Learning to Recommend With Multiple Cascading Behaviors

Chen Gao, Xiangnan He, *Member, IEEE*, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, *Senior Member, IEEE*, Tat-Seng Chua, Lina Yao, Yang Song, and Depeng Jin, *Member, IEEE*

Abstract—Most existing recommender systems leverage user behavior data of one type only, such as the purchase behavior in E-commerce that is directly related to the business Key Performance Indicator (KPI) of conversion rate. Besides the key behavioral data, we argue that other forms of user behaviors also provide valuable signal, such as views, clicks, adding a product to shopping carts and so on. They should be taken into account properly to provide quality recommendation for users. In this work, we contribute a new solution named short for **N**eural **M**ulti-**T**ask **R**ecommendation (NMTR) for learning recommender systems from user multi-behavior data. We develop a neural network model to capture the complicated and multi-type interactions between users and items. In particular, our model accounts for the cascading relationship among different types of behaviors (e.g., a user must click on a product before purchasing it). To fully exploit the signal in the data of multiple types of behaviors, we perform a joint optimization based on the multi-task learning framework, where the optimization on a behavior is treated as a task. Extensive experiments on two real-world datasets demonstrate that NMTR significantly outperforms state-of-the-art recommender systems that are designed to learn from both single-behavior data and multi-behavior data. Further analysis shows that modeling multiple behaviors is particularly useful for providing recommendation for sparse users that have very few interactions.

Index Terms—Multi-behavior recommendation, collaborative filtering, deep learning

1 Introduction

In online information systems, users interact with a system in a variety of forms. For example, in an E-commerce website, a user can click on a product, add a product to shopping cart, purchase a product and so on.

In traditional recommender systems, only user-item interaction data of one behavior type is considered for collaborative filtering, such as the purchase behavior in E-commerce and the rating behavior on movies [1], [2]. While it is particularly useful to optimize a recommender model on the data that is directly related to the business KPI, the other forms of behaviors should not be neglected, since they also provide valuable signal on a user's preference.

- *C. Gao, X. Chen, Y. Li, and D. Jin are with the Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. E-mail: {gc16, cxn15}@mails.tsinghua.edu.cn, {liyong07, jindp}@tsinghua.edu.cn.*
- *X. He is with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230052, China. E-mail: xiangnanhe@gmail.com.*
- *D. Gan is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, 15213 PA USA. E-mail: dgan@andrew.cmu.edu.*
- *F. Feng and T.-S. Chua are with the School of Computing, National University of Singapore, Singapore 117417, Singapore. E-mail: fulifeng93@gmail.com, dcscts@nus.edu.sg.*
- *L. Yao and Y. Song are with the School of Computer Science and Engineering, the University of New South Wales (UNSW), Sydney, NSW 2052, Australia. E-mail: {lina.yao, yang.song1}@unsw.edu.au.*

Manuscript received 8 Dec. 2018; revised 12 Sept. 2019; accepted 12 Nov. 2019. Date of publication 10 Dec. 2019; date of current version 29 Apr. 2021. (Corresponding author: Yong Li.)
Recommended for acceptance by Q. He.
Digital Object Identifier no. 10.1109/TKDE.2019.2958808

Existing approaches for multi-behavior recommendation can be divided into two categories. The first category is based on collective matrix factorization (CMF) [3], [4], [5], [6], which extends the matrix factorization (MF) method to jointly factorize multiple behavior matrices. In MF, a user (or an item) is described as an embedding vector to encode her preference (or its property), and a user-item interaction is estimated as the inner product of the user embedding and item embedding. To correlate MF on multiple behavior matrices, it is essential to share the embedding matrix of entities of one side (e.g., items), and let the entities of the other side (e.g., users) learn different embedding matrices for different types of behaviors.

The second category approaches the problem from the perspective of learning [7], [8]. To learn recommender models from the (implicit) data of interactions, it is natural to assume that a user's interacted items should be more preferable over the non-interacted items. Bayesian Personalized Ranking (BPR) [1] is a representative method that implements the assumption of relative preference; it is then extended to address multi-behavior recommendation [7] by enriching the training data of relative preference from the multi-behavior data.

Despite effectiveness, we argue that existing models for multi-behavior recommendation suffer from three limitations.

- *Lack of behavior semantics*. Each behavior type has its own semantics and contexts, and more importantly, there exist strong ordinal relations among different behavior types. For example, the behaviors may represent the action sequence of a user on a product: click is not likely to happen after add-to-cart; add-to-cart behavior is not likely to happen after a purchase. Moreover, the semantics make some intermediate feedback rather meaningful, such as the products that are viewed but not purchased. However, existing models have largely ignored the semantics of different behavior types.

- *Unreasonable embedding learning*. The CMF paradigm needs to enforce the entities of one side (either users or items) have different embedding matrices for different types of behaviors. From the perspective of representation learning and interpretation of latent factor models [9], [10], this setting is unreasonable. Specifically, a user's embedding vector represents his/her inherent interests and multiple behaviors with one item always happen in a short period. Therefore a user's embedding vector should remain unchanged when the user performs different types of behaviors on one item; and similarly for the item side. Only the interaction function [2] should be changed when predicting a user's different types of behaviors on an item.
- *Incapability in modeling complicated interactions*. Existing methods largely rely on MF to estimate a user's preference on an item. In MF, the interaction function is a fixed inner product, which is insufficient to model the complicated and multi-type interactions between users and items. This is also a major reason why these CMF methods need to enforce entities of one side to have different embedding matrices for predicting different types of behaviors; otherwise, the model could not make distinct predictions for different behavior types.

To address the above mentioned limitations in multi-behavior recommendation, we propose a new solution named **N**eural **M**ulti-**T**ask **R**ecommendation (NMTR). Briefly, our method combines the recent advance of neural collaborative filtering with multi-task learning to effectively learn from multiple types of user behaviors. Specifically, we separate the two components of embedding learning and interaction as advocated by the neural collbaborative filtering (NCF) [2] framework. We then design that 1) a user (and an item) has a shared embedding across multiple types of behaviors, and 2) a data-dependent interaction function is learned for each behavior type. Through this way, we address the inherent limitations of CMF methods and make the model more suitable for learning from behaviors of multiple types.

Moreover, to incorporate the behavior semantics, especially the ordinal relation among behavior types, we relate the model prediction of each behavior type in a cascaded manner. To be specific, assuming we have two form of behaviors, view and purchase, which form a natural ordinal relation: view $\rightarrow$ purchase. We enforce that the prediction of a high-level behavior (i.e., purchase) comes from the prediction of the low-level behavior (i.e., view). Through this way, we can capture the underlying semantics that a user must view a product in order to purchase it.

To summarize, the main contributions of this work are as follows.

- We propose a novel neural network model tailored to learning user preference from multi-behavior data. The model shares the embedding layer for different behavior types, and learns separate interaction function for each behavior type.
- To capture the ordinal relations among behavior types, we propose to correlate the model prediction of each behavior type in a cascaded way. Furthermore, we train the whole model in a multi-task manner to make full use of multiple types of behaviors.
- To demonstrate the effectiveness of our proposal, we implement three variants of NMTR using different neural collaborative filtering models as the interaction function. Extensive experiments on two real-world datasets show that our method outperform best existing methods by 6.08 and 30.76 percent on the hit-ratio effect for two datasets, respectively. Further studies demonstrate the effectiveness of the multi-task learning manner.

The remainder of the paper is as follows. We first formalize the problem and introduce some preliminaries in Section 2. We then present our proposed method in Section 3. We conduct experiments in Section 4, before reviewing related work in Section 5 and concluding the paper in Section 6.

2 Preliminaries

We first formulate the problem to solve in this paper. Then we recapitulate the neural collaborative filtering technique [2]. Lastly, we introduce collective matrix factorization, a prevalent solution for multi-behavior recommendation.

2.1 Problem Formulation

In recommender systems, there typically exists a key type of user behaviors to be optimized, which we term it as the *target behavior*. For example, in an E-commerce site, the target behavior is usually purchase, since it is directly related with the conversion rate of recommendation and is the strongest signal to reflect a user's preference. Traditional collaborative filtering techniques [1], [11] focus on the target behavior only and forgo other types of user behaviors such as views, clicks, etc., which are readily available in the server logs. The focus of this work is to leverage these other types of user behaviors to improve the recommendation for the target behavior.

Let $\{\mathbf{Y}^1, \mathbf{Y}^2, \ldots, \mathbf{Y}^R\}$ denote the user-item interaction matrices for all the R types of behaviors. Each interaction matrix is of size $M \times N$, where M and N denote the number of users and items, respectively. Since in real-world applications, most user feedback are in the implicit form [1], [12], we assume that each entry of a interaction matrix has a value of 1 or 0:

$$y_{ui}^r = \begin{cases} 1, & \text{if } u \text{ has interacted with } i \text{ under behavior } r; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

As we have discussed in the introduction, many user behavior types in real-world applications follow an ordinal (or sequential) relationship. Without loss of generality, we assume that the behavior types have a total order and sort them from the lowest level to the highest level: $\mathbf{Y}^1 \rightarrow \mathbf{Y}^2 \ldots \rightarrow \mathbf{Y}^R$, where $\mathbf{Y}^R$ denotes the target behavior to be optimized. Since the target behavior typically concerns the conversion rate, we regard it as having highest priority.

The problem of multi-behavior recommendation is then formulated as follows.

Input: The user-item interaction data of the target behavior $\mathbf{Y}^R$, and the interaction data of other behavior types $\{\mathbf{Y}^1, \mathbf{Y}^2, \ldots, \mathbf{Y}^{R-1}\}$.

Output: A model that estimates the likelihood that a user u will interact with an item i under the target behavior.

After obtaining the predictive model, we can use it to score all items for a user u, and select the top-ranked items as the recommendation results for u.

2.2 Neural Collaborative Filtering (NCF)

NCF is generic neural network framework for performing collaborative filtering (CF) on single-behavior data [2]. It applies a representation learning view [13] for CF, representing each user (and item) as an embedding vector. To predict a user's preference on an item, it feeds their embeddings into a neural network

$$\hat{y}_{ui} = f_\Theta(\mathbf{p}_u, \mathbf{q}_i|\Theta), \tag{2}$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ denote the embedding vector for user u and item i, respectively; f_Θ denotes the neural network with parameters Θ, which is also called as the *interaction function*, since it is responsible for learning the interaction between user embedding and item embedding to obtain the prediction score. The model parameters of NCF can be learned in an end-to-end fashion. Specifically, the authors opt to optimize a pointwise log loss, where the positive instances are the entries of value 1 (aka., observed entries) in the user-item interaction matrix $\mathbf{Y}^R$ and the negative instances are randomly sampled from the entries of value 0 (aka., missing data).

The matrix factorization model can be seen as the special case of NCF—by specifying the interaction function f_Θ as an inner product, NCF exactly recovers MF. As such, under the NCF framework, MF can be interpreted as using a fixed, data-independent interaction function. As demonstrated in the NCF paper and its follow-up work [14], using such a fixed interaction function is suboptimal and can be improved by learning the interaction function from data. It is this evidence that motivates us to develop neural network models to address the multi-behavior recommendation task.

In the NCF paper, the authors present three instantiations of NCF, namely, GMF, MLP and NeuMF. Briefly, GMF generalizes MF by defining f_Θ as an element-wise product layer followed by a weighted output layer. MLP employes multi-layer perceptron above the concatenation of $\mathbf{p}_u$ and $\mathbf{q}_i$ to learn the interaction function. The best performance is achieved by NeuMF, which concatenates the element-wise product layer of GMF and the last hidden layer of MLP, feeding it to a weighted output layer to obtain the prediction score. Our NMTR uses NCF as a building block, and as such, any design of f_Θ can be used as a component to learn the interaction function for one behavior type in our method.

2.3 Collective Matrix Factorization

CMF is originally proposed to factorize multiple data matrices that have certain common entities [3]. For example, it can be used to factorize user-movie and movie-genre matrix, where movies are the common entities of the two data matrices.

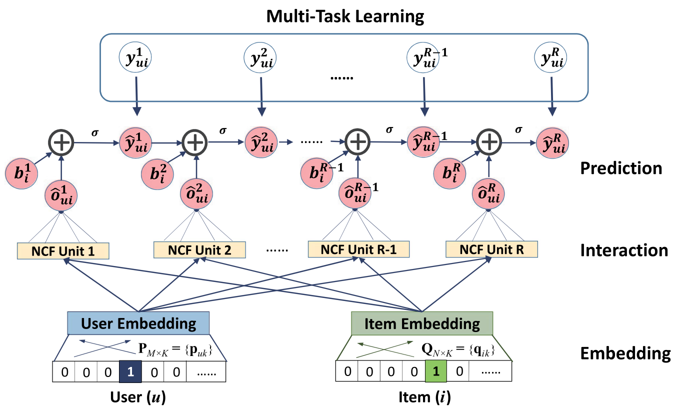


Fig. 1. Illustration of our proposed NMTR model.

The idea is to correlate the multiple factorization processes by sharing the embeddings of common entities.

Nevertheless, in multi-behavior recommendation, both sides of entities are shared in data matrices of different behavior types. Directly applying CMF will fail to produce different predictions for different behavior types. To address this problem, Zhao *et al.* [6] proposed to share the item embedding matrix for all behavior types, allowing a user to learn different embedding vectors for different behavior types. To be specific, the objective function to optimize is as follows:

$$\min_{\mathbf{p}_*, \mathbf{q}_*} \sum_{r=1}^{R} \sum_{u=1}^{M} \sum_{i=1}^{N} c^r_{ui}(y^r_{ui} - {\mathbf{p}^r_u}^T \mathbf{q}_i)^2, \tag{3}$$

where c^r_{ui} denotes the importance of the entry y^r_{ui} in factorization, $\mathbf{q}_i$ denotes the embedding vector for item i that is shared by all behavior types, and $\mathbf{p}^r_u$ denotes the embedding vector for user i in reconstructing the behaviors of the rth type. Note that we have omitted the L_2 regularization term for clarity.

As argued earlier in the introduction, this setting is irrational and non-interpretable as a latent factor model. Specifically, an embedding vector for a user encodes his/her latent interest, which should remain unchanged when the user seeks items of interest to consume at a particular time. Moreover, other potential limitations of existing CMF methods include the use of a fixed interaction function of inner product, and the use of squared regression loss for optimization, which may be suboptimal for item recommendation with implicit feedback [1], [2].

3 Methods

Fig. 1 illustrates our proposed NMTR model. Given a user-item pair (u, i) as the input, the model aims to predict the likelihood that u will perform a behavior (of any of the R types) on item i, represented as the output of $\{\hat{y}^1_{ui}, \hat{y}^2_{ui}, \ldots, \hat{y}^R_{ui}\}$.

Our NMTR method is featured with four special designs:

- *Shared embedding layer*. To make it reasonable under the paradigm of representation learning, we share the embedding layer of users and items for the modeling of all behavior types.
- *Separated interaction function*. We learn different interaction functions for predicting the behaviors

of different types. This is achieved by using the expressive NCF unit for each type of behaviors.

- *Cascaded predictions*. To capture the ordinal relations among behavior types, we correlate the predictions of different behavior types through cascading.
- *Multi-task learning*. To optimize the cascaded architecture, we simultaneously train the predictive models for all behavior types by performing multi-task learning.

In what follows, we present our method by elaborating the above four designs.

3.1 Shared Embedding Layer

In order to make our proposed model extensible, we apply one-hot encoding to encode the input of user ID and item ID. One advantage is that it can be easily extended to incorporate other features of a user and an item (e.g., user demographics and item attributes), if they are available in the application [12]. Let $\mathbf{v}_u^U$ and $\mathbf{v}_i^I$ denote the one-hot feature vector for user u and item i. Then the embedding layer is defined as a linear fully connected layer without the bias terms

$$\mathbf{p}_u = \mathbf{P}^T \mathbf{v}_u^U, \quad \mathbf{q}_i = \mathbf{Q}^T \mathbf{v}_i^I, \tag{4}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are the user embedding matrix and item embedding matrix, respectively. When only the ID feature is used to describe a user (or an item), $\mathbf{P}$ and $\mathbf{Q}$ are of the size $M \times E$ and $N \times E$, respectively, where E denotes the embedding size; and $\mathbf{p}_u$ and $\mathbf{q}_i$ are essentially the $u-$th and $i-$th row vector of $\mathbf{P}$ and $\mathbf{Q}$, respectively.

It is worth noting that NMTR has only one embedding layer in the lower part of the model, which is to be used for the prediction of all behavior types in the upper part. Although there are some works modeling dynamic interests [15], [16], [17] with dynamic user embeddings, in our task it is better to use static embeddings for users. In fact, different with these works studying user dynamic intention in a long period, we focuses on modeling users multiple types of interactions on one item, which always happen in a relatively shorter time period. Based on this design, we can interpret the model under the paradigm of representation learning, where $\mathbf{p}_u$ and $\mathbf{q}_i$ are the latent features to be learned to represent user u and item i, respectively.

3.2 Separated Interaction Function

Above the embedding layer is the hidden layers that model the interaction between $\mathbf{p}_u$ and $\mathbf{q}_i$ to obtain the prediction score. Since we need to predict the likelihood of multiple behavior types with the same input, it is essential to learn a separated interaction function for each type. Let f_Θ^r denote the interaction function for the rth type of behaviors with parameters Θ, which outputs the likelihood that u will perform a behavior of the rth type

$$\hat{y}_{ui}^r = \sigma(f_\Theta^r(\mathbf{p}_u, \mathbf{q}_i)), \tag{5}$$

where σ denotes the sigmoid function converting the output to a probability. A good design of f_Θ^r is to have the ability and sufficient flexibility to learn the possible complicated patterns (e.g., collaborative filtering and others) in user behaviors. To achieve this, we consider the three neural network units proposed in the NCF paper [2]:

- *GMF* generalizes MF by allowing different dimensions of the embedding space to have different weights. To be specific, it first uses an element-wise product to get an interacted vector, and then project the vector to an output score with a weight vector
$$f_\Theta^{GMF}(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i), \tag{6}$$
where $\mathbf{h} \in \mathbb{R}^{E\times 1}$ denotes the learnable weight vector. The parameters of the GMF unit are $\Theta_{GMF} = \{\mathbf{h}\}$.
- *MLP* applies a multi-layer perceptron on the concatenation of $\mathbf{p}_u$ and $\mathbf{q}_i$ to learn the interaction function in a hierarchical and non-linear manner
$$\begin{aligned} \mathbf{z}_1 &= ReLU(\mathbf{W}_1 \begin{bmatrix} \mathbf{p_u} \\ \mathbf{q}_i \end{bmatrix} + \mathbf{b}_1), \\ &\cdots\cdots \\ \mathbf{z}_L &= ReLU(\mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L), \\ f_\Theta^{MLP}(\mathbf{p}_u, \mathbf{q}_i) &= \mathbf{h}^T \mathbf{z}_L, \end{aligned} \tag{7}$$
where L denotes the number of hidden layers in the multi-layer perceptron, $\mathbf{W}_x$ and $\mathbf{b}_x$ denote the weight matrix and bias vector for the xth hidden layer, and $\mathbf{z}_x$ are the intermediate neurons. By default, the rectifier unit (ReLU) is used as the activation function for the hidden layer, which is beneficial to build deep models. The parameters of the MLP unit are $\Theta_{MLP} = \{\mathbf{h}, \{\mathbf{W}_x\}_{x=1}^L, \{\mathbf{b}_x\}_{b=1}^L\}$.
- *NeuMF* combines the advantage of the linear GMF with the nonlinear MLP to learn the interaction function
$$f_\Theta^{NeuMF}(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{h}^T \begin{bmatrix} \mathbf{p_u} \odot \mathbf{q_i} \\ \mathbf{z}_L \end{bmatrix}, \tag{8}$$
where $\mathbf{z}_L$ indicates the last hidden layer of MLP, as have been defined in Equation (7). The $\mathbf{z}_L$ is concatenated with $\mathbf{p}_u \odot \mathbf{q}_i$ as the hidden layer of NeuMF, which is then projected to a score through the weighted vector $\mathbf{h} \in \mathbb{R}^{2E\times 1}$. In the original design of NeuMF, the authors used different embedding layers for GMF and MLP. While in our method, we have only one set of embeddings for users and items. As such, we tweak the NeuMF unit by sharing the embedding layer of GMF and MLP.

Note that any of the three units can be used to model for behaviors of any type, and the optimal setting may depend on the dataset. We will empirically evaluate the performance of three NCF choices and their impact on our NMTR method in Section 4. There are many other possible designs for the NCF units, such as placing more layers above the hidden layer of NeuMF to thoroughly merge GML and MLP, among others [12], [14]. Since the focus of this paper is not to develop new NCF units for interaction learning, we leverage existing ones as the building block for our NMTR model.

3.3 Cascaded Predictions

Typically there are certain ordinal relations among behavior types in a real-world application, such as a user must view a product (i.e., click the product page) before she can purchase it. The existence of such relations implies that the predictive models for different behavior types should be related with each other, rather than being independent. To encode the sequential effect, we enforce that the prediction on a behavior type lies in the predictions of the precedent behavior types. Formally, we cascade the predictions of different behaviors as

$$\begin{aligned}
\hat{y}^R_{ui} &= \sigma(\hat{y}^{R-1}_{ui} + f^R_\Theta(\mathbf{p}_u, \mathbf{q}_i) + b^R_i),\\
&\cdots\cdots\\
\hat{y}^2_{ui} &= \sigma(\hat{y}^1_{ui} + f^2_\Theta(\mathbf{p}_u, \mathbf{q}_i) + b^2_i),\\
\hat{y}^1_{ui} &= \sigma(f^1_\Theta(\mathbf{p}_u, \mathbf{q}_i) + b^1_i),
\end{aligned} \tag{9}$$

where b^r_i denotes the bias of item i in the data of the rth behavior type, and f^r_Θ denotes the interaction function for the rth type of behaviors, which can be any of the three NCF units as introduced before. The item bias term can capture some discrepancy effects in different types of behaviors, for example, some items are likely to be clicked by users (e.g., products on campaign) but less likely to be purchased. Moreover, some previous work has demonstrated that incorporating item bias is more effective than incorporating user bias for learning from single-behavior implicit feedback [11].

A graphical illustration of our cascading design[1] can be found in the top part of Fig. 1. Such a design is particularly useful for predicting the preference of inactive users that have few data on the target behavior. Typically, the data of low-level behaviors (e.g., clicks) is easier to collect and has a larger volume than the target behavior (e.g., purchases). By basing the prediction of target behavior on its precedent types of behaviors, we can achieve better prediction when the target behavior data of a user is insufficient to estimate f^R_Θ well.

3.4 Multi-Task Learning

As we have a dedicated model for each type of behaviors and the models follow a cascading prediction, it is intuitive to train models separately by following the order of $\hat{y}^1_{ui}, \hat{y}^2_{ui}, \ldots, \hat{y}^R_{ui}$. Since these models share the same embedding layer and the final recommendation is based on the last target model $\hat{y}^R_{ui}$, this way can be seen as pre-training the embedding layer of the target model using other types of behaviors. We argue that such a sequential training manner does not make full use of the multi-behavior data, since it only uses precedent models to improve the next model while there is no benefit for the precedent models. A better solution could be to let the models reinforce each other.

In contrast to training the models separately, multi-task learning (MTL) is a paradigm that performs joint training on the models of different but correlated tasks, so as to obtain a better model for each task [18]. The intuition for our design of cascaded predictions is that, if we can obtain improved models for other types of behavior, the model for the target behavior can also be improved. As such, we opt for MTL that trains all models simultaneously, where the model learning for each behavior type is treated as a task.

Objective Function. Following the probabilistic optimization framework [2], we first define the likelihood function for a single behavior type as

$$P_r = \prod_{(u,i)\in\mathcal{Y}^+_r} \hat{y}^r_{ui} \prod_{(u,i)\in\mathcal{Y}^-_r} (1 - \hat{y}^r_{ui}), \tag{10}$$

where $\mathcal{Y}^+_r$ denotes the set of observed interactions in behavior matrix $\mathbf{Y}^r$, and $\mathcal{Y}^-_r$ denotes negative instances to be sampled from the unobserved interactions in $\mathbf{Y}^r$. We then get the joint probability for multiple types of behaviors as

$$P = \prod_{r=1}^{R} P_r = \prod_{r=1}^{R} \prod_{(u,i)\in\mathcal{Y}^+_r} \hat{y}^r_{ui} \prod_{(u,i)\in\mathcal{Y}^-_r} (1 - \hat{y}^r_{ui}). \tag{11}$$

Taking the negative logarithm of the joint probability, we obtain the loss function to be minimized as

$$L = -\sum_{r=1}^{R} \lambda_r \left(\sum_{(u,i)\in\mathcal{Y}^+_r} \log \hat{y}^r_{ui} + \sum_{(u,i)\in\mathcal{Y}^-_r} \log (1 - \hat{y}^r_{ui}) \right), \tag{12}$$

where we additionally include the term λ_r to control the influence of the rth type of behaviors on the joint training. This is a hyper-parameter to be specified for different datasets, since the importance of a behavior type may vary for problems of different domains and scales. We additionally enforce that $\sum_{r=1}^{R} \lambda_r = 1$ to facilitate the tuning of these hyper-parameters.

Directly optimizing this joint loss function will update the parameters of models for multiple behavior types together. As such, a better embedding learned from a gradient step of the data of one type will benefit the learning of other types.

Training. Since our model is composed of nonlinear neural networks, we optimize parameters with stochastic gradient descent (SGD), a generic solver for neural network models. As most machine learning toolkits (e.g., TensorFlow, Theano, PyTorch etc.) provide the function of automatic differentiation, we omit the derivation of the derivatives of our model. Instead, we elaborate on how to form a mini-batch to facilitate faster training, since modern computing units like GPU and CPU provide acceleration for matrix-wise float operations.

To generate a mini-batch, we first sample a user-item pair (u, i) such that user u has at least one observed interaction on item i (regardless of the behavior type). We then inspect the interactions of the (u, i) pair—for each observed interaction, we sample a negative instance from $u's$ unobserved interactions of the behavior type. As an example, if the sampled (u, i) pair has an interaction in the 1st behavior and 2nd behavior, we get two positive training instances y^1_{ui} and y^2_{ui}; we then sample two items t and s that u did not interact under first two behaviors, respectively, to get two negative instances y^1_{ut} and y^2_{us}. We iterate the above sampling step until the desired size of a mini-batch is reached.

Note that we empirically find that sampling multiple negative instances to pair with a positive instance in a mini-batch can improve the performance. This finding has been reported before in optimizing neural recommender models with log loss on single-behavior data [2], [19]. As such, in our experiments, we allow a flexible tuning of the negative sampling ratio.

1. Note that we assume that the behaviors can form a full-order cascading relationship, while in real world the relationship might be more complicated. For example, there is no sequential relation between sharing a product to social network and adding it to cart by nature. Technically speaking, we can adapt to such partial-order relation by sorting the behaviors by their strength in reflecting user preference. We leave this exploration as future work.

TABLE 1
Statistics of our Evaluation Datasets

Dataset	User#	Item#	Purchase#	Cart#	View#
Beibei	21,716	7,977	295,622	642,622	2,412,586
Tmall	15,670	9,076	136,648	–	813,396

4 Experiments

In this section, we conduct extensive experiments on two real-world datasets to answer the following research questions:

- *RQ1:* How does our proposed NMTR perform as compared with state-of-the-art recommender systems that are designed for learning from single-behavior and multi-behavior data?
- *RQ2:* How do the key hyper-parameters affect NMTR's performance, and how is the effectiveness of our designed multi-task learning for the task?
- *RQ3:* Can NMTR help to address the data sparsity problem, i.e., improving recommendations for sparse users with fewer interactions of the target behavior?

In what follows, we first describe the experimental settings, and then answer the above three research questions.

4.1 Experimental Settings

4.1.1 Datasets and Evaluation Protocol

We experimented with two real-world E-commerce datasets that contain multiple types of user behaviors including purchases, views, adding to carts, etc.

- *Beibei Dataset*[2]. This dataset is collected from Beibei, the largest E-commerce platform for maternal and infant products in China. We sampled a subset of user interactions that contain views, adding to carts (abbreviated as *carts*), and purchases within the time period from 2017/06/01 to 2017/06/30.
- *Tmall Dataset*[3]. This is the dataset released in IJCAI-15 challenge,[4] which is collected from Tmall, the largest business-to-consumer E-Commerce website in China. It records two types of user behaviors, views and purchases, within the time period from 2014/05/01 to 2014/11/30.

For both datasets, we merged the duplicated user-item interactions by keeping the earliest one; the rationality here is to test the performance of a method in recommending novel items that a user did not consume before. Moreover, we focused on users with more than one type of behavior. After the above pre-processing steps, we obtained the final evaluation datasets, the statistics of which are summarized in Table 1. For these two datasets, there exist strict cascading relationships. For example, in Beibei dataset, a user must click first before adding to cart, and must add to cart first before purchasing. In the evaluation stage, given a user in the testing set, each algorithm ranks all items that the user has not interacted before. We applied the widely used leave-one-out technique to obtain the training set and test set, which means for every user, there is a test item her has not interacted with. We then adopted two popular metrics, *HR* and *NDCG*, to judge the performance of the ranking list:

- *HR@K: Hit Ratio* (HR) measures whether the test item is contained by the top-K item ranking list (1 for yes and 0 for no).
- *NDCG@K: Normalized Discounted Cumulative Gain* (NDCG) complements HR by assigning higher scores to the hits at higher positions of the ranking list.

4.1.2 Baselines

We compared the performance of our proposed NMTR with 9 baselines, which can be divided into two groups based on whether it models single-behavior or multi-behavior data. The compared single-behavior methods are introduced as follows.

BPR [1] *Bayesian Personalized Ranking* is a widely used pairwise learning framework for item recommendation with implicit feedback. Same as the original paper, we used BPR to optimize the MF model.

NCF [2] *Neural Collaborative Filtering* is a neural framework to learn interactions between the latent features of users and items. As we employed three NCF methods, named *GMF*, *MLP* and *NeuMF* to learn the interaction function for each behavior type, we evaluated how the three methods perform for single-behavior data. The second group of five compared methods that can leverage multiple types of behavior data are as follows.

CMF [6] As have described in Section 2.3, CMF decomposes the data matrices of multiple behavior types simultaneously. We adapted the method by sharing the user embeddings for factorizing different interaction matrices of various types of behaviors. As our datasets are implicit feedback, we further augmented the method by sampling negative instances in the same way as our NMTR.

MC-BPR [7] Multi-Channel BPR [7] is the state-of-the-art solution for multi-behavior recommendation. It adapts the negative sampling rule in BPR to account for the levels of user feedback in multi-behavior data. For example on the Tmall dataset that has two behavior types—purchase and view, to generate a negative sample for a purchase interaction, it assigns different probabilities for sampling from 1) items that are viewed but not purchased, and 2) items that are not viewed. We tuned the probability distribution for sampling and reported the best results.

MC-NCF Since Multi-Channel BPR is a generic learning method that is applicable to any differentiable recommender model, we replaced the basic MF model in it with state-of-the-art NCF models, and named this extension as MC-NCF. That is, we optimized the three NCF models with the Multi-Channel BPR learner, and named the respective methods as *MC-GMF*, *MC-MLP* and *MC-NeuMF*.

4.1.3 Parameter Settings

We implemented our NMTR[5] and baseline methods in TensorFlow.[6] Since we have three choices of NCF units as the

2. https://www.beibei.com
3. https://www.tmall.com
4. https://tianchi.aliyun.com/datalab/dataSet.htm?id=5
5. We release our implementation along with datasets at https://github.com/fiblab
6. https://www.tensorflow.org

interaction function, we name the respective methods as *NMTR-GMF*, *NMTR-MLP* and *NMTR-NeuMF*. We randomly selected a training instance for each user as the validation set to tune hyper-parameters. For all methods, we set the embedding size to 64, a relatively larger number that achieves good performance on our datasets. For CMF, one important hyper-parameter is the weight of different behavior types in the joint loss. We tuned the weight for each behavior in [0, 0.2, 0.4, 0.6, 0.8, 1]. To be specific, the weight for each behavior represent the influence of each interaction matrix on the collective matrix factorization task. For MC-methods, we carefully tune the sampling distribution following the original paper. For neural network models, we initialized their parameters using the method proposed in [20]. For models that have multiple hidden layers, i.e., MLP, MC-MLP, NMTR-MLP, NeuMF, MC-NeuMF and NMTR-NeuMF, we employed a tower structure for the hidden layers same as [2], and tuned the number of layers from 1 to 5. We set the negative sampling ratio as 4 for all methods, an empirical value that shows good performance. We tried two SGD-based optimizers, Adam [21] and Adagrad [22], and tuned the learning rate for each optimizer in [0.001, 0.005, 0.01, 0.02, 0.05]. Moreover, we applied L_2 regularization to all methods to prevent overfitting.

4.2 Performance Comparison (RQ1)

We first compare the top-K recommendation performance with state-of-the-art methods. We investigate the top-K performance with K setting to [50, 80, 100, 200]. Note that for a user, our evaluation protocol ranks all unobserved items in the training set [11]. Though this all-ranking protocol can be very time-consuming, the obtained results are more persuasive than ranking a random subset of negative times only (e.g., as have done in [2]). In this case, small values of K will make the results have a large variance and unstable. As such, we report results of a relatively large.[7] We report the best parameter setting for our proposed NMTR methods in Table 2.

Table 3 shows the performance of HR@K and NDCG@K for our three NMTR methods, five multi-behavior recommendation methods, and four single-behavior methods. From the results, we have the following observations:

- *NMTR achieves the best performance.* Our proposed NMTR methods obtain the best performance in terms of HR@K and NDCG@K as compared to all baselines. The one-sample paired t-tests indicate that all improvements are statistically significant for $p < 0.05$. Among the three NMTR methods, NMTR-GMF and NMTR-NeuMF are better than NMTR-MLP, which verifies the effectiveness of the element-wise operator in learning the user-item interaction function.

  Compared with the best single-behavior baseline NeuMF, NMTR outperforms it by 9.01 percent in HR and 6.72 percent in NDCG on the Beibei dataset; and the improvements are 13.04 percent in HR and 9.91 percent in NDCG on the Tmall dataset. Compared with MC-NeuMF, which extends NeuMF on multi-behavior data with the Multi-Channel BPR [7], NMTR obtains an improvement in HR of 6.08 and 10.23 percent on Beibei and Tmall, respectively. In addition, we can observe that MF based methods (CMF, MC-BPR and BPR), achieve the worst performance on the Beibei dataset, which has more complicated and richer behaviors than the Tmall dataset. This confirms the incapability of MF in modeling complicated interactions between users and items, being inferior to the multi-layer neural networks.
- *NMTR is a better framework than MC.* For each NCF model, we find that optimizing it under our NMTR framework outperforms optimizing it under the Multi-Channel BPR framework. Specifically, NMTR-NeuMF outperforms MC-NeuMF by 6.08 percent on Beibei dataset and 30.76 percent on Tmall dataset in HR@100. Thus, we can conclude that NMTR performs better than the MC framework in adapting a single-behavior recommender model for multiple behaviors.

To better understand the difference between two frameworks, we present the training loss and the testing performance in each training iteration in Fig. 2 (for Beibei) and Fig. 3 (for Tmall). In our NMTR framework, the training loss is defined as the joint loss in multi-task learning, which is a combination of the prediction loss of behaviors of multiple types. We can observe that, for both datasets, although training loss of NMTR is the highest, it essentially demonstrates the best generalization performance. For the Beibei dataset, we find that the HR score of MC-NeuMF starts to decrease after 40 iterations, even though the L_2 regularization and dropout have been adopted. Note that in Table 3, we have reported the peak performance of each baseline evaluated per iteration (such a setting is to fully explore the potential of all methods). Even so, our NMTR still outperforms MC-NeuMF by 6.08 percent in HR@100 and 5.70 percent in NDCG@100. However, on the Tmall dataset, in which only two behaviors are available and the data is of a smaller scale, MC-NeuMF fails to utilize the view behavior to improve the performance (i.e., underperforms NeuMF). In contrast, our NMTR-NeuMF outperforms NeuMF by 30.76 percent in HR@100 and

TABLE 2
Best Parameter Settings of our Proposed NMTR Methods for Top-K Recommendation

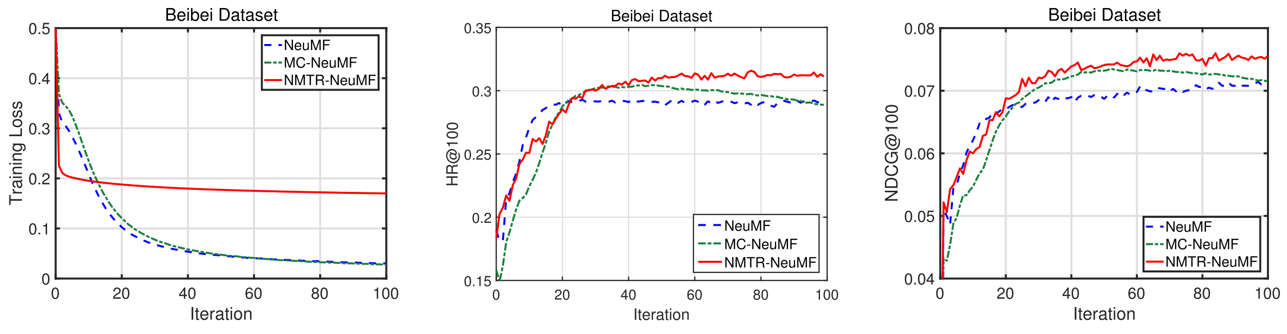
Dataset	Parameter	NMTR-GMF	NMTR-MLP	NMTR-NeuMF
Beibei	Optimzer	Adagrad	Adagrad	Adagrad
	Learning rate	0.01	0.01	0.01
	Number of layer	-	3	3
	Loss coefficient	[1/3,1/3,1/3]	[1/3,1/3,1/3]	[1/3,1/3,1/3]
	Regularization	[0,1e-5]	[0,1e-5]	[0,1e-5]
Tmall	**Optimzer**	Adagrad	Adagrad	Adagrad
	Learning rate	0.01	0.01	0.05
	Number of layer	-	3	3
	Loss coefficient	[0.4,0.6]	[0.5,0.5]	[0.4,0.6]
	Regularization term	[0,5e-5]	[0,1e-5]	[0,0]

7. There is another reason to choose a relatively larger K. In practical recommender systems, the procedure of item recommendation is typically divided into two stages [23]: candidate selection and re-ranking. Since collaborative filtering methods are typically applied in the first stage to retrieve a few hundreds of relevant items, a larger K to evaluate CF methods is more reasonable.

TABLE 3
Top-K Recommendation Performance Comparison on the Beibei and Tmall Datasets (K is Set to 50, 80, 100, 200)

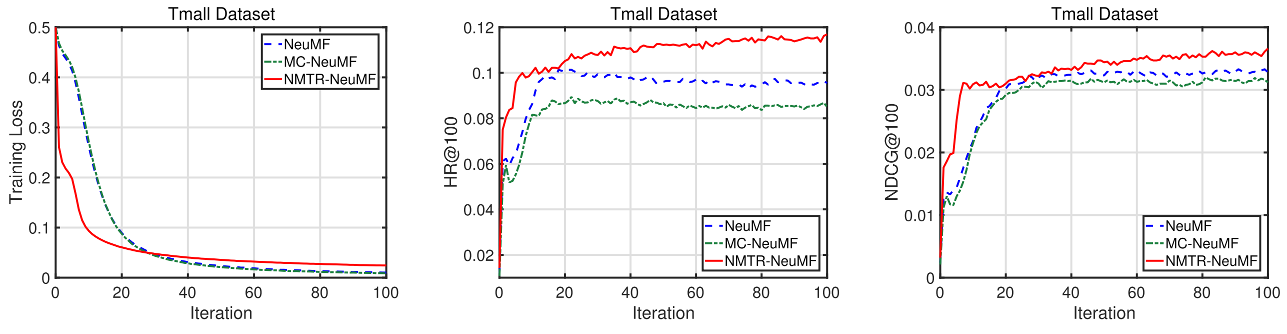
		Beibei Dataset							
Group	Method	HR@50	NDCG@50	HR@80	NDCG@80	HR@100	NDCG@100	HR@200	NDCG@200
Our NMTR Model	NMTR-GMF	0.2050	0.0590	**0.2721**	**0.0688**	0.3119	0.0741	0.4543	0.0961
	NMTR-MLP	0.1928	0.0560	0.2690	0.0676	0.3188	**0.0762**	0.4732	0.0967
	NMTR-NeuMF	**0.2079**	**0.0609**	0.2689	0.0683	**0.3193**	0.0760	**0.4766**	**0.0971**
Multi-behavior	CMF	0.1596	0.0481	0.2377	0.0606	0.2829	0.0663	0.4191	0.0850
	MC-BPR	0.1743	0.0503	0.2299	0.0604	0.2659	0.0647	0.3852	0.0786
	MC-GMF	0.1822	0.0508	0.2425	0.0611	0.2975	0.0690	0.4262	0.0891
	MC-MLP	0.1810	0.0534	0.2342	0.0598	0.2810	0.0684	0.4116	0.0834
	MC-NeuMF	0.2014	0.0577	0.2522	0.0669	0.3010	0.0719	0.4300	0.0897
Single-behavior	BPR	0.1199	0.0348	0.1686	0.0419	0.2002	0.0463	0.3039	0.0624
	GMF	0.1792	0.0475	0.2555	0.0608	0.2920	0.0665	0.4090	0.0828
	MLP	0.1711	0.0483	0.2383	0.0459	0.2679	0.0617	0.3947	0.0792
	NeuMF	0.1828	0.0573	0.2559	0.0668	0.2929	0.0714	0.4078	0.0852

		Tmall Dataset							
Group	**Method**	**HR@50**	**NDCG@50**	**HR@80**	**NDCG@80**	**HR@100**	**NDCG@100**	**HR@200**	**NDCG@200**
Our NMTR Model	NMTR-GMF	0.0778	0.0250	0.1042	0.0293	**0.1196**	0.0314	**0.1751**	0.0390
	NMTR-MLP	0.0734	0.0251	0.0884	0.0277	0.0982	0.0290	0.1672	0.0338
	NMTR-NeuMF	**0.0854**	**0.0315**	**0.1045**	**0.0347**	0.1169	**0.0366**	0.1668	**0.0428**
Multi-behavior	CMF	0.0738	0.0234	0.0940	0.0269	0.1085	0.0287	0.1565	0.0359
	MC-BPR	0.0674	0.0218	0.0928	0.0260	0.1072	0.0282	0.1597	0.0357
	MC-GMF	0.0653	0.0243	0.0778	0.0258	0.0846	0.0264	0.1084	0.0294
	MC-MLP	0.0617	0.0195	0.0784	0.0219	0.0868	0.0228	0.1122	0.0238
	MC-NeuMF	0.0711	0.0296	0.0820	0.0311	0.0894	0.0320	0.1172	0.0359
Single-behavior	BPR	0.6666	0.0200	0.0926	0.0240	0.1058	0.0263	0.1647	0.0342
	GMF	0.0742	0.0271	0.0927	0.0295	0.1027	0.0306	0.1407	0.0355
	MLP	0.0666	0.0194	0.0824	0.0220	0.0905	0.0233	0.1194	0.0273
	NeuMF	0.0760	0.0299	0.0925	0.0321	0.1013	0.0333	0.1383	0.0377



(a) Training Loss (b) HR@100 (c) NDCG@100

Fig. 2. Training loss and testing performance of NeuMF, MC-NeuMF, and NMTR-NeuMF in each iteration on Beibei.



(a) Training Loss (b) HR@100 (c) NDCG@100

Fig. 3. Training loss and testing performance of NeuMF, MC-NeuMF, and NMTR-NeuMF in each iteration on Tmall.

TABLE 4
Average Training Time of One Epoch of NeuMF, MC-NeuMF and NMTR-NeuMF on Two Datasets

Dataset	NMTR-NeuMF#	MC-NeuMF#	NeuMF#
Beibei	204.4s	472.4s	101.9s
Tmall	91.5s	73.1s	88.0s

14.37 percent in NDCG@100, which are very significant improvements. We also present the average training time per epoch of three models in Table 4 and we can find our proposed NMTR framework's efficiency is acceptable.

- *The performance on multiple behaviors are relevant to that on single behavior.* No matter which framework is chosen, NMTR or MC, we can observe that the performance of the multi-behavior setting is relevant to that of single-behavior. This is because that they use the same set of CF functions, which on the other hand implies that the performance on multi-behavior data maybe limited by the choice of the CF function. An empirical evidence is that NMTR-MLP performs the worst among the three NMTR methods, which can be caused by the poor performance of MLP in modeling CF effect (in single-behavior data). In addition, for some metrics, such as HR@50 and NDCG@50 on both dataset, and HR@80 and NDCG@80 on Tmall dataset, NMTR-MLP and NMTR-GMF are outperformed by some baseline methods such as MC-NeuMF. It can be explained that MC-NeuMF's relatively better performance is due to NeuMF's best performance compared all single-behavior methods. Therefore, our NMTR-NeuMF achieves better performance than MC-NeuMF on these metrics. Moreover, another important finding is that auxiliary behaviors could adversely degrade the performance without a proper modeling. An evidence can be found in the results of the Tmall dataset, where the methods under the MC framework fail to improve the performance in general.

To summarize, the extensive comparison on two real datasets verify that our proposed NMTR solution can effectively leverage multiple types of behaviors to improve the recommendation performance, i.e., our model outperforms the best baseline method by 6.08 and 30.76 percent on two datasets, respectively.

4.3 Impact of Auxiliary Behaviors and Parameters (RQ2)

In order to understand how auxiliary behavior data affect the recommendation performance, we choose the Beibei dataset for further investigation since it has more types of behaviors. Since the motivation of multi-behavior recommendation is to utilize interaction data of other types of behaviors to help improving recommendation quality on target behavior, we investigate how the data quality of auxiliary behaviors affects our NMTR model's performance. An intuitive experimental setting is that to random sample auxiliary behaviors for our utilized two datasets while keeping target behaivor (i.e., purchase) intact. Table 5 shows the performance of different combinations of behavioral data. There are four sampling rules for obtaining a subset. For example, (Purchase, 50 percent view) means that intact purchase records are kept and half records of view behavior are randomly selected to be kept for each user. As mentioned above, when investigating top-K performance, K=100 is a reasonable setting. Thus, here we evaluated the performance via two metrics: HR@100 and NDCG@100. We tuned hyper-parameters, with a similar way as Section 4.1, to report the best performance for various subsets of interaction data. From the results, we have the following two observations.

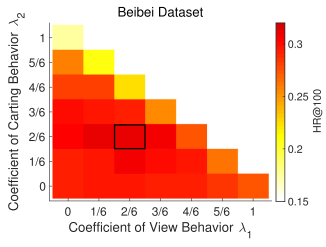
First, adding views data leads to better performance than adding carts data. The main reason is probably that the cart data contains too similar signal with the purchase data and provides fewer new signal on user preference. Specifically, a purchase record is often accompanied by a carting record. On the contrary, the view behaviors provide some useful intermediate feedback such as, viewed and not bought, which can effectively improve the learning on binary implicit feedback.

Second, by using only 50 percent of the cart and view interactions, we find that the performance is worse than the previous two experiments. Specifically, the performance of (Purchase, 50 percent Carting) is worse than only using purchase, while (Purchase, 50 percent View) is better than only using purchase. There are two major reasons. On one hand, view is the weakest signal to reflect user preference and the total number of views is very large, making the missing of part of view data is acceptable. Therefore, missing of some view records shall not affect the result too much. On the other hand, random missing of carts records can bring some noises, as cart behavior is very similar with the purchase behavior, and this validates the hypothesis in [24]: those missing records of some behaviors are more likely taken as negative value rather than missing value by model.

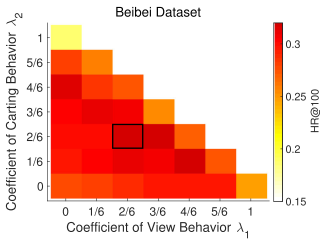
In order to understand how hyper-parameters impact the performance, we focus on the coefficient in the joint loss function of MTL, λ_r, since it controls the weight of each type of behavior and is a key parameter of our method. There are three and two behavior types for Beibei and Tmall, respectively. For the Beibei dataset, there are three types of behaviors (view, cart and purchase), which means there are three

TABLE 5
Performance of NMTR Model with Different Combination of Interaction Data on the Beibei Dataset

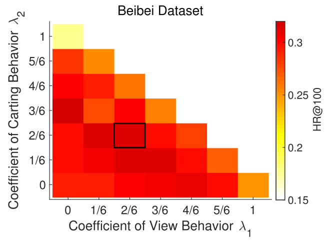
	Beibei Dataset							
Interaction Subset	(Purchase, Carting)		(Purchase, View)		(Purchase, 50% Carting)		(Purchase, 50% View)	
Performance	HR@100	NDCG@100	HR@100	NDCG@100	HR@100	NDCG@100	HR@100	NDCG@100
NMTR-GMF	0.2979	0.0705	0.3029	0.0726	0.2947	0.0701	0.2953	0.0698
NMTR-MLP	0.2770	0.0670	0.3140	0.0741	0.2726	0.0654	0.3058	0.0725
NMTR-NeuMF	0.2882	0.0691	0.3147	0.0743	0.2778	0.0676	0.3107	0.0737



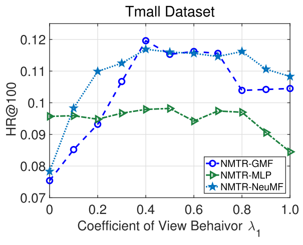
(a) NMTR-GMF on Beibei Dataset



(b) NMTR-MLP on Beibei Dataset



(c) NMTR-NeuMF on Beibei Dataset



(d) NMTR on Beibei Dataset

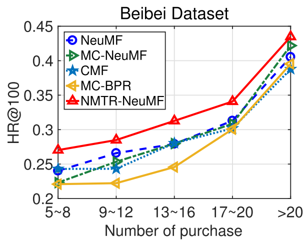
Fig. 4. HR@100 Performance of NMTR with different loss coefficient on the Beibei and Tmall datasets.

loss coefficients λ_1, λ_2 and λ_3, respectively. Note that $\lambda_1 + \lambda_2 + \lambda_3 = 1$, we tune the three coefficients in $[0, \frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}, 1]$ and plot the performance of HR@100 in Figs. 4a, 4b and 4c. When λ_1 and λ_2 are given, then value of λ_3 is determined. Therefore each block represents a setting of λ_r. And in these three figures, darker blocks means better performance. Similarly, for the Tmall dataset, there are only two types of behaviors (view and purchase), so there are two coefficients: $\lambda_1 + \lambda_2 = 1$. We tune λ_1 from 0 to 1 with step size 0.1 and plot the HR@100 performance in Fig. 4d. For both datasets, the best performance of the NMTR methods are achieved at almost the same setting, (2/6,2/6,2/6) for the Beibei dataset and about(0.4, 0.6) for the Tmall dataset, which verifies that it is not so independent on the utilized CF unit. For Beibei dataset, in Figs. 4a, 4b and 4c, upper-right blocks are rather shallow since they represent a relatively small λ_3 which is the coefficient of purchase behavior. However, for Tmall dataset, in Fig. 4d, a relatively low coefficient of purchase behavior outperforms that of view behavior. We argue that it is due to size difference of auxiliary behavioral data in two datasets.

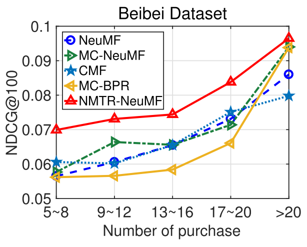
Furthermore, as mentioned in Section 3.4, we utilize multi-task learning rather than sequential learning to optimize our proposed model. Then to study how multi-task learning outperforms the intuitive sequential learning in optimizing the cascaded prediction models, we compare the performance of the two training methods in Table 6. Here we still adopt

TABLE 6
Performance Comparison of Sequential Training and Multi-Task Learning on the Beibei and Tmall Datasets

Dataset	Beibei		Tmall	
Performance	HR@100	NDCG@100	HR@100	NDCG@100
NMTR-GMF	0.3119	0.0741	0.1196	0.0314
Sequential-GMF	0.2730	0.0672	0.0913	0.0290
NMTR-MLP	0.3188	0.0762	0.0982	0.0290
Sequential-MLP	0.2663	0.0692	0.0856	0.0226
NMTR-NeuMF	0.3193	0.0760	0.1169	0.0366
Sequential-NeuMF	0.2704	0.0658	0.0946	0.0304



(a) HR@100



(b) NDCG@100

Fig. 5. Performance of NeuMF, MC-NeuMF, CMF, MC-BPR and NMTR-NeuMF on users with different number of purchase records.

HR@100 and NDCG@100 as evaluation metrics. For the sequential learning, we feed the cascaded CF units for each behavior type with separated samples following the order of behaviors. We can find that training in the sequential manner achieved worse performance, which verified the effectiveness of our proposed multi-task training component.

In summary, our NMTR incorporates the semantics of different behavior interactions and capture the ordinal relations among them. In addition, coefficient λ_r, as a significant hyper-parameter in our NMTR model, is independent with CF unit. Furthermore, multi-task training is demonstrated to far better than sequential training.

4.4 Impact of Data Sparsity (RQ3)

Data sparsity is a big challenge for recommender systems based on implicit feedbacks [11], [25], and multi-behavior recommendation is a typical solution of it. Thus, we study how our proposed NMTR model improves the recommendation for those users having few records of target behavior. Specifically, we divided all users of the Beibei dataset into several groups according to the number of purchase records: [5-8, 9-12, 13-16, 17-20, $>$ 20]. In each group, the number of users are in the range of 4,000 to 5,000, which eliminates the randomness of experimental results. For each group, we compare the performance of our methods with baseline methods. For NMTR and MC models, we only plot the most competitive ones, NMTR-NeuMF and MC-NeuMF, for clarity; for baselines for single-behavior data, we also only plot the best one, NeuMF.

The results are shown in Fig. 5. From the results, we can observe that when the user purchase data becomes sparser, the recommendation performance of NMTR-NeuMF decreases slower than other methods. Especially for NDCG, from fifth to first user group, NMTR-NeuMF is decreased by 27.56 percent while MC-BPR and MC-NeuMF is decreased by 40.09 and 38.62 percent. Furthermore, even in the first user group with only 5-8 purchase records, our NMTR still keeps a good recommendation performance of 0.027 for HR@100 and 0.07 for NDCG@100, which outperforms the best baseline by 11.23 and 15.35 percent, respectively. As a result, the performance gap between NMTR and other methods becomes larger when data become sparser. Since NMTR model learns the other type of behaviors in a reasonable way, it can achieve a good performance for users with sparse interactions. As a summary, we conclude that our proposed NMTR model solves data sparsity problem efficiently to some extent.

In conclusion, we conduct extensive experiments on two real-word datasets, which verifies that our proposed NMTR model outperform existing recommendation methods. Further studies demonstrate our model can alleviate data sparsity problem efficiently.

5 Related Work

5.1 Multi-Behavior Recommendation

In today's online information systems, user can interact with a system in multiple forms. There are many works [26], [27] analyzing and modeling such multiple types of behaviors. Lo *et al.* [26] studied the influence of click and save behavior on the final purchase decision via case study. Moe *et al.* [28], Dong *et al.* [27] and Lee *et al.* [29] utilized various time-evolving behavioral features to predict purchase behaviors. Olbrich *et al.* [30] and Yehezki *et al.* [31] proposed to extract features from user clickstreams to help predict purchase. These works verify the effectiveness of other types of behaviors to help model the target behavior. Multi-behavior based recommendation aims to leverage the behavior data of other types to improve the recommendation performance on the target behavior. Matrix factorization, a prevalent method for single-behavior based recommendation [1], [11], has been adapted to the multi-behavior scenario. Ajit *et al.* [3] first proposed a collective matrix factorization model (CMF) to simultaneously factorize multiple user-item interaction matrices with sharing item-side embeddings across matrices. Some other works extended the CMF to handle datasets of multiple user behaviors [5], [6]. Zhe *et al.* [6] considered different behaviors in online social network (comment, re-share, and create-post), while Artus *et al.* [5] extended CMF with sharing user-side embeddings in recommendation based social network data. On the other hand, some works approach multi-behavior recommendation from the perspective of learning [7], [8], [32], [33]. Babak *et al.* [7] proposed an extension of Bayesian Personalized Ranking [1], as Multi-channel BPR, to adapt the sampling rule from different types of behavior in training of standard BPR. Qiu *et al.* [8] proposed an adaptive sampler for BPR considering co-occurrence of multiple feedbacks while Guo *et al.* [32] proposed to sample unobserved items as positive items based on item-item similarity, which is calculated by multiple types of feedbacks. Ding *et al.* [33] developed a margin-based pairwise learning framework when view-data is available. As discussed in the introduction, these existing models suffer from several limitations, which are addressed by our neural network-based solution NMTR.

5.2 Neural Network Based Recommendation

Salakhutdinov *et al.* [34] proposed a Restricted Boltzmann Machines to predict explicit ratings, which was the first to apply neural network to recommender system. Recently, lots of works utilize neural network to extract the auxiliary information and features in recommender system, such as textual [35], [36], [37], [38], [39], visual [40], [41], audio [42], [42] and video [43]. Rather than these other side features, some other works make use of recurrent neural network to model temporal features in recommender system [44], [45], [46], [47].

More recently, He *et al.* [2] proposed a neural network architecture for collaborative filtering, named Neural Collaborative Filtering, which learns the user-item interaction function using neural networks. It has been extended to adapt to different recommendation scenarios [12], [48]. For example, Wang *et al.* [12] applied NCF to model user-item interaction in both information domain and social domain, and Chen *et al.* [48] combined NCF with attention mechanism to recommend videos and images. Recently, inspire the advances in graph representation learning, some works [49], [50], [51] utilize graph neural network [52] for recommendation tasks. Our work extends the architecture of NCF to a multi-task learning framework, which aims to solve the problem of learning recommender systems from multi-behavior data.

5.3 Multi-Task Learning for Recommendation

In multi-task learning (MTL) framework, various related tasks can share common representations, while training in parallel. Traditional multi-task learning works are mainly based on matrix regularization [18], [53] and neural-based approach [54], [55]. To the best of our knowledge, [56] is the first work to apply multi-task learning to recommender system, which built a MTL framework to limit the similarity between users and similarity between items. Bansal *et al.* [57] proposed a gated-recurrent-units based MTL network which share the embedded representation of texts and output personalized text for different users. In contrast, our work adapts MTL our task to effectively learn from multiple user behaviors.

6 Conclusion and Future Work

In this work, we designed a recommendation system to exploit multiple types of user behaviors. We proposed a neural network method named NMTR, which combines the recent advances of NCF modeling and the efficacy of multi-task learning. We conducted extensive experiments on two real-world datasets and demonstrated the effectiveness of our NMTR method on multiple recommender models. This work makes the first step towards understanding how to integrate the rich semantics of users' multiple behaviors into recommender systems. With increasing kinds of user behaviors on the Web, we believe multi-behavior recommendation is an important topic and will attract more attention in the future.

As for future work, we will perform online evaluation of our NMTR method through A/B tests, and focus more on the practical issues of online learning and incremental learning. On the other hand, we will study multi-behavior recommendation in the scenarios that user behaviors cannot form a full-order cascading relation. These behaviors not only contain the normal interactions between users and items, but may also include social interactions among users, such as sharing, following, etc. It is interesting to investigate how to integrate these heterogeneous kinds of user behaviors into a unified recommendation framework. Lastly, we will study time-aware models to capture the evolution of user preference in multi-behavior recommendation, especially for capturing dynamic user interests with RNN-based or other models for better recommendation.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1800804, the National Nature Science Foundation of

China under Grants U1936217, 61971267, 61972223, 61941117, 61861136003, Beijing Natural Science Foundation under Grant L182038, Beijing National Research Center for Information Science and Technology under Grant 20031887521, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology. An early conference version of this work was published in IEEE ICDE 2019 [58].

References

[1] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[3] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 650–658.

[4] C. Park, D. Kim, J. Oh, and H. Yu, "Do also-viewed products help user rating prediction?" in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 1113–1122.

[5] A. Krohn-Grimberghe, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme, "Multi-relational matrix factorization using Bayesian personalized ranking for social network data," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, 2012, pp. 173–182.

[6] Z. Zhao, Z. Cheng, L. Hong, and E. H. Chi, "Improving user topic interest profiles by behavior factorization," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1406–1416.

[7] B. Loni, R. Pagano, M. Larson, and A. Hanjalic, "Bayesian personalized ranking with multi-channel user feedback," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 361–364.

[8] H. Qiu, Y. Liu, G. Guo, Z. Sun, J. Zhang, and H. T. Nguyen, "BPRH: Bayesian personalized ranking for heterogeneous implicit feedback," *Inf. Sci.*, vol. 453, pp. 80–98, 2018.

[9] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 83–92.

[10] R. Heckel, M. Vlachos, T. Parnell, and C. Dünner, "Scalable and interpretable product recommendations via overlapping co-clustering," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 1033–1044.

[11] S. Kabbur, X. Ning, and G. Karypis, "FISM: Factored item similarity models for top-N recommender systems," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 659–667.

[12] X. Wang, X. He, L. Nie, and T.-S. Chua, "Item silk road: Recommending items from information domains to social users," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 185–194.

[13] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[14] T. Bai, J. Wen, J. Zhang, and W. X. Zhao, "A neural collaborative filtering model with interaction-based neighborhood," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1979–1982.

[15] L. Xiang *et al.*, "Temporal recommendation on graphs via long- and short-term preference fusion," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 723–732.

[16] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov Chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 811–820.

[17] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–10.

[18] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 109–117.

[19] X. Chen, Y. Zhang, Q. Ai, H. Xu, J. Yan, and Z. Qin, "Personalized key frame recommendation," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 315–324.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[21] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[22] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[23] W. Wang, H. Yin, S. Sadiq, L. Chen, M. Xie, and X. Zhou, "SPORE: A sequential personalized spatial item recommender system," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 954–965.

[24] H. Steck, "Training and testing of recommender systems on data missing not at random," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 713–722.

[25] H. Yin, L. Chen, W. Wang, X. Du, Q. V. H. Nguyen, and X. Zhou, "Mobi-SAGE: A sparse additive generative model for mobile app recommendation," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 75–78.

[26] C. Lo, D. Frankowski, and J. Leskovec, "Understanding behaviors that lead to purchasing: A case study of pinterest," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 531–540.

[27] Y. Dong and W. Jiang, "Brand purchase prediction based on time-evolving user behaviors in e-commerce," *Concurrency Computation: Practice Experience*, vol. 31, no. 1, 2019, Art. no. e4882.

[28] W. W. Moe and P. S. Fader, "Dynamic conversion behavior at e-commerce sites," *Manage. Sci.*, vol. 50, no. 3, pp. 326–335, 2004.

[29] M. Lee, T. Ha, J. Han, J.-Y. Rha, and T. T. Kwon, "Online footsteps to purchase: Exploring consumer behaviors on online shopping sites," in *Proc. ACM Web Sci. Conf.*, 2015, Art. no. 15.

[30] R. Olbrich and C. Holsing, "Modeling consumer purchasing behavior in social shopping communities with clickstream data," *Int. J. Electron. Commerce*, vol. 16, no. 2, pp. 15–40, 2011.

[31] S. Yehezki and A. Dhini, "Classifying purchase decision based on user clickstream in e-commerce using web usage mining," in *Proc. Int. Conf. Bus. Inf. Manage.*, 2017, pp. 57–61.

[32] G. Guo, H. Qiu, Z. Tan, Y. Liu, J. Ma, and X. Wang, "Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems," *Knowl.-Based Syst.*, vol. 138, pp. 202–207, 2017.

[33] J. Ding *et al.*, "Improving implicit recommender systems with view data," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3343–3349.

[34] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 791–798.

[35] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 425–434.

[36] L. Tang and E. Y. Liu, "Joint user-entity representation learning for event recommendation in social network," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 271–280.

[37] Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli, "Aspect-aware latent factor model: Rating prediction with ratings and reviews," in *Proc. World Wide Web Conf.*, 2018, pp. 639–648.

[38] Z. Cheng, X. Chang, L. Zhu, R. C. Kanjirathinkal, and M. Kankanhalli, "MMALFM: Explainable recommendation by leveraging reviews and images," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, 2019, Art. no. 16.

[39] Z. Cheng, Y. Ding, X. He, L. Zhu, X. Song, and M. S. Kankanhalli, "A^3 NCF: An adaptive aspect attention model for rating prediction," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3748–3754.

[40] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 43–52.

[41] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What your images reveal: Exploiting visual contents for point-of-interest recommendation," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 391–400.

[42] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2643–2651.

[43] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 191–198.

[44] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in NetEase," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 1218–1229.

[45] S. Okura, Y. Tagami, S. Ono, and A. Tajima, "Embedding-based news recommendation for millions of users," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1933–1942.

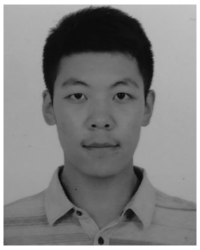
[46] Y. Song, A. M. Elkahky, and X. He, "Multi-rate deep learning for temporal recommendation," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 909–912.

[47] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 197–206.
[48] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 335–344.
[49] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 974–983.
[50] W. Fan *et al.*, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 417–426.
[51] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 165–174.
[52] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.
[53] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 41–48.
[54] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang, "Exploiting feature and class relationships in video categorization with regularized deep neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 352–364, Feb. 2018.
[55] Y. Yang and T. Hospedales, "Deep multi-task representation learning: A tensor factorisation approach," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–12.
[56] X. Ning and G. Karypis, "Multi-task learning for recommender system," in *Proc. 12th ACM Conf. Recommender Syst.*, 2010, pp. 269–284.
[57] T. Bansal, D. Belanger, and A. McCallum, "Ask the GRU: Multi-task learning for deep text recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 107–114.
[58] C. Gao *et al.*, "Neural multi-task recommendation from multi-behavior data," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 1554–1557.

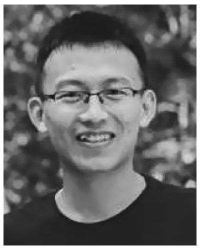
Chen Gao received the BS degree in electronic engineering from Tsinghua University, Beijing, China, in 2016. He is currently working toward the PhD degree in the Department of Electronic Engineering, Tsinghua University, Beijing, China. His reserach interests include recommender systems, data mining, and data privacy. He has publications appeared in several top conferences such as WWW, ICDE, KDD, and SIGIR.

Xiangnan He received the PhD degree in computer science from National University of Singapore. He is currently a research fellow with the School of Computing, National University of Singapore (NUS). His research interests span recommender system, information retrieval, natural language processing and multi-media. His work on recommender system has received the Best Paper Award Honorable Mention in WWW 2018 and SIGIR 2016. Moreover, he has served as the PC member for top-tier conferences including SIGIR, WWW, MM, KDD, WSDM, CIKM, AAAI, and ACL, and the invited reviewer for prestigious journals including the *IEEE Transactions on Knowledge and Data Engineering*, the *ACM Transactions on Information Systems*, the *ACM Transactions on Knowledge Discovery from Data*, the *IEEE Transactions on Multimedia*, and the *World Wide Web Journal*. He is a member of the IEEE.

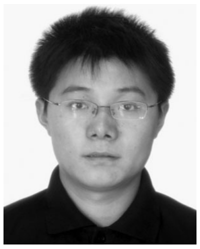
Dahua Gan received the BS degree in electrical engineering from Tsinghua University, Beijing, China, in 2018. He is currently working toward the MS degree in the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania. His research focus on data mining.



Xiangning Chen is currently working toward the undergraduate degree in Electronic Engineering Department, Tsinghua University, Beijing, China. His work mainly focuses on recommender systems.



Fuli Feng received the BE degree from the School of Computer Science and Engineering, Baihang University, Beijing, in 2015. He is a research fellow with the School of Computing, National University of Singapore. His research interests include information retrieval, data mining, and multi-media processing. He has more than 10 publications appeared in several top conferences such as SIGIR, WWW, and MM. His work on Bayesian Personalized Ranking has received the Best Poster Award of WWW 2018. Moreover, he has been served as the PC member and external reviewer for several top conferences including SIGIR, ACL, KDD, IJCAI, AAAI, WSDM etc.



Yong Li (M'2009-SM'2016) received the BS degree in electronics and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2007 and the PhD degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. He is currently a faculty member of the Department of Electronic Engineering, Tsinghua University. He has served as general chair, TPC chair, TPC member for several international workshops and conferences, and he serves in the editorial board of two IEEE journals. His papers have total citations more than 5200. Among them, ten are ESI highly cited papers in computer science, and four receive conference best paper (run-up) awards. He received IEEE 2016 ComSoc Asia-Pacific Outstanding Young Researchers and Young Talent Program of China Association for Science and Technology. He is a senior member of the IEEE.

Tat-Seng Chua received the PhD degree from the University of Leeds, United Kingdom. He is the KITHCT chair professor with the School of Computing, National University of Singapore. He was the acting and founding dean of the School from 1998 – 2000. His main research interests include multimedia information retrieval and social media analytics. In particular, his research focuses on the extraction, retrieval and question-answering (QA) of text and rich media arising from the web and multiple social networks. He is the co-director of NExT, a joint Center between NUS and Tsinghua University to develop technologies for live social media search. He is the 2015 winner of the prestigious ACM SIGMM Award for Outstanding Technical Contributions to Multimedia Computing, Communications and Applications. He is the chair of steering committee of ACM International Conference on Multimedia Retrieval (ICMR) and Multimedia Modeling (MMM) conference series. He is also the general co-chair of ACM Multimedia 2005, ACM CIVR (now ACM ICMR) 2005, ACM SIGIR 2008, and ACMWeb Science 2015. He serves in the editorial boards of four international journals. He is the co-founder of two technology startup companies in Singapore.

Lina Yao is currently a senior lecturer with the School of Computer Science and Engineering, the University of New South Wales (UNSW), Australia. Her research interests include data mining and machine learning applications with the focuses on Internet of Things analytics, recommender systems, human activity recognition, and brain computer interface.

Yang Song received the BEng degree in computer engineering from Nanyang Technological University, Singapore, and received the PhD degree in computer science from the University of Sydney, in 2013. She is a lecturer with the School of Computer Science and Engineering, the University of New South Wales (UNSW), Australia. She received the highly competitive Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA), in 2015, and was an ARC DECRA fellow with the University of Sydney before joining UNSW, in 2018. She also received the Engineering Deans Research Award, the University of Sydney, in 2017.

Depeng Jin (M'2009) received the BS and PhD degrees from Tsinghua University, Beijing, China, in 1995 and 1999, respectively both in electronics engineering. Now he is an associate professor at Tsinghua University and vice chair of Department of Electronic Engineering. He was awarded National Scientific and Technological Innovation Prize (Second Class), in 2002. His research interests include telecommunications, high-speed networks, ASIC design, and future Internet architecture. He is a member of the IEEE.