	0.391	0.237
DMF	0.672	0.390	0.712	0.429	0.867	0.759	**0.417**	**0.253**	**0.426**	0.252
DeepCF	0.674	0.386	0.710	0.422	0.867	0.759	0.409	0.252	0.419	0.259
EIFCF	0.687	0.399	0.711	0.428	0.871	0.789	0.408	0.255	0.417	0.256
MTCF	**0.695**	**0.407**	**0.719**	**0.442**	**0.887**	**0.793**	**0.506**	**0.310**	**0.530**	**0.319**
Improvement	1.0%	1.5%	0.7%	2.3%	1.5%	0.4%	21.3%	22.5%	24.4%	21.3%

- **EIFCF** [29] is a collaborative filtering algorithm that integrates implicit feedback and explicit feedback in stages.

We use the Pytorch framework to implement our proposed method. All MTCF tasks are learned by optimizing the squared loss of Eq.(6), where we use uniformly a two-layer neural network. It is worth noting that for the neural network, we randomly initialize the training parameters to a Gaussian distribution (with a mean of 0 and standard deviation of 0.01), and Use the mini-batch Adam optimization model. We set the batch size to 512 and the learning rate of [0.0005,0.0001,0.00005].

The results of the comparison are summarized in Table 3. The best and the second best scores are shown in bold. According to the table, we have the following key captures:

- Our results prove the feasibility of neural networks in the recommendation problem. Although the MF performs better than state-of-the-art representation learning-based methods on high-density (*density* > 1%) datasets, the latter are better on sparse datasets. However, as for proposed architecture, on all datasets, our models achieve the best performance in both metrics of NDCG and HR, compared to other methods.
- In particular, on sparse datasets, compared to state-of-the-art representation learning-based methods, our model obtain 21.3-24.4% (22.9% average) and 21.3-22.5% (21.9% average) relative improvements in NDCG and HR metrics, respectively. Even compared to the MF on high-density datasets, our model also gets 0.7-1.5% (1.1% average) and 0.4-2.3% (1.4% average) relative improvements in NDCG and HR metrics, respectively.
- In terms of explicit feedback and implicit feedback, our method is significantly better than DMF, which only uses explicit feedback data, and is also better than NeuMF, DeepCF, etc., which rely only on implicit feedback. In addition, for EIFCF, which combines explicit feedback and implicit feedback in stages, our multi-task learning strategy shows better results.

5.3 Impact of pre-training

In our work, we adapt time changes for pre-training as mentioned earlier. We first use the entire dataset to pre-train a model, and then further train the model using only a subset of recent data. In order to show that this training strategy is not only effective for our model, but also applicable to other methods, we have performed pre-training operations on the above methods, and the results are shown in Table 4. There are two points to note. For one

Table 4 Comparison results with pre-training of different methods in terms of HR@10 and NDCG@10

	ML100K		ML1M		LastFM		AMusic		AToy	
	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
MF-pre	**0.716**	**0.437**	**0.774**	**0.516**	**0.875**	**0.794**	**0.382**	**0.249**	**0.42**	**0.262**
NeuMF-pre	**0.684**	**0.404**	**0.759**	**0.498**	0.867	**0.767**	0.355	0.225	0.386	0.225
DMF-pre	**0.705**	**0.431**	**0.756**	**0.481**	**0.872**	**0.770**	0.413	0.249	0.423	0.248
DeepCF-pre	**0.691**	**0.424**	**0.744**	**0.471**	0.864	**0.763**	0.409	0.249	0.413	0.251
EIFCF-pre	**0.732**	**0.437**	**0.754**	**0.486**	0.862	0.761	**0.411**	**0.257**	**0.423**	0.239
MTCF-pre	**0.755**	**0.456**	**0.777**	**0.522**	**0.888**	**0.797**	**0.508**	0.308	**0.542**	**0.325**

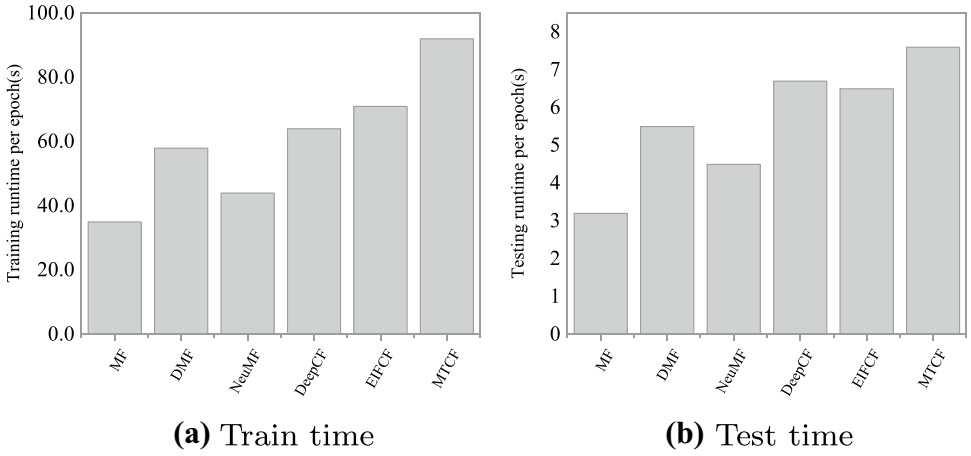

Fig. 6 Efficiency comparison between MTCF and other baseline and state-of-the-art models on average runtime per epoch on ML1M dataset

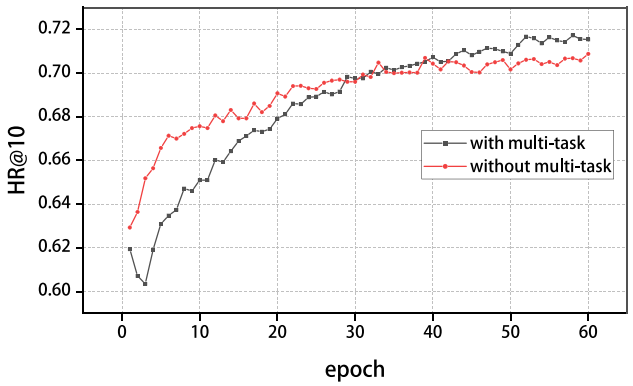

Fig. 7 The effect of multi-task learning strategy on ML1M dataset

thing is that we only choose the strongest MF for the non-deep neural network model, for another is that if there is an improvement, we show it in bold. From Table 4, we can observe that almost most methods have improved on most datasets. Look carefully, the improvement is more obvious on the high-density dataset, but the improvement is less on the sparse dataset, and some may even cause performance degradation due to insufficient training data. However, in general, our method is better than any baseline method, especially in MovieLens datasets. Even on sparse datasets, there is a small improvement.

5.4 Research on the effectiveness of multi-task learning

5.4.1 Time efficiency

Generally, traditional neural networks with iterative training mechanisms may cause time-consuming training of the entire model [48–50], so we discussed the training time of the model. Figure 6 shows the running time comparison of MTCF and baseline models and state-of-the-art models on ML1M. It is worth noting that we use the leave-one-out test method, so the test time is not much different. The vertical axis shows the average running time of each epoch of all models and the hardware settings of all models are consistent. From the figure, we can find that during training, compared with other methods, MTCF has the longest training time per epoch. However, in each training, our model has three optimization tasks and compared with other single-task neural network models, the time is not much. Compared with other non-neural network models MF and EIFCF, the time of our proposed model is mainly spent on constructing multiple representations of user items.

5.4.2 Ablation Study of multi-task learning

In order to explore the effectiveness of multi-task learning, we removed two auxiliary tasks and performed an ablation experiment on ML1M dataset. The experimental results are shown in Fig. 7. It can be seen from the figure that the multi-task model with auxiliary tasks does not perform as well as the single-task model due to the noise of the multi-task at the beginning. But after about 30 epochs, the multi-task model will surpass the single-task model with its own advantages.

5.5 Sensitivity to hyper-parameters

Although the time factor pre-training mode has higher performance for high-density datasets, it is not so obvious for sparse datasets, and may even decline. Therefor, in order to make the experiment fairer and to explore the sensitivity to hyper-parameters of the model itself, our experiments in this section did not use pre-training.

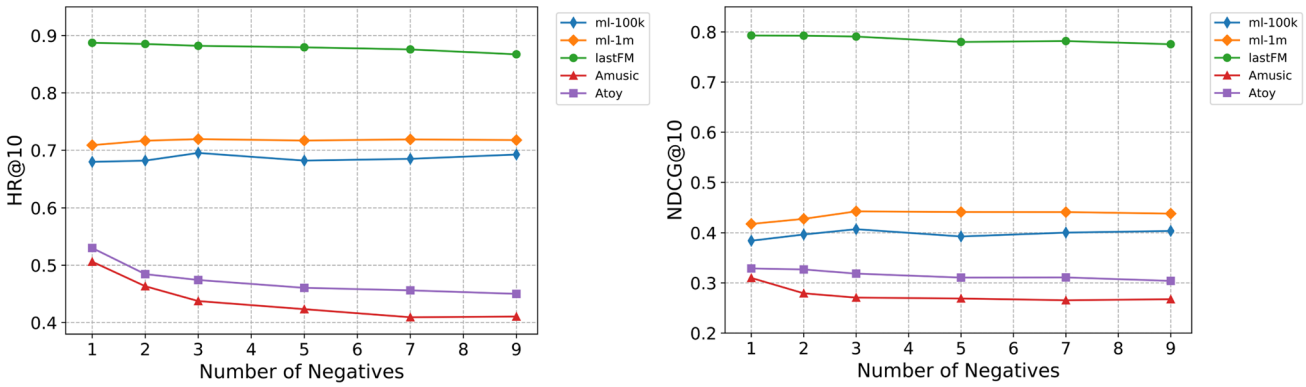

Fig. 8 The effect of negative sampling ratio on performance

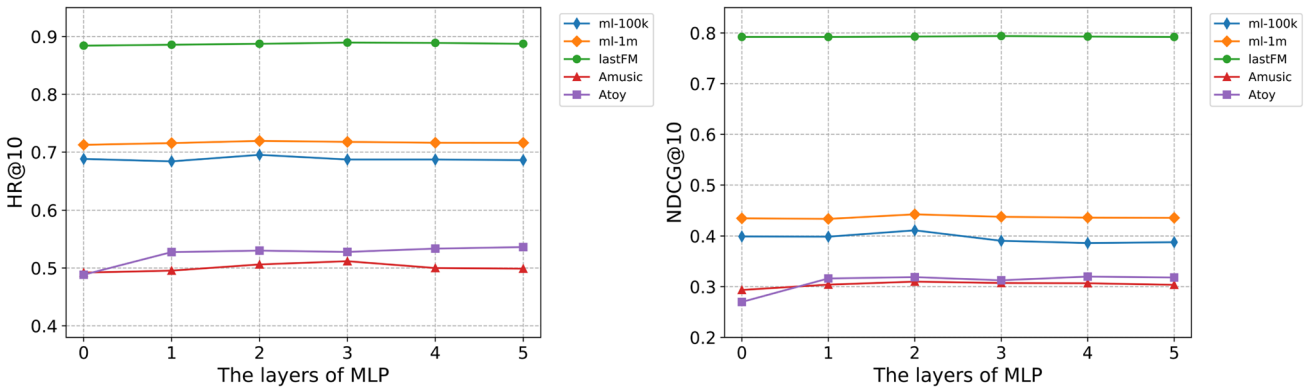

Fig. 9 The performance with different deep layers

5.5.1 Negative sampling ratio

Compared with the pair-wise objective function, one advantage of point-wise loss is the flexible sampling ratio for negative instances [51, 52]. To illustrate the impact of negative sampling for MTCF method, we test different negative sampling ratios, i.e., the number of negative samples per positive instance, on all the datasets. From the results in Fig. 8, we can find that the number of sampling instances is also related to the sparsity of the dataset. The more sparse the dataset, the fewer the number of negatives. For ML1M dataset, the optimal negative sampling ratio is around 3 to 5 which is consistent with the results by previous work [11]. Sampling more negative instances not only requires more time to train the model but also reduces the performance of the model.

5.5.2 Depth of layers in network

In our proposed model, we cross explicit feedback and implicit feedback data to get the complex features of both through the neural network with multiple hidden layers. We conduct an extensive experiment on the datasets to investigate our model with the different number of hidden layers. From the results in Fig. 9, the neural network has indeed achieved a good performance, compared with the zero-layer. The above performance is particularly evident on the Amazon datasets (Amusic, Atoy). The deeper the depth, the better the performance on the Atoy dataset. While on the Amusic dataset, it is not so stable, and when the number of layers exceeds three, the performance starts to decrease. on the two MovieLens datasets, our model with two layers illustrates the best performance, which is

similar to the results by previous work [6, 11]. In addition, it seems that it is not obvious for lastFM dataset to add layers, but it also has certain effects.

6 Conclusion

In this work, we propose a new collaborative filtering framework. We are the first to mine the relationship between users and items from a multi-task perspective, combining user explicit and implicit feedback data. In our proposed framework, we make full use of both explicit ratings and implicit feedback in two ways and design a neural network to cross the explicit and implicit representations. We conducted extensive experiments on five real-world datasets and demonstrated the effectiveness of our MTCF method. This work responds to some recent questions about using deep neural networks to complete recommendation tasks.

As for future work, we will look for some datasets with a large difference between explicit feedback and implicit feedback to conduct further experiments and improve our framework. Because the user's behavioral preferences change with time, in our work, we simply use the interaction of the last week to reflect the user's recent performance is not comprehensive. So, we will study time-aware models to capture the evolution of user preference.

Funding This research was funded by Natural Science Foundation of China, Grant nos [61962038, 61962038] and Guangxi Bagui Teams for Innovation and Research, Grant no [2019].

References

1. Ebesu T, Shen B, Fang Y (2018) Collaborative memory network for recommendation systems. In: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 515–524
2. Liang D, Krishnan RG, Hoffman MD, Jebara T (2018) Variational autoencoders for collaborative filtering. In: Proceedings of the 2018 World Wide Web Conference, pp 689–698
3. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp 285–295
4. Zhang H, Shen F, Liu W, He X, Luan H, Chua T-S (2016) Discrete collaborative filtering. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp 325–334
5. He X, Zhang H, Kan M-Y, Chua T-S (2016) Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp 549–558
6. Hong-Jian X, Xinyu D, Jianbing Z, Shujian H, Jiajun C (2017) Deep matrix factorization models for recommender systems. IJCAI 17:3203–3209
7. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 770–778
8. Serban I, Sordoni A, Bengio Y, Courville A, Pineau J (2016) Building end-to-end dialogue systems using generative hierarchical neural network models. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016). AAAI Press, pp 3776–3783
9. Huang F, Li X, Yuan C, Zhang S, Zhang J, Qiao S (2021) Attention-emotion-enhanced convolutional lstm for sentiment analysis. IEEE Trans Neural Netw Learn Syst:1–14
10. Hornik K, Stinchcombe M, White H et al (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
11. He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S (2017) Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp 173–182
12. Rendle S, Krichene W, Zhang L, Anderson J (2020) Neural collaborative filtering vs. matrix factorization revisited, pp 240–248
13. Van den Oord A, Dieleman S, Schrauwen B (2013) Deep content-based music recommendation. Advances in neural information processing systems. Springer, Berlin, pp 2643–2651
14. Wang H, Wang N, Yeung D-Y (2015) Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1235–1244
15. Zhang F, Yuan NJ, Lian D, Xie X, Ma W-Y (2016) Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 353–362
16. Alter O, Brown PO, Botstein D (2000) Singular value decomposition for genome-wide expression data processing and modeling. Proc Natl Acad Sci 97(18):10101–10106
17. Lee DD, Sebastian Seung H (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791
18. Salakhutdinov R, Mnih A, Hinton G (2007) Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th International Conference on Machine Learning, pp 791–798
19. Georgiev K, Nakov P (2013) A non-iid framework for collaborative filtering with restricted boltzmann machines. In: International Conference on Machine Learning, pp 1148–1156
20. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 426–434
21. Bai T, Wen J-R, Zhang J, Zhao WX (2017) A neural collaborative filtering model with interaction-based neighborhood. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp 1979–1982
22. Zhang S, Tay Y, Yao L, Sun A (2018) Next item recommendation with self-attention. CoRR. arXiv:1808.06414
23. Chen S, Peng Y (2018) Matrix factorization for recommendation with explicit and implicit feedback. Knowl-Based Syst 158:109–117
24. Liu NN, Xiang EW, Zhao M, Yang Q (2010) Unifying explicit and implicit feedback for collaborative filtering. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp 1445–1448
25. Shi Y, Karatzoglou A, Baltrunas L, Larson M, Hanjalic A (2013) xclimf: optimizing expected reciprocal rank for data with multiple levels of relevance. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp 431–434

26. Bell RM, Koren Y (2007) Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), pp 43–52. IEEE
27. Pan W, Liu Z, Ming Z, Zhong H, Wang X, Congfu X (2015) Compressed knowledge transfer via factorization machine for heterogeneous collaborative recommendation. Knowl-Based Syst 85:234–244
28. Li G, Chen Q (2016) Exploiting explicit and implicit feedback for personalized ranking. Math Probl Eng 2016(11):2535329. https://doi.org/10.1155/2016/2535329
29. Chen BY, Huang L, Wang CD, Jing LP (2020) Explicit and implicit feedback based collaborative filtering algorithm. J Softw 3:794–805
30. Caruana R (1997) Multitask learning. Mach Learn 28(1):41–75
31. Xie Z, Cao W, Ming Z (2021) A further study on biologically inspired feature enhancement in zero-shot learning. Int J Mach Learn Cybern 12(1):257–269
32. Long L, Yin Y, Huang F (2021) Graph-aware collaborative filtering for top-n recommendation. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp 1–8
33. Hadash G, Sar SO, Osadchy R (2018) Rank and rate: multi-task learning for recommender systems. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp 451–454
34. Lu Y, Dong R, Smyth B (2018) Why i like it: multi-task learning for recommendation and explanation. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp 4–12
35. Ma X, Zhao L, Huang G, Wang Z, Hu Z, Zhu X, Gai K (2018) Entire space multi-task model: an effective approach for estimating post-click conversion rate. In: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 1137–1140
36. Ni Y, Ou D, Liu S, Li X, Ou W, Zeng A, Si L (2018) Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 596–605
37. Zhao Z, Hong L, Wei L, Chen J, Nath A, Andrews S, Kumthekar A, Sathiamoorthy M, Yi X, Chi E (2019) Recommending what video to watch next: a multitask ranking system. In: Proceedings of the 13th ACM Conference on Recommender Systems, pp 43–51
38. Deng Z-H, Huang L, Wang C-D, Lai J-H, Yu PS (2019) Deepcf: a unified framework of representation learning and matching function learning in recommender system. Proc AAAI Conf Artif Intell 33:61–68
39. Mnih A, Salakhutdinov RR (2008) Probabilistic matrix factorization. Advances in neural information processing systems. Springer, Berlin, pp 1257–1264
40. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
41. Salakhutdinov R, Mnih A (2008) Bayesian probabilistic matrix factorization using markov chain monte carlo. In: Proceedings of the 25th International Conference on Machine learning, pp 880–887
42. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR). arXiv:1412.6980
43. Wang H, Zhang F, Zhao M, Li W, Xie X, Guo M (2019) Multi-task feature learning for knowledge graph enhanced recommendation. In: The World Wide Web Conference, pp 2000–2010
44. Xin X, He X, Zhang Y, Zhang Y, Jose J (2019) Relational collaborative filtering: Modeling multiple item relations for recommendation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 125–134
45. Bayer I, He X, Kanagal B, Rendle S (2017) A generic coordinate descent framework for learning from implicit feedback. In: Proceedings of the 26th International Conference on World Wide Web, pp 1341–1350
46. He X, Chen T, Kan M-Y, Chen X (2015) Trirank: Review-aware explainable recommendation by modeling aspects. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp 1661–1670
47. Tan YK, Xu X, Liu Y (2016) Improved recurrent neural networks for session-based recommendations. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp 17–22
48. Cao W, Wang X, Ming Z, Gao J (2018) A review on neural networks with random weights. Neurocomputing 275:278–287
49. Cao W, Hu L, Gao J, Wang X, Ming Z (2020) A study on the relationship between the rank of input data and the performance of random weight neural network. Neural Comput Appl:1–12
50. Wang X, Cao W (2018) Non-iterative approaches in training feed-forward neural networks and their applications. Soft computing-a fusion of foundations, methodologies and applications 22(11):3473–3476
51. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp 452–461
52. Socher R, Chen D, Manning CD, Ng A (2013) Reasoning with neural tensor networks for knowledge base completion. Advances in neural information processing systems. Springer, Berlin, pp 926–934