Multi-Task Recommendations with Reinforcement Learning

Ziru Liu
City University of Hong Kong
ziruliu2-c@my.cityu.edu.hk

Jiejie Tian
City University of Hong Kong
jiejitian2-c@my.cityu.edu.hk

Qingpeng Cai*
Kuaishou Technology
cqpcurry@gmail.com

Xiangyu Zhao*
City University of Hong Kong
xianzhao@cityu.edu.hk

Jingtong Gao
City University of Hong Kong
jt.g@my.cityu.edu.hk

Shuchang Liu
Kuaishou Technology
liushuchang@kuaishou.com

Dayou Chen
City University of Hong Kong
dayouchen2-c@my.cityu.edu.hk

Tonghao He
City University of Hong Kong
tonghaohe2-c@my.cityu.edu.hk

Dong Zheng
Kuaishou Technology
zhengd07@qq.com

Peng Jiang
Kuaishou Technology
jp2006@139.com

Kun Gai
Unaffiliated
gai.kun@qq.com

ABSTRACT

In recent years, Multi-task Learning (MTL) has yielded immense success in Recommender System (RS) applications [40]. However, current MTL-based recommendation models tend to disregard the session-wise patterns of user-item interactions because they are predominantly constructed based on item-wise datasets. Moreover, balancing multiple objectives has always been a challenge in this field, which is typically avoided via linear estimations in existing works. To address these issues, in this paper, we propose a Reinforcement Learning (RL) enhanced MTL framework, namely RMTL, to combine the losses of different recommendation tasks using dynamic weights. To be specific, the RMTL structure can address the two aforementioned issues by (*i*) constructing an MTL environment from session-wise interactions and (*ii*) training multi-task actor-critic network structure, which is compatible with most existing MTL-based recommendation models, and (*iii*) optimizing and fine-tuning the MTL loss function using the weights generated by critic networks. Experiments on two real-world public datasets demonstrate the effectiveness of RMTL with a higher AUC against state-of-the-art MTL-based recommendation models. Additionally, we evaluate and validate RMTL's compatibility and transferability across various MTL models.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Multi-task Learning; Recommendation; Reinforcement Learning

ACM Reference Format:
Ziru Liu, Jiejie Tian, Qingpeng Cai*, Xiangyu Zhao*, Jingtong Gao, Shuchang Liu, Dayou Chen, Tonghao He, Dong Zheng, Peng Jiang, and Kun Gai. 2023. Multi-Task Recommendations with Reinforcement Learning. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583467

* Corresponding authors.

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583467

1 INTRODUCTION

The evolution of the Internet industry has led to a tremendous increase in the information volume of online services [1], such as social media and online shopping. In this scenario, the Recommender System (RS), which distributes various types of items to match users' interests, has made significant contributions to the enhancement of user online experiences in a variety of application fields, such as products recommendation in e-commerce platforms, short video recommendation in social media applications [30, 45]. In recent years, researchers have proposed numerous techniques for recommendations, including collaborative filtering [34], matrix factorization based approaches [22], deep learning powered recommendations [9, 50], etc. The primary objective of RS is to optimize a specific recommendation object, such as click-through rate and user conversion rate. However, users usually have varying interaction behaviors on a single item. In short-video recommendation services, for instance, users exhibit a wide range of behavior indicators, such as clicks, thumbs, and continuous dwelling time [17]; while in e-commerce platforms, the developers not only focus on the users' clicks but also on the final purchases to guarantee profits. All these potential issues prompted the development of Multi-Task Learning (MTL) techniques for recommender systems in research and industry communities [29, 45, 46].

MTL-based recommendation models learn multiple recommendation tasks simultaneously by training in a shared representation

and transferring information among tasks [5], which has been developed for a wide range of machine learning applications, including computer vision [38], natural language processing [14], click-through rate (CTR) and click-through&conversion rate (CTCVR) prediction [29]. The objective functions for most existing MTL works are typically linear scalarizations of the multiple-task loss functions [29, 30, 45], which fix the weight with a constant. This item-wise multi-objective loss function is incapable of ensuring the convergence of the global optimum and typically yields limited prediction performance. On the other hand, at the representation level, the input of most existing MTL models is assumed to be the feature embeddings and user-item interaction (called item-wise), despite the fact that sequentially organized data (i.e., session-wise inputs) are relatively more prevalent in real-world RS applications. For example, the click and conversion behaviors of short video users typically occur during a specific session, so their inputs are also timing-related. However, this will downgrade the MTL model performance, while some tasks may have conflicts between session-wise and item-wise labels [5]. Exiting MTL models concentrate on the design of network structures to improve the generalization ability of the model, while the study of proposing a new method that enhances the multi-task prediction weights considering the session-wise patterns has not received sufficient attention.

To address the two above-mentioned problems, we propose an RL-enhanced multi-task recommendation framework, RMTL, which is capable of incorporating the sequential property of user-item interactions into MTL recommendations and automatically updating the task-wise weights in the overall loss function. Reinforcement Learning (RL) algorithms have recently been applied in the RS research, which models the sequential user behaviors as Markov Decision Process (MDP) and utilizes RL to generate recommendations at each decision step [31, 57]. The RL-based recommender system is capable of handling the sequential user-item interaction and optimizing long-term user engagement [2]. Therefore, our RL-enhanced framework RMTL can convert the session-wise RS data into MDP manner, and train an actor-critic framework to generate dynamic weights for optimizing the MTL loss function. To achieve multi-task output, we employ a two-tower MTL backbone model as the actor network, which is optimized by two distinct critic networks for each task. In contrast to existing MTL models with item-wise input and constant loss function weight design, our RMTL model extracts sequential patterns from session-wise MDP input and updates the loss function weights automatically for each batch of data instances. In this paper, we focus on the CTR/CTCVR prediction, which is a crucial metric in e-commerce and short video platform [25]. Experiments against state-of-the-art MTL-based recommendation models on two real-world datasets demonstrate the effectiveness of the proposed model.

We summarize the contributions of our work as follows: (i) The multi-task recommendation problem is converted into an actor-critic reinforcement learning scheme, which is capable of achieving session-wise multi-task prediction; (ii) We propose an RL-enhanced Multi-task learning framework RMTL, which can generate adaptively adjusted weights for loss function design. RMTL is compatible with most existing MTL-based recommendation models; (iii) Extensive experiments on two real-world datasets demonstrate the superior performance of RMTL than SOTA MTL models, we also verify RMTL's transferability across various MTL models.

2 THE PROPOSED FRAMEWORK

This section will give a detailed description of our method, the RMTL framework, which effectively addresses the bottlenecks of exiting works by realizing session-wise multi-task prediction with dynamically adjusted loss function weights.

2.1 Preliminary and Notations

Session-wise Multi-task Recommendations. We note that the choice of loss function as an item-wise multi-objective combination may lack the ability to extract sequential patterns from data. In our work, we propose the session-wise multi-objective loss, which optimizes the objective by minimizing weighted cumulative loss at each session. Given a K-task[1] prediction dataset with $D := \left\{user_n, item_n, (y_{n,1}, ..., y_{n,K})\right\}_{n=1}^{N}$, which consist of N user-item interaction records, where $user_n$ and $item_n$ means the n-th user-item id. Each data record has K binary 0-1 labels $y_1, \cdots, y_K$ for the corresponding task. We define the session as $\{\boldsymbol{\tau}_m\}_{m=1}^{M}$, each session $\boldsymbol{\tau}_m$ contains T_m different discrete timestamps: $\tau_m = \left\{\tau_{m,1}, \cdots, \tau_{m,t}, \cdots, \tau_{m,T_m}\right\}$, where $\tau_{m,t}$ stands for the timestamp t in session m. The corresponding label is denoted by $\left\{(y^s_{t,1}, ..., y^s_{t,K})\right\}$. The session-wise loss function for all sessions parameterized by $\theta^s_1, ..., \theta^s_K$ is defined as:

$$\begin{aligned} \underset{\{\theta^s_1,...,\theta^s_K\}}{\arg\min}\ \mathcal{L}^s(\theta^s_1, \cdots, \theta^s_K) &= \sum_{k=1}^{K}\Big(\sum_{m=1}^{M}\sum_{t=1}^{T_m}\omega^s_{k,\tau_{m,t}} L^s_{k,\tau_{m,t}}(\theta^s_k)\Big) \\ s.t.\quad L^s_{k,\tau_{m,t}}(\theta^s_k) &= -[y^s_{k,\tau_{m,t}} log(z^s_{k,\tau_{m,t}}(\theta^s_k)) + \\ &\quad (1 - y^s_{k,\tau_{m,t}}) log(1 - z^s_{k,\tau_{m,t}}(\theta^s_k))] \end{aligned} \tag{1}$$

where $\mathcal{L}^s(\theta^s_1, \cdots, \theta^s_K)$ is the session-wise total loss function, and ω^s_{k,τ_m} is the corresponding weight for the BCE loss, which adaptively adjusted by the system. $z^s_{n,k}(\theta^s_k)$ is the session-wise prediction value at n-th data point for task k parameterized by θ^s_k. In order to obtain such weight, we take advantage of the reinforcement learning framework to estimate the weight using several dynamic critic networks that could tune their outputs at different solving steps of the objective function.

2.2 Framework Overview

We present the framework of RMTL in Figure 1 with a state representation network, an actor network, and critic networks. The learning process of RMTL can be summarized as two steps:

- **The Forward Step.** Given the user-item combined features, the state representation network generates the state s_t based on the input feature at timestamp t. Then, we get the action $(a_{1,t}, a_{2,t})$ from the actor network extracting state information. The action value $(a_{1,t}, a_{2,t})$ and user-item combined features are further processed by the MLP layer and embedding layer as the input of the critic network for calculating Q-value from critic network $Q(s_t, a_{k,t}; \phi_k)$ for each task k. Finally, the overall loss function for

[1]Note that for multi-task recommendations in CTR/CTCVR prediction setting, we have $K = 2$.

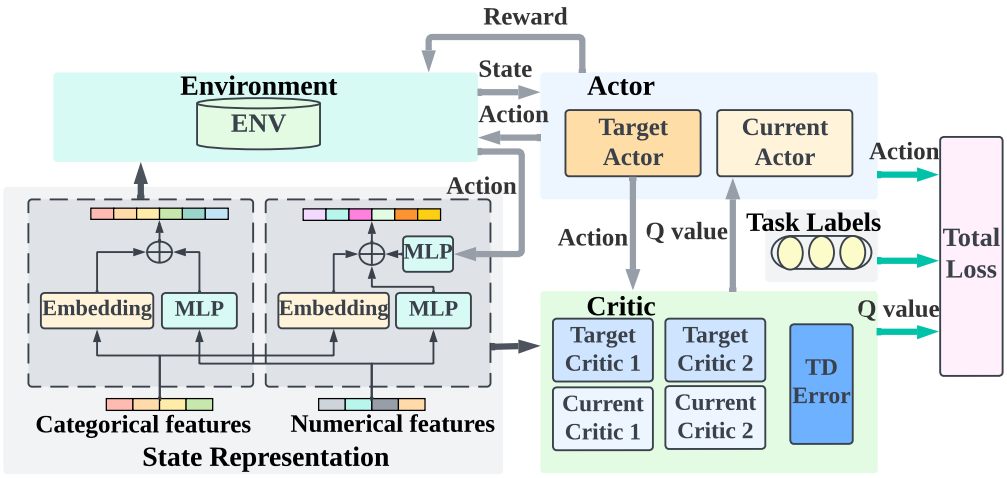

Figure 1: Overview of the RMTL framework.

multi-task $\mathcal{L}(\theta_1, \theta_2)$ can be estimated according to BCE loss for each task and adapted weight $\omega_{k,t}$ controlled by $Q(s_t, a_{k,t}; \phi_k)$.

- **The Backward Step.** We first update the critic network parameters ϕ_k based on TD error σ and gradients of Q-value. Then, we optimize the parameters of the actor network θ_k with respect to the overall loss function.

2.3 Markov Decision Process (MDP) Design

We translate the multi-task prediction data into an MDP format $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ to analyze in the reinforcement learning framework.

- **State space** $\mathcal{S}$ is the set of state $s_t \in \mathrm{R}^d$ containing the user-item combined features, where d is the combined feature dimension;
- **Action space** In our CTR/CTCVR prediction setting, $\mathcal{A}$ is the set of continuous action pairs $(a_{1,t}, a_{2,t}) \in [0, 1]^2$ containing the actions for two specific prediction tasks, where each element in $\mathcal{A}$ represents prediction value for CTR and CTCVR;
- **Reward** $r_{k,t} = r(a_{k,t}, y_{k,t}) \in \mathrm{R}, k = 1, 2$, is the feedback from the environment related to the current actions and the ground-truth click/pay label pair $(y_{1,t}, y_{2,t})$ of the current item. In order to be consistent with the definition of BCE loss, we define the reward function for each step using the negative BCE value, i.e.,

$$r_{k,t} = y_{k,t} log(a_{k,t}) + (1 - y_{k,t}) log(1 - a_{k,t}), \quad k = 1, 2 \tag{2}$$

- **Transition** $\mathcal{P}_{sas'} = \chi(s' = s_{t+1})$ is the transition probability from state s to state s' based on the user interaction sequence, and χ is the indicator function that set the probability as 1 when the next state (item) corresponds to the sequence;
- **Discount** γ the discount rate is set as 0.95 for our case.

The RMTL model learns a optimal policy $\pi(s_t; \theta^*)$ which converts the item-user feature $s_t \in \mathcal{S}$ into the continuous action value $(a_{1,t}, a_{2,t}) \in \mathcal{A}$ to maximize the weighted total rewards:

$$\max_{\{\theta_1, \theta_2\}} \sum_{k=1}^{2} \Big(\sum_{m=1}^{M} \sum_{t=1}^{T_m} \omega^r_{k,\tau_{m,t}} R_{k,\tau_{m,t}}(\theta_k) \Big)$$
$$s.t. \quad R_{k,\tau_{m,t}}(\theta_k) = r(a_{k,\tau_{m,t}}, y_{k,\tau_{m,t}}) \quad k = 1, 2 \tag{3}$$

where $\omega^r_{k,\tau_{m,t}}$ is the weight for reward at session τ_m in timestamp t for task k. Since the reward function defined by Equation (2) is the negative BCE, and action value is generated from policy parameterized by θ_k. The optimization problem above is equivalent to minimizing the session-wise loss function in Equation (1).

2.4 Session MDP Construction

Based on the MDP design in Section 2.3, we construct session MDP for the RL training, which can improve the performance of the MTL model. Classic MTL methods usually face challenges in introducing sequential user behaviors into the modeling, where the timestamps of user behaviors are highly correlated. Reinforcement learning built upon the MDP sequences can be a solution to address this problem. However, the alignment order of MDP may have great influences on the performance of RL, since the decision-making of the next action depends on the temporal difference (TD). In this paper, we construct the session MDP, which is organized by *session id*. For each session τ_m, the transition records are separated by the timestamps stored in the original dataset. This construction generates session MDP sequences organized by sequential user behaviors and has the advantage of overall loss weight updating.

For CTR/CTCVR prediction task, i.e., $K = 2$, assume we have a session-wise dataset $D_s = \{S_{\tau_m} 1_{\tau_m}, U_{\tau_m} 1_{\tau_m}, \mathbf{I}_{\tau_m}, (\mathbf{y}_{1,\tau_m}, \mathbf{y}_{2,\tau_m})\}_{m=1}^{M}$ consisting of M session records, where S_{τ_m} and U_{τ_m} are the m-th session and user *id*. $\mathbf{I}_{\tau_m}$ is item *id* vector whose dimension equals the item number that interacted with the user in m-th session. Each record stores the user-item information $\mathbf{x}_{\tau_m} = \{U_{\tau_m} 1_{\tau_m}, \mathbf{I}_{\tau_m}\}$ [2] and corresponding binary 0-1 label vector $(\mathbf{y}_{1,\tau_m}, \mathbf{y}_{2,\tau_m})$ for click and convert indicators. Use feature vector generated from mapping function f for further processing by state function$F$, then input to agent policy $\pi(s; \theta_k)$, which is a pretrained MTL network. The replay buffer $\mathcal{B}$ contains M session MDP sequences $\{(s_{\tau_m,1}, \cdots, s_{\tau_m,T_m})\}_{m=1}^{M}$, each of them have T_m transition information based on session τ. The detail of session MDP construction is shown in Algorithm 1. Specifically, we separate each session-wise data based on timestamps. Then, randomly sample a session and generate state information by the functions f and F (line 2). The state information of records in a specific session are sequentially input into the agent policy $\pi(s; \theta_k)$, which generates estimated

[2] 1_{τ_m} is the unit vector with dimension equals the item number in m-th session.

Algorithm 1 Session-wise CTR/CTCVR Prediction Setup

Input: Session-wise dataset organized as $\{S_{\tau_1} : (\mathbf{x}_{\tau_1}, \mathbf{y}_{1,\tau_1}, \mathbf{y}_{2,\tau_1}), \cdots, S_{\tau_M} : (\mathbf{x}_{\tau_M}, \mathbf{y}_{1,\tau_M}, \mathbf{y}_{2,\tau_M})\}$
Reorganize: Separate each session with state-label pairs denoted as $(f(x_{\tau_{m,t}}), y_{1,\tau_{m,t}}, y_{2,\tau_{m,t}})$, where f is a mapping function that returns the feature vector for any given user-item *id*

1: **while** Environment not closed **do**
2: **Reset**: randomly sample a session $\boldsymbol{\tau}_m$ and initialize state s_1 with $F(f(x_1))$, where F is state representation function.
3: **while** Session not done **do**
4: **Input:** Use action value $a_{1,t}, a_{2,t}$ from agent policy $\pi(s; \theta_k)$ as prediction values for CTR/CTCVR
5: **Step:** Compute the reward $r_{1,t}, r_{2,t}$ according to 2 with actions and labels, return next state s_{t+1}. Store the transition $(s_t, a_{1,t}, a_{2,t}, s_{t+1}, r_{1,t}, r_{2,t})$ into replay buffer $\mathcal{B}$
6: **if** $t + 1 > T_m$ **then**
7: Finish the session $\boldsymbol{\tau}_m$
8: **end if**
9: **end while**
10: **end while**

action pairs $(a_{1,t}, a_{2,t})$ (line 4). Then, we further calculate the reward value according to the Equation (2) and store the transition $(s_t, a_{1,t}, a_{2,t}, s_{t+1}, r_{1,t}, r_{2,t})$ at timestamp t into the experience replay buffer $\mathcal{B}$, which provides a more stable training environment (line 5). This completes the session-wise CTR/CTCVR prediction setup for the RL training.

2.4.1 ***State Representation Network.*** The original features of item-user pairs are categorical (represented by one-hot encoding format) or numerical. The state representation network is the combination of embedding layer and multi-layer perceptron to extract the user-item features, which corresponds to the mapping function F in Algorithm 1. The categorical features are firstly converted into binary vectors with length l_k, and then input to the embedding layer $R^{l_k} \rightarrow R^{d_e}$ with embedding dimension d_e. Besides, the numerical features are converted to the same dimension with d_e by a linear transformation. The features translated by the above process are combined and further used as input for the MLP network, which is the fully-connected neural network with multiple layers:

$$\begin{aligned} \mathbf{h}_{l+1} &= \sigma(\mathbf{W}_l \mathbf{h}_l + \mathbf{b}_l) \quad n = 0, 1, ..., N-1 \\ \mathbf{h}_L &= \sigma^*(\mathbf{W}_{L-1} \mathbf{h}_{L-1} + \mathbf{b}_{L-1}) \end{aligned} \tag{4}$$

where $\mathbf{h}_l$ is the l-th hidden layer with weight $\mathbf{W}_l$ and bias $\mathbf{b}_l$, the corresponding activation function σ is ReLU. The output layer is denoted by $\mathbf{h}_L$ with the sigmoid function σ^*. The output $\mathbf{h}_L$ is embedded with the extracted feature and represented by $F(f(x_t))$.

2.5 RMTL Framework

The RMTL aims to optimize the overall loss function with adaptive dynamic weights. In this direction, existing works [45] try to design the constant rate updating strategy for the loss of weight by:

$$\omega'_{k,t} = \omega'_{k,0} \gamma'^t \tag{5}$$

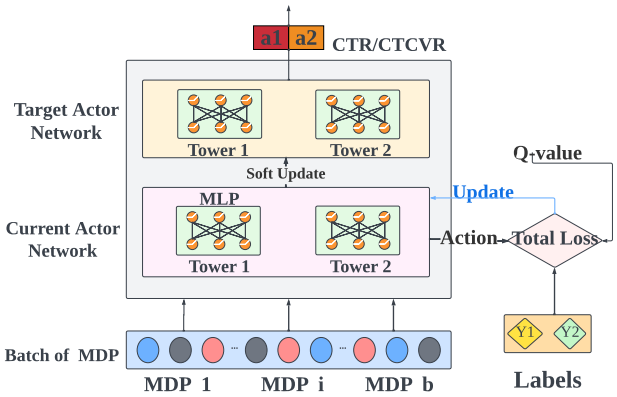

Figure 2: Structure of Actor Network.

where $\omega'_{k,t}$ is the weight for task k at timestamp t, and γ' is the constant decay rate. However, this weight is only controlled by time t and has no interaction with the type of task. To address this problem, we apply the critic network for evaluating the estimated total reward V_k of action a_k generated from agent policy for each task. Then, the value of loss function weight for each task is calculated as the linear transformation with polish variable λ. This novel overall loss design can optimize the MTL model parameter in a better direction, enhancing the efficiency and prediction performance.

The reinforcement learning-based CTR/CTCVR prediction training process consists of two major ingredients: actor network and task-specific critic network. The actor network is the main structure for predicting CTR/CTCVR in our work, which can be thought of as the agent policy $\pi(s_t; \theta_k)$ parameterized by θ_k. Based on this actor network, the agent can choose an action for each state. The critic network denoted by $Q(s_t, a_{k,t}; \phi_k)$ is the differentiable action value network updated by the gradient of Q-value and TD error δ, which can observe the potential rewards of the current state by learning the relationship between the environment and the rewards. Besides, it generates the adaptively adjusted weights of BCE loss for the objective function and updates the actor network parameter θ_k along the direction of better policy decisions.

2.5.1 ***Actor Network.*** The actor network in the RL setting can be referred to as a policy that is represented by a probability distribution $\pi(s; \theta_k) = \mathrm{P}(a|s, \theta_k)$. In practice, this method usually suffers from two problems: (i) The actor critic network performs poorly in a setting with a large action space, and (ii) the estimated agent policy is trained by its own expected action value that is not the ground truth data. In this section, we address the above two problems by applying a specific MTL model as the agent policy with deterministic action output, which reduces the capacity of action space. In addition, we update the neural network based network parameters using ground truth task labels. The structure of actor network is shown in Figure 2, which has a similar structure to specific MTL models, we may specify it using the ESMM model in this subsection. The shared-bottom layer in the original actor network design is removed and set as a state representation network mentioned in Sub-subsection 2.4.1 for generating replay buffer $\mathcal{B}$. Given a batch of MDP sequences from the replay buffer $\mathcal{B}$, we input the state information s_t into the actor network, which is two parallel neural

networks denoted by two tower layers parameterized by θ_1 and θ_2,

$$\pi(s_t;\theta_k) = a_{k,t}, \qquad k = 1,2 \tag{6}$$

The output of each tower layer is the deterministic action value, which represents the prediction for the specific task. In our setting, one tower outputs the CTR prediction value, and the other outputs the CTCVR prediction value. After the training process for the batch of MDP sequences, we calculate the overall loss function based on the weighted summation of BCE loss:

$$\mathcal{L}(\theta_1,\theta_2) = \sum_{(s_t,\cdots,s_{t+1},\cdots)\in b}\sum_{k=1}^{2}\omega_{k,t}BCE(\pi(s_t;\theta_k),y_{k,t})) \tag{7}$$

This differentiable overall loss function $\mathcal{L}(\theta_1,\theta_2)$ is controlled by binary cross entropy between estimated action and real task labels $y_{k,t}$, which enhances the decision accuracy of the policy.

2.5.2 ***Critic Network.*** The traditional critic network estimates the action value Q generated from actor network and then updates the actor network parameters based on value Q. While we choose the MTL model as the actor network, the problem is how to design a suitable critic network structure for multi-parameter updating. In this paper, we propose a multi-critic structure with two parallel MLP networks sharing one bottom layer. This design is capable of updating MTL model parameters in the actor network and polishing the loss function weights for specific tasks. The structure of the critic network is shown in Figure 3. The first part of the critic network is one shared-bottom layer, which transforms the user-item features and action information simultaneously. Similar to the state representation network in Sub-subsection 2.4.1, we apply one embedding layer and an MLP structure for feature extraction. We then combine the user-item feature and action information as the input of two differentiable action value networks parameterized by ϕ_k, which output the estimated Q-value based on the state-action information for each task. Given the current state s_t and action $a_{k,t}$, the Q-value is calculated by:

$$\begin{aligned} Q'(s_t,a_{k,t}) &= \mathbb{E}[r_{k,t}+\gamma V(s_{t+1})|s_t,a_{k,t}] \\ &= r_{k,t}+\gamma\sum_{s^{t+1}\in\mathcal{S}} p_{s_t,a_t,s_{t+1}}\cdot\max Q'(s_{t+1},a_{k,t+1}) \end{aligned} \tag{8}$$

where $V(s_{t+1})$ is state value function. In our case, the action value $a_{t,k}$ for task k is estimated by the actor network, and the next state s_{t+1} is determined with a probability equal to 1. Therefore, the Q-value function in multi critic structure can be calculated as:

$$Q(s_t,a_{k,t};\phi_k) = r_{k,t}+\gamma Q(s_{t+1},a_{k,t+1};\phi_k) \tag{9}$$

The weight of the objective loss function in Equation (1) is reversely adjusted along the direction of the Q value to improve the optimization process of actor network, which is the linear transformation with punish variable λ:

$$\omega_{k,t} = 1-\lambda * Q(s_t,a_{k,t};\phi_k) \tag{10}$$

where $a_{k,t}$ is an action for task k at timestamp t. $Q(s_t,a_{k,t};\phi_k)$ is action value of state s_t and action $a_{k,t}$.

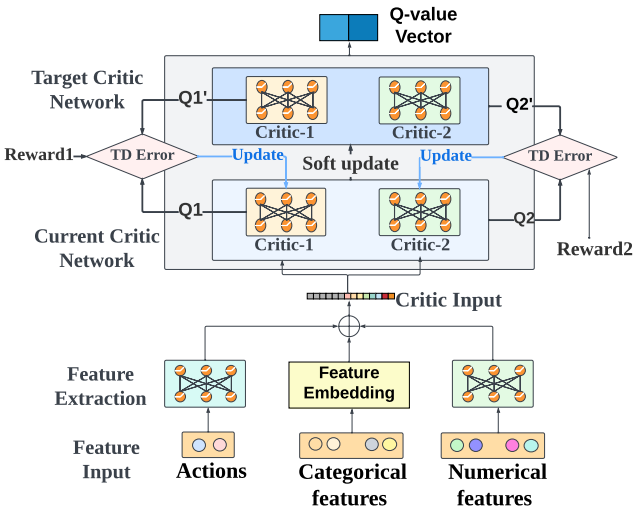

Figure 3: Structure of Critic Network.

2.6 Optimization

The general framework of the actor-critic network usually encounters an inevitable problem: the convergence of the critic network is not guaranteed. To overcome this problem, we leverage the idea of Deterministic Policy Gradient Algorithms [40], introducing the *target network* into the learning framework. The target networks have exact the same structure as the proposed actor and critic networks (i.e., *estimation networks*), and we denote them as $\pi(s_t;\widetilde{\theta}_k)$ and $Q(s_t,a_{k,t};\widetilde{\phi}_k)$, which have lagging parameters $\widetilde{\theta}_k$ and $\widetilde{\phi}_k$. In the rest part of this paper, we specify the main actor-critic network by estimation actor networks parameterized by θ_k and estimation critic networks parameterized by ϕ_k. The traditional actor-critic network usually suffers from the problem of parameter divergence and the convergence of the critic network is not guaranteed. The target network can address these problems by generating stationary target values from logging parameterized neuron networks, which smooths the training process. Next, we detail the optimization process for model parameters of the estimation actor-critic network, and also the soft update for target networks.

Estimation critic network updates. ϕ_k is the crucial parameter in the critic network, which determines action Q-value. Given transition $(s_t,a_{1,t},a_{2,t},s_{t+1},r_{1,t},r_{2,t})$ from replay buffer $\mathcal{B}$, the TD target of the target critic network for k-th task is derived from:

$$TD_{k,t} = r_{k,t}+\gamma Q(s_{t+1},a_{k,t+1};\widetilde{\phi}_k) \tag{11}$$

where $a_{k,t+1} = \pi(s_{t+1};\widetilde{\theta}_k)$ is the estimated next action from target actor network. The Q-value generated from the estimation critic network, which estimates the current action value is defined as:

$$Q_{k,t} = Q(s_t,a_{k,t};\phi_k) \tag{12}$$

After the training process among b batch of transitions from the replay buffer, we calculate the average TD error δ:

$$\begin{aligned}\delta &= \frac{1}{2|b|}\sum_{(s_t,\cdots,s_{t+1},\cdots)\in b}\sum_{k=1}^{2}(TD_{k,t}-Q_{k,t}) \\ &= \frac{1}{2|b|}\sum_{(s_t,\cdots,s_{t+1},\cdots)\in b}\sum_{k=1}^{2} \\ &\quad [r_{k,t}+\gamma Q(s_{t+1},a_{k,t+1};\widetilde{\phi_k})-Q(s_t,a_{k,t};\phi_k)]\end{aligned} \tag{13}$$

Then we update the current critic network for each task by the following gradient decent method:

$$\phi_k^{t+1}=\phi_k^t-\alpha^{\phi}\mathrm{I}\delta\nabla_{\phi_k}Q(s_t,a_{k,t};\phi_k) \tag{14}$$

where $\nabla_{\phi_k}Q(s_t,a_{k,t};\phi_k)$ is the gradient of k-th target Q-value. This completes the optimization of the estimation critic networks.

Estimation actor network updates. Before the TD error δ converges to threshold ϵ, we update the current actor networks parameterized by θ_k through the gradients back-propagation of loss function for each tower layer after the forward process of each batch transitions from $\mathcal{B}$:

$$\theta_k^{t+1}=\theta_k^t+\alpha^{\theta}\mathrm{I}\nabla_{\theta_k}\mathcal{J}(\theta_k) \tag{15}$$

where the loss for tower layers is defined by the negative of average Q-value $\mathcal{J}(\theta_k)=-\frac{1}{b}\sum_{(s_t,\cdots,s_{t+1},\cdots)\in b}Q'(s_t,\pi(s_t;\theta_k))$. We further update the current actor networks with respect to the gradients of the overall loss function of CTR/CTCVR prediction:

$$\theta_k^{t+1}=\theta_k^t-\alpha^{\theta}\mathrm{I}\nabla_{\theta_k}\mathcal{L}(\theta_1,\theta_2) \tag{16}$$

$$\mathcal{L}(\theta_1,\theta_2)=\sum_{(s_t,\cdots,s_{t+1},\cdots)\in b}\sum_{k=1}^{2}\omega_{k,t}BCE(\pi(s_t;\theta_k),y_{k,t}) \tag{17}$$

where the weights $\omega_{k,t}=1-\lambda*Q(s_t,a_{k,t};\phi_k)$ controlled by the corresponding current critic network, which is adaptively adjusted to the negative direction of estimated action Q-value at t.

Soft update of the target network. The target actor network and two specific target critic networks are updated until the current critic network reaches the convergence condition towards the direction of parameters in current networks:

$$\begin{aligned}\widetilde{\theta_k}&=\beta\widetilde{\theta_k}+(1-\beta)\theta_k \\ \widetilde{\phi_k}&=\beta\widetilde{\phi_k}+(1-\beta)\phi_k\end{aligned} \tag{18}$$

where $\beta\in[0,1]$ is the soft update rate.

3 EXPERIMENT

In this section, we conduct several experiments using two real-world datasets to evaluate the effectiveness of the RMTL framework.

3.1 Experimental Setup

3.1.1 ***Dataset.*** From our survey, there is only one open-source dataset, RetailRocket[3], that contains sequential labels of click and pay. In order to have a comprehensive comparison of RMTL, we also apply the "Kuairand-1K" dataset[4] for further analysis. Each dataset is split into training, validation, and testing sets with proportion 6:2:2 sorted by timestamp t for model learning.

3.1.2 ***Baselines.*** Since our RMTL structure modifies the MTL objective loss function, we choose the MTL models with their default losses and one RL-based model as our baselines. **Single Task** [28]: The naive method to learn each task separately, which is widely used to compare the performance of MTL models. **Shared Bottom** [29]: The basic structure of MTL models with a shared bottom layer. **ESMM** [30]: A model specified in CTR/CVR predictions that focus on the different sample spaces for each task. **MMoE** [29]: MMoE uses gates to control the relationship between the shared bottom and the task-specific towers. **PLE** [45]: PLE is able to capture complicated task correlations by using shared experts as well as task-specified expert layers. **D-PLE** [40]: We apply DDPG (deep deterministic policy gradient) with the PLE model as the RL-based baseline. Actually, all the loss functions for the baseline models are obtained by taking the unchangeable weighted average.

3.1.3 ***Evaluation Metrics.***

- **AUC** and **Logloss**: These are the straightforward metrics to evaluate CTR/CTCVR prediction tasks. According to [15], it is statistically significant in recommendation tasks that AUC or Logloss changes more than 0.001-level.
- **s-Logloss**: This metric is defined as the averaged Logloss with respect to all sessions.

3.2 Implementation Details

We implement the baseline models using the public library[5], which integrates 7 standard MTL models (without ESMM) and gives a reference performance of each model. All the models have the same basic network structure, i.e., an input embedding layer with dimension 128, a $128\times512\times256$ Multi-Layer Perceptron (MLP) as the bottom layer, and a $256\times128\times64\times1$ MLP as the tower layer. Specifically, for the models using expert layers (same structure as the bottom layer), we fix the number of experts as 8. The activation function for the hidden layers is ReLU and Sigmoid for the output. We also set a 0.2 dropout rate. We use Adam optimizer ($lr=1e-3$) to learn all the models. We also implement ESMM sharing the hyperparameters of the above models.

Since the actor network in our RMTL model is a two-tower design with multi-task output, it is compatible with most existing MTL-based recommendation models. In this paper, we apply our RMTL structure on three representative MTL models: ESMM, MMoE, and PLE. As to the RMTL structure, we set the Actor network with the same structure as specified MTL models. The Critic networks also share the same bottom layer and have their out through one tower layer. Moreover, we add a 1×128 action layer to be consistent with the input features, and we change the output activation to negative ReLU in that our reward only has negative values. In order to improve data efficiency, we initialize the Actor with pretrained parameters from the MTL models and keep them frozen until the Critics converge. Then we multiply the critic value in the total loss and retrain the MTL model. The default learning rates of the actor α^{θ} and critic network α^{ϕ} are set as 0.001. We set the default soft update rate $\beta=0.2$, punish variable $\lambda=0.7$. The implementation code is available online to ease reproducibility[6].

[3]https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset
[4]https://kuairand.com/
[5]https://github.com/easezyc/Multitask-Recommendation-Library.git
[6]https://github.com/Applied-Machine-Learning-Lab/RMTL

Table 1: Performance on CTR/CTCVR tasks for different methods.

Dataset	Task	Metric	Methods								
			Single Task	Shared Bottom	ESMM	MMoE	PLE	D-PLE	RMTL-ESMM	RMTL-MMoE	RMTL-PLE
RetailRocket	CTR	AUC ↑	0.7273	0.7287	0.7282	0.7309	0.7308	0.7308	0.7338*	**0.7350***	0.7339*
		Logloss ↓	0.2065	0.2048	0.2031	0.2021	0.2056	0.2058	0.2024	**0.1995**	0.2013
		s-Logloss ↓	0.0846	0.0839	0.0852	0.0853	0.0827	0.0830	0.0836	0.0848	**0.0824**
RetailRocket	CTCVR	AUC ↑	0.7250	0.7304	0.7316	0.7347	0.7387	0.7386	0.7341	0.7396	**0.7419***
		Logloss ↓	0.0489	0.0493	0.0486	0.0496	0.0486	0.0490	0.0485	0.0490	**0.0480**
		s-Logloss ↓	0.0150	0.0149	0.0150	0.0145	0.0147	0.0149	0.0150	**0.0143**	0.0146
Kuairand	CTR	AUC ↑	0.7003	0.7018	0.7009	0.7014	0.7026	0.7025	0.7031	0.7029	**0.7053***
		Logloss ↓	0.6127	0.6114	0.6128	0.6119	0.6111	0.6123	0.6111	0.6105	**0.6092***
		s-Logloss ↓	0.6263	0.6250	0.6261	0.6255	0.6252	0.6257	0.6252	0.6243	**0.6242**
Kuairand	CTCVR	AUC ↑	0.7342	0.7310	0.7350	0.7324	0.7339	0.7339	**0.7377***	0.7345	0.7367*
		Logloss ↓	0.5233	0.5249	0.5220	0.5237	0.5235	0.5237	**0.5200***	0.5225	0.5221
		s-Logloss ↓	0.5449	0.5468	0.5436	0.5450	0.5454	0.5451	**0.5433**	0.5446	0.5447

↑: higher is better; ↓: lower is better. Underline: the best baseline model. **Bold**: the best performance among all models.
"*": the statistically significant improvements (i.e., two-sided t-test with $p < 0.05$) over the best baseline.

3.3 Overall Performance and Comparison

We compare the performance of five baseline MTL models with RMTL models for CTR/CTCVR prediction tasks on two different datasets. The overall performance on CTR/CTCVR tasks for different methods is shown in Table 1. It can be observed that:

- The SingleTask model achieves the worst performance in both prediction tasks on two datasets. The feature representation network and loss function optimization are separated for each task, which fails to detect the intrinsic relation between multiple tasks. Thus, we can understand that the shared-bottom method, which shares the feature extraction parameters, can outperform the SingleTask method for both datasets.
- The PLE model achieves almost the best performance among all MTL baseline models in most cases. On the basis of the MMoE model, which controls the parameter interaction between the shared bottom and specific task tower, the PLE model specifies the parameter sharing between expert layers to extract hidden interaction patterns between each task. This demonstrates that the PLE baseline model can improve the efficiency of information sharing among tasks to achieve better prediction performance.
- Each version of our proposed RMTL model outperforms the corresponding non-RL version baseline model (e.g., RMTL-ESMM v.s. ESMM) in both prediction tasks with AUC and logloss metrics on two datasets. Especially on the RetialRocket dataset, the RMTL model achieves around 0.003-0.005 AUC gains compared with the corresponding baseline model. By leveraging the sequential property of the reinforcement learning framework, the RL-enhanced method is capable of processing session-wise recommendation data and achieves significant improvement in CTR/CTCVR prediction tasks by adaptively adjusted loss function weights.
- The CTCVR prediction result on the Kuairand dataset presented in the fourth row of Table 1, where the ESMM baseline model and its advanced RL version achieve better performance than other MTL models. For short video recommendations, click-and-convert behaviors usually have a relatively higher correlation. In this phenomenon, the ESMM model outperforms MMoE and PLE by overcoming the sample selection bias.
- The RMTL models achieve s-Logloss improvement of less than 0.001 compared with the corresponding baseline models, which is lower than that of Logloss. This phenomenon may be caused by the fact that session-based metric has similar performance with online A/B test, which is more difficult to be improved [18]. Note that the ratio of logloss and s-logloss is different for the two datasets since they have different average session lengths while the average session Logloss is affected by the average session length for a specific dataset.

To summarize, the RMTL model outperforms the state-of-the-art MTL models on both CTR and CTCVR prediction tasks for different real-world datasets. In addition, it can be employed in most MTL models, validating its desirable compatibility.

3.4 Transferability Study

This subsection presents the transferability study for the RMTL method for the CTR task on the RetailRocket dataset. We try to figure out whether the critic network learned from different logging policies can be applied to the same MTL baseline model and improve the prediction performance. For each MTL baseline model, we leverage the model parameters from the critic network pretrained by ESMM, MMoE, and PLE model on the RetailRocket dataset. The results of AUC and logloss are presented in Figure 4 (a)-(c) and (d)-(f), where "mmoe-ESMM" means ESMM model applying critic network trained from MMoE and "ple-ESMM" means ESMM model applying critic network trained from PLE. It can be observed that: (i) The pretrained critic network from three MTL models can significantly increase AUC for each baseline model. (ii) The pretrained

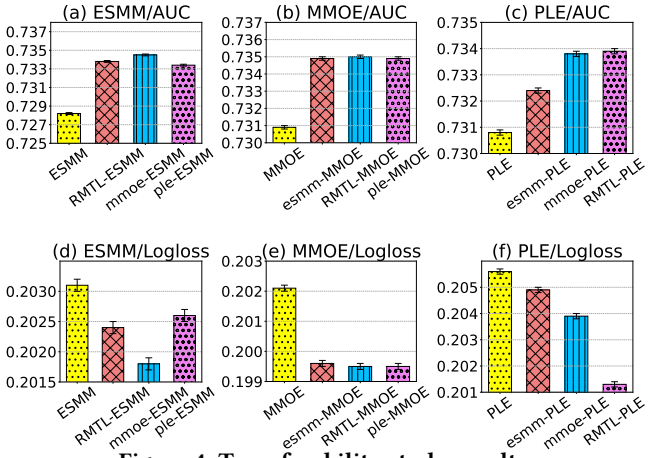

Figure 4: Transferability study results.

Table 2: Ablation study on RetailRocket.

Task	Metric	Methods			
		CW	WL	NLC	RMTL-PLE
CTR	AUC ↑	0.7308	0.7323	0.7325	**0.7339***
	Logloss ↓	0.2056	0.2218	0.2020	**0.2013** *
CTCVR	AUC ↑	0.7387	0.7411	0.7418	**0.7419***
	Logloss ↓	0.0486	0.0485	0.0481	**0.0480***

"*" indicates the best performance among all variants.

critic network from three MTL models can significantly decrease logloss for each baseline model.

To summarize, the pretrained critic network is capable of improving the prediction performance of most MTL models, which demonstrates the excellent transferability of the RMTL model.

3.5 Ablation Study

The most essential design of the RMTL model is the overall loss function with learnable weights, which is the linear combination of Q-value from the critic network. In this subsection, in order to figure out the importance of this loss function in the proposed model, we change some components and define three variants below: (i) **CW:** The CW variant indicates applying constant weights for the overall loss function and no gradient policy update for the actor network, which eliminates the contribution of the critic network. We assign all weights equal to constant 1. (ii) **WL:** The WL variant indicates the adjustment of weights ω is controlled by session behavior labels y_k multiplied by Q-value. (iii) **NLC:** No linear transformation is performed to the loss weights, instead, we directly assign the negative Q-value to loss weights.

The ablation study results for the PLE model on the RetailRocket dataset are shown in Table 2. It can be observed that: (i) CW has the worst performance for AUC and logloss metrics on both prediction tasks. It may be caused by equivalently assigned BCE loss weights, which fail to capture the hidden dynamics of multi-tasks. (ii) WL and NLC have almost the same performance in this study that outperform the CW variant with 0.002-0.003 AUC gain. This demonstrates the effect of weights updating from the critic network. (iii) RMTL-PLE with our proposed total loss setting achieves the best performance on both tasks, which illustrates the validity of our linear combination weight design. We may summarize that our proposed MTL model can achieve better CTR/CTCVR prediction performance, where total loss design has great contributions.

4 RELATED WORK

In this section, we give a brief introduction to existing works related to RMTL, i.e., RL-based and MTL-based recommender systems.

Reinforcement Learning Based Recommender Systems. Plenty of research has been done combining Reinforcement Learning (RL) [2, 42, 47, 49, 52, 54, 56] into the field of Recommender Systems (RS). Compared to traditional learning-to-rank solutions of RS [26] that optimize the immediate user feedback, the RL-based RS focuses on optimizing the cumulative reward function that estimates multiple rounds of interactions between the recommendation policy and the user response environment. The general problem formulates the sequential user-system interactions as a Markov Decision Process (MDP) [39]. RL solutions have been studied under this formulation including early-age tabular-based methods [21, 31, 33] that optimize an evaluation table for a set of state-action pairs, value-based methods [19, 44, 55, 58] that learn to evaluate the quality of an action or a state, policy gradient methods [6, 7, 41] that optimize the recommendation policy based on the long-term reward, and the actor-critic methods [23] based on policy gradient method [4, 10, 37, 43] that simultaneously learn an action evaluator and an action generator. Our method falls into the actor-critic category and extends it toward the multi-task problem setting. The main challenges of applying RL for the recommendation task consist of the large combinatorial state/actions space [11, 24], the uncertainty of user environment [20, 53], the stability and efficiency of exploration [3, 8], and the design of a user's reward function over heterogeneous behaviors [59]. Our work is related to the reward function design but also enhances the performance of each task.

Multi-Task Learning in Recommender Systems. As stated in [51], Multi-Task Learning (MTL) is a machine learning framework that learns a task-invariant representation of an input data in a bottom network, while each individual task is solved in one's respective task-specific network and boosted by the knowledge transfer across tasks. Recently, MTL has received increasing interest in recommender systems [16, 27, 30, 35, 36] due to its ability to share knowledge among different tasks especially its ability to capture heterogeneous user behaviors. A series of works seek to improve on it by designing different types of shared layer architectures. These works either introduce constraints on task-specific parameters [12, 32, 48] or separate the shared and the task-specific parameters [29, 45]. The general idea is to disentangle and share knowledge through the representation of the input feature. Additionally, there are also researches on applying multi-agent RL for the multi-scenario setting [13] where the recommendation task is bundled with other tasks like search, and target advertising. Different from the above ideas, we resort to knowledge distillation to transfer ranking knowledge across tasks on task-specific networks and we combine RL to improve the long-term satisfaction of users. Notably, our model is a general framework and could be leveraged as an extension for most off-the-shelf MTL models.

5 CONCLUSION

In this paper, we propose a novel multi-task learning framework, RMTL, to improve the prediction performance of multi-tasks by generating dynamic total loss weights in an RL manner. The RMTL model can adaptively modify the weights of BCE for each prediction task by Q-value output from the critic network. By constructing a session-wise MDP environment, we estimate the multi-actor-critic networks using a specific MTL agent and then polish the optimization of the MTL overall loss function using dynamic weight, which is the linear transformation of the critic network output. We conduct several experiments on two real-world commercial datasets to verify the effectiveness of our proposed method with five baseline MTL-based recommendation models. The results demonstrate that RMTL is compatible with most existing MTL-based recommendation models and can improve multi-task prediction performance with excellent transferability.

ACKNOWLEDGEMENTS

This research was partially supported by APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for New Faculty of City University of Hong Kong), SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074), HKIDS Early Career Research Grant (No.9360163), Huawei Innovation Research Program and Ant Group (CCF-Ant Research Fund).

REFERENCES

[1] Giuseppe Aceto, Valerio Persico, and Antonio Pescapé. 2020. Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0. *Journal of Industrial Information Integration* 18 (2020), 100129.

[2] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2021. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys (CSUR)* (2021).

[3] Xueying Bai, Jian Guan, and Hongning Wang. 2019. A model-based reinforcement learning with adversarial training for online recommendation. *Advances in Neural Information Processing Systems* 32 (2019).

[4] Shalabh Bhatnagar, Mohammad Ghavamzadeh, Mark Lee, and Richard S Sutton. 2007. Incremental natural actor-critic algorithms. *Advances in neural information processing systems* 20 (2007).

[5] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[6] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3312–3320.

[7] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[8] Minmin Chen, Bo Chang, Can Xu, and Ed H Chi. 2021. User response models to improve a reinforce recommender system. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 121–129.

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[10] Thomas Degris, Patrick M Pilarski, and Richard S Sutton. 2012. Model-free reinforcement learning with continuous action in practice. In *2012 American Control Conference (ACC)*. IEEE, 2177–2182.

[11] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679* (2015).

[12] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*. 845–850.

[13] Jun Feng, Heng Li, Minlie Huang, Shichen Liu, Wenwu Ou, Zhirong Wang, and Xiaoyan Zhu. 2018. Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning. In *Proceedings of the 2018 World Wide Web Conference*. 1939–1948.

[14] Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis lectures on human language technologies* 10, 1 (2017), 1–309.

[15] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[16] Guy Hadash, Oren Sar Shalom, and Rita Osadchy. 2018. Rank and rate: multi-task learning for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 451–454.

[17] Yanxiang Huang, Bin Cui, Jie Jiang, Kunqian Hong, Wenyu Zhang, and Yiran Xie. 2016. Real-time video recommendation exploration. In *Proceedings of the 2016 International Conference on Management of Data*. 35–46.

[18] Guangda Huzhang, Zhen-Jia Pang, Yongqing Gao, Wen-Ji Zhou, Qing Da, Anxiang Zeng, and Yang Yu. 2020. Validation Set Evaluation can be Wrong: An Evaluator-Generator Approach for Maximizing Online Performance of Ranking in E-commerce. *CoRR* abs/2003.11941 (2020). https://arxiv.org/abs/2003.11941

[19] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A tractable decomposition for reinforcement learning with recommendation sets. (2019).

[20] Eugene Ie, Chih wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. RecSim: A Configurable Simulation Platform for Recommender Systems. (2019). arXiv:1909.04847 [cs.LG]

[21] Thorsten Joachims, Dayne Freitag, Tom Mitchell, et al. 1997. Webwatcher: A tour guide for the world wide web. In *IJCAI (1)*. Citeseer, 770–777.

[22] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[23] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027* (2018).

[24] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, Yuzhou Zhang, and Xiuqiang He. 2020. State representation modeling for deep reinforcement learning based recommendation. *Knowledge-Based Systems* 205 (2020), 106170.

[25] Hu Liu, Jing Lu, Xiwei Zhao, Sulong Xu, Hao Peng, Yutong Liu, Zehua Zhang, Jian Li, Junsheng Jin, Yongjun Bao, et al. 2020. Kalman filtering attention for user behavior modeling in ctr prediction. *Advances in Neural Information Processing Systems* 33 (2020), 9228–9238.

[26] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[27] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Coevolutionary recommendation model: Mutual learning between ratings and reviews. In *Proceedings of the 2018 World Wide Web Conference*. 773–782.

[28] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114* (2015).

[29] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.

[30] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1137–1140.

[31] Tariq Mahmood and Francesco Ricci. 2007. Learning and adaptivity in interactive recommender systems. In *Proceedings of the ninth international conference on Electronic commerce*. 75–84.

[32] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3994–4003.

[33] Omar Moling, Linas Baltrunas, and Francesco Ricci. 2012. Optimal radio channel recommendations with explicit and implicit feedback. In *Proceedings of the sixth ACM conference on Recommender systems*. 75–82.

[34] Raymond J Mooney and Loriene Roy. 2000. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*. 195–204.

[35] Huashan Pan, Xiulin Li, and Zhiqiang Huang. 2019. A Mandarin Prosodic Boundary Prediction Model Based on Multi-Task Learning.. In *Interspeech*. 4485–4488.

[36] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. 2019. Value-aware recommendation based on reinforcement profit maximization. In *The World Wide Web Conference*. 3123–3129.

[37] Jan Peters and Stefan Schaal. 2008. Natural actor-critic. *Neurocomputing* 71, 7-9 (2008), 1180–1190.

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 28 (2015).

[39] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).

[40] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*. Pmlr, 387–395.

[41] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st international acm sigir conference on research & development in information retrieval*. 235–244.

[42] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[43] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[44] Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. 2007. Usage-based web recommendations: a reinforcement learning approach. In *Proceedings of the 2007 ACM conference on Recommender systems*. 113–120.

[45] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM Conference on Recommender Systems*. 269–278.

[46] Nelson Vithayathil Varghese and Qusay H Mahmoud. 2020. A survey of multi-task deep reinforcement learning. *Electronics* 9, 9 (2020), 1363.

[47] Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed H Chi, and Minmin Chen. 2022. Surrogate for Long-Term User Experience in Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4100–4109.

[48] Yongxin Yang and Timothy Hospedales. 2016. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391* (2016).

[49] Qihua Zhang, Junning Liu, Yuzhuo Dai, Yiyan Qi, Yifan Yuan, Kunlun Zheng, Fan Huang, and Xianfeng Tan. 2022. Multi-Task Fusion via Reinforcement Learning for Long-Term User Satisfaction in Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4510–4520.

[50] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[51] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[52] Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Hui Liu, and Jiliang Tang. 2021. DEAR: Deep Reinforcement Learning for Online Advertising Impression in Recommender Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 750–758.

[53] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. 2019. " Deep reinforcement learning for search, recommendation, and online advertising: a survey" by Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin with Martin Vesely as coordinator. *ACM sigweb newsletter* Spring (2019), 1–15.

[54] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. In *Proceedings of the 12th ACM Recommender Systems Conference*. ACM, 95–103.

[55] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1040–1048.

[56] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2017. Deep Reinforcement Learning for List-wise Recommendations. *arXiv preprint arXiv:1801.00209* (2017).

[57] Yufan Zhao, Donglin Zeng, Mark A Socinski, and Michael R Kosorok. 2011. Reinforcement learning strategies for clinical trials in nonsmall cell lung cancer. *Biometrics* 67, 4 (2011), 1422–1433.

[58] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*. 167–176.

[59] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.