Learning to Collaborate in Multi-Module Recommendation via Multi-Agent Reinforcement Learning without Communication

Xu He, Bo An
Nanyang Technological University
{hexu0003,boan}@ntu.edu.sg

Yanghua Li, Haikai Chen
Alibaba Group
{yichen.lyh,haikai.chk}@taobao.com

Rundong Wang, Xinrun Wang, Runsheng Yu
Nanyang Technological University
{rundong001,xinrun.wang,runsheng.yu}@ntu.edu.sg

Xin Li, Zhirong Wang
Alibaba Group
xin.l@alibaba-inc.com,qingfeng@taobao.com

ABSTRACT

With the rise of online e-commerce platforms, more and more customers prefer to shop online. To sell more products, online platforms introduce various modules to recommend items with different properties such as huge discounts. A web page often consists of different independent modules. The ranking policies of these modules are decided by different teams and optimized individually without cooperation, which might result in competition between modules. Thus, the global policy of the whole page could be sub-optimal. In this paper, we propose a novel multi-agent cooperative reinforcement learning approach with the restriction that different modules cannot communicate. Our contributions are three-fold. Firstly, inspired by a solution concept in game theory named correlated equilibrium, we design a signal network to promote cooperation of all modules by generating signals (vectors) for different modules. Secondly, an entropy-regularized version of the signal network is proposed to coordinate agents' exploration of the optimal global policy. Furthermore, experiments based on real-world e-commerce data demonstrate that our algorithm obtains superior performance over baselines.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Multi-agent systems**.

KEYWORDS

Reinforcement learning

ACM Reference Format:
Xu He, Bo An, Yanghua Li, Haikai Chen, Rundong Wang, Xinrun Wang, Runsheng Yu, and Xin Li, Zhirong Wang. 2020. Learning to Collaborate in Multi-Module Recommendation via Multi-Agent Reinforcement Learning without Communication. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 21–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383313.3412233

RecSys '20, September 21–26, 2020, Virtual Event, Brazil

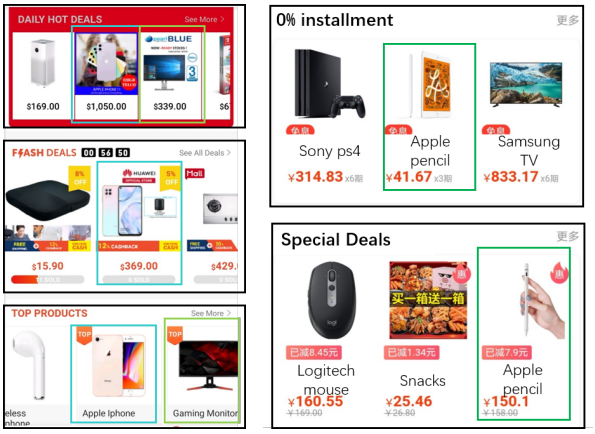
ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412233

1 INTRODUCTION

The web pages of many online e-commerce platforms consist of different modules. Each of the modules shows items with different properties. As an example, consider the web pages depicted in Fig. 1. The page on the left includes three modules: the daily hot deals, the flash sales, and the top products. There are two modules in the page on the right: the 0% installment and the special deals. The candidate items of each module are selected according to predefined conditions. For instance, the top products module includes the best selling items in the period of the past few days. The items in the flash sales module and the special deals module offer special discounts provided by qualified shops, either daily or hourly. Because several modules are shown to users at the same time, the interaction between modules affects the users' experience.

However, different teams are usually in charge of ranking strategies of different modules. Due to the lack of cooperation between the teams, the whole page suffers from competition between different modules. As a consequence, the users might find the same product or category in multiple modules, which wastes the limited space on the page. For example, the phones appear in all modules in Fig. 1(a) and the apple pencil is recommended by two modules in Fig. 1(b).

To find the optimal global strategy, it is crucial to design a proper cooperation mechanism. Multi-agent reinforcement learning (RL) algorithms are proposed to solve the recommendation problems that involve sequential modules [3, 28]. However, their approaches rely on an underlying communication mechanism. Each agent is hence required to send and receive messages during the execution. This might be a problem as ranking strategies of different modules are usually deployed by different teams in real-time and the modules cannot communicate with each other. There are many examples of multi-agent RL algorithms in the literature which do not need communication. However, their performance suffers a lot from their inability to coordinate, as we illustrate in the experiments. In this paper, we propose a novel approach for the multi-module recommendation problem. The first key contribution of this paper is a novel multi-agent cooperative reinforcement learning structure. The structure is inspired by a solution concept in game theory called correlated equilibrium [1] in which the predefined signals received by the agents guide their actions. In our algorithm, we propose to use a signal network to maximize the global utility by taking the information of a user as input and sending signals to different modules. The signal network can act as a high-level leader coordinating the individual agents. All agents act solely on the basis of their signals, without any communication.



(a) Example 1 (b) Example 2

Figure 1: Two examples of the multi-module recommendation scenarios. The black boxes represent modules. Boxes in different colors mark similar items in different modules. In subfigure 1(a), phones and monitors appear more than once. Meanwhile, apple pencils are recommended by two modules in subfigure 1(b).

The second key contribution is an entropy-regularized version of the signal network to coordinate agents' exploration. Since the state and action spaces are huge, exploration remains essential in finding the optimal policy. We add the entropy terms to the loss function of the signal network to encourage exploration in view of the global performance. In contrast, the agents in the existing work [9] explore individually. To maximize the entropy term, the distributions of signals should be flat. In that case, the diverse signals encourage agents to explore more when the global policy converges to a sub-optimal solution.

Third, we conduct extensive experiments on a real-world dataset from Taobao, one of the largest e-commerce companies in the world. Our proposed method outperforms other state-of-the-art cooperative multi-agent reinforcement learning algorithms. Moreover, we show the improvement caused by the entropy term in the ablation study.

2 RELATED WORK

We briefly review works that apply RL methods in recommender systems and introduce the concept of correlated equilibrium in this section.

Many deep reinforcement learning methods are used in the recommender system domain. The works focusing on the single-agent setting mainly consider three aspects: 1) the different kinds of rewards, 2) the structures of web pages and 3) the large space of actions. DRN updates periodically after obtaining long term-reward such as return time [29]. An algorithm is proposed to use two individual LSTM modules for items with short-term and long-term rewards respectively [30]. The diversity of recommended sets is added to the reward function [15]. Transition probabilities from users' actions (such as click) to purchase are used as rewards [19]. Modified MDPs for recommendation are proposed by redefining the structure of reward function and the transition function respectively [7, 8]. A method is proposed to improve profit by detecting fraud transactions [25]. Second, page structures including different types of content and positions of items are taken into consideration. A CNN-based approach is proposed to recommend items according to their positions on the web page [27]. A hierarchical algorithm is proposed to aggregate topics, blog posts, and products on one web page [22]. Similarly, the Double-Rank Model is proposed to learn how to rank the display positions and with which documents to fill those positions [18]. The problem 'when and where should advertising be added?' is addressed for web pages that contain advertising [26]. Third, other works focusing on the large space of actions and states usually adopt clustering techniques to reduce the space [2, 21]. To decide which items to recommend, the policy network outputs the feature of an ideal item and clustering methods are adopted to find neighbors of the ideal item in the candidate set of items. However, these works do not consider recommendation problems that involve more than one agent and thus cannot be used to solve our problem.

The most similar works use multi-agent frameworks to promote cooperation between different pages. Inspired by RL methods involving communication like [23], a multi-agent RL framework is proposed where agents can communicate by messages [3]. A model-based RL algorithm is proposed by using neural networks to predict transition probability and shape reward [28]. Differing from our setting, the agents in their works recommend items for different pages (e.g., the entrance page and the item detail page), and execute sequentially rather than simultaneously. It means that agents can send messages to others when users leave one page and enter another page. Moreover, the immediate reward is only related to one page (module). However, our problem considers cooperation between different modules on one page in which agents cannot communicate during execution, and the immediate rewards are determined by more than one module.

Correlated equilibrium is a solution concept in game theory, which is first discussed by Aumann [1]. The idea is that each player or agent chooses strategy according to their observation and a signal. The signal usually is a recommended strategy that assigns actions to all agents. If the expected payoff from playing the recommended strategy is no worse than playing any other strategy, it is called correlated equilibrium. An example is the traffic light, which suggests to each player whether to go or stop. Following its advice is the best response for everyone involved. A simple RL algorithm is proposed to find the correlated equilibrium in an existing work [5]. In our work, we use a neural network to learn how to send signals inspired by this concept.

3 PROBLEM STATEMENT AND FORMULATION

We firstly introduce the details of the multi-module recommendation problem. Fig. 2 shows three stages in a recommendation session. First, when a user enters the recommendation scenario, he firstly browses the *entrance page*, which contains more than one module. The ranking strategy of each module recommends items from its candidate set depending on users' information. A list of items is ranked and the top-3 will be shown on the entrance page. The user can 1) go to the *module page* if he clicks any module, or 2) refresh the web page to access new items shown in modules, in which ranking strategies are called again to rank items. Second, the *module*

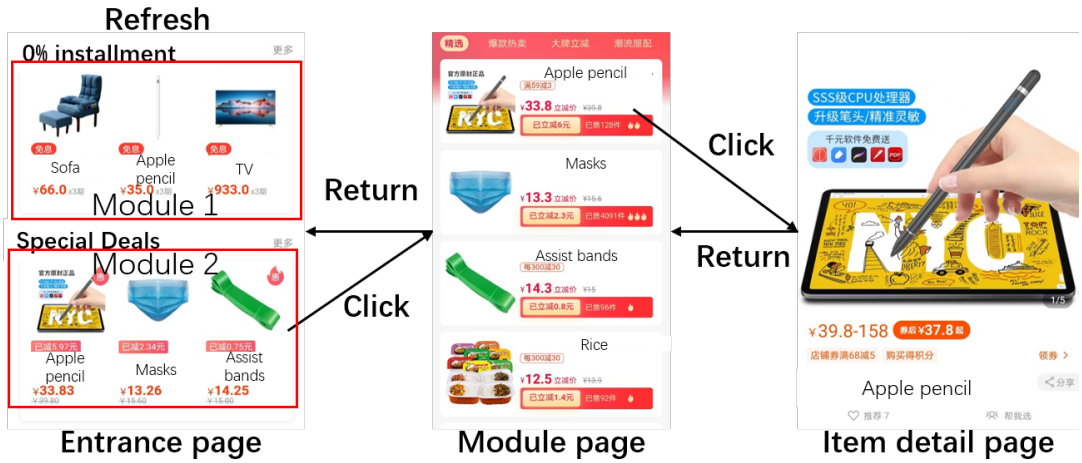


Figure 2: The flow of a multi-module recommendation system. Pages shown in this figure are the entrance page, the module page, and the item detail page. The entrance page contains two modules.

page shows a list of recommended items for this module and the first three items are consistent with the items showing on the entrance page. The user can 1) slide the screen to browse more items, 2) go to the *item detail page* by clicking an item, or 3) return to the entrance page. The agent will recommended more items if the whole list is browsed. Third, the *item detail page* demonstrates the details of an item. The user can 1) purchase the item, or 2) return to the module page. The recommended items do not change when the user returns to the module page and he can continue to explore more preferred items by sliding the screen.

Since different modules aim to collectively maximize the global performance, we can model our problem as a multi-agent extension of Markov Decision Processes (MDPs) [13]. Formally, the MDP for multiple agents is a tuple consisting of five elements $\langle N, S, A, R, P \rangle$:

Agent N is the number of agents. We treat modules as different agents rather than pages in existing works [3, 28].

State S includes information that each agent has received about users. In our problem, s is the information of users which contains: 1) static features such as age, gender, and address. 2) sequential features $[h_1, \ldots, h_K]$ including features of K items that a user purchased or clicked recently.

Action $A = [A^1, \ldots, A^N]$ is a set including the action sets of each agent. Specifically, $a = [a^1, \ldots, a^N]$, where $a^i \in A^i$ is the action of the agent i. The action of each agent is defined as a weight vector that determines the rank of candidate items. Formally, the j-th element of the i-th agent's action $a^i = [a^i_1, \ldots, a^i_j, \ldots]$ is the weight of the j-th element of the item's feature. The weighted sum of the action and an item's feature determines the rank of the item, that is $score_{item} = a^T e_{item}$, where e_{item} is the embedding of an item's features.

Reward $R = [R^1, \ldots, R^N]$, where $R^i : S \times A \to \mathbb{R}$ is the reward function for agent i. After agents take action a at the state s, the user would provide feedback like clicking an item or skipping the module, which can be converted to reward. The global reward r will be obtained according to the reward function $r = R(s, a)$, where $r = [r^1, \ldots, r^N]$ including rewards for N agents.

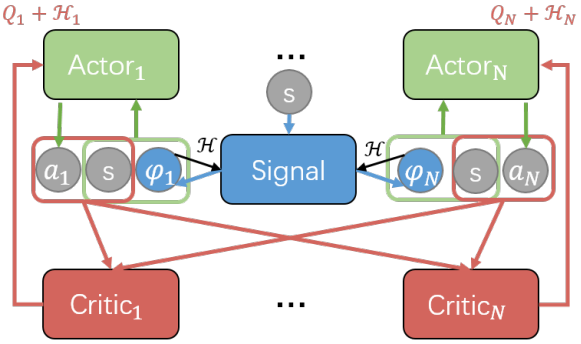


Figure 3: The architecture of our approach. During the training, critics leverage other agents' actions to output the estimate of Q value. For the execution, each agent does not communicate with each other.

Transition probability P defines the probability $p(s_{t+1}|s_t, a_t)$ that the state transits from s_t to s_{t+1} given the action a_t of all the agents in round t. In our setting, the transition probability is equivalent to user behavior probability, which is unknown and associated with a_t. The details are described in the experiment part.

The **objective** of our problem is to maximize the discounted total reward of the platform

$$\sum_{i=1}^{N} \sum_{t=0}^{T} \gamma^t r_t^i$$

rather than the independent reward of each agent r_t^i, where T is the time horizon, and γ^t is t-th power of the discounted parameter γ to decide the weights of future rewards.

4 MULTI-AGENT RL WITH A SOFT SIGNAL NETWORK

In this section, we propose a novel multi-agent reinforcement learning algorithm to address the multi-module recommendation problem. The main idea is to use a signal network to coordinate all the agents to maximize the global reward. Signals can be considered as the information of a general cooperative structure for all the agents. Then, agents act based on signals to cooperate.

Fig. 3 illustrates the structure of our algorithm, which is based on MADDPG [16]. Three components are involved in our structure. A shared signal network takes the state s as input and sends signals ϕ to all the agents to maximize the overall performance. An actor maintained by each agent maps state and signal to action. The i-th actor-network only depends on the state and the signal for the i-th agent, without the knowledge of other agents. To estimate the expected future cumulative reward $Q^i(s, a)$ for given actions and states, each agent has a critic. In the centralized training, critics can evaluate the value of actions with information of all agents. We describe the details of our model and training method in the following respectively.

4.1 Actor-Critic with a Signal Network

Embedding of state We leverage the embedding layer and attention mechanism to extract useful information. The structure is shown in Fig. 4(a). As mentioned in Section 3, the state is the information of users that can be divided into two types, static and sequential features. For the static features like gender, each feature is processed by an independent embedding layer. While for the sequential features, $s_{sequential}$ includes different types' features of K historical clicked items of a user such as item IDs and categories. Features belonging to one type share an embedding layer. For example, the item IDs of the 1st and the K-th items use the same layer. After embedding, sequential features are transformed to a set of vectors $h = [h_1, h_2, \ldots, h_K]$, where h_k is a vector containing the k-th item's features. We build an attention network to estimate the importance w_k of h_k. The attention network takes the embedding of static information e_{static} and h as input. The outputs are mapped by the softmax function to obtain the regularized weights $w = [w_1, w_2, \ldots, w_K]$, where $0 \le w_k \le 1$ and $\sum_k w_k = 1$. The embedding of sequential features $e_{sequential}$ is generated by the weighted sum $\sum_k w_k h_k$. And it is concatenated with e_{static} to get the embedding of the state e_s. This embedding structure is included in actors, critics and the signal network to process the state. The parameters of embedding structures are not shared among different agents and components.

Signal The signal network Φ is shared by all agents during execution to maximize the overall reward. It maps state to a set of vectors $[\phi^1, \phi^2, \ldots, \phi^N]$, where ϕ^i is the signal vector for the i-th agent. The state is processed by the embedding layer mentioned above and fully-connected layers output the signals depending on the embedding of state. Differing from the communication mechanism that needs information sent by all agents, the signal network only depends on states. We adopt stochastic signal policies in which ϕ^i is sampled from a Gaussian distribution $\mathcal{N}(\mu_{\phi^i}, diag(\sigma_{\phi^i}))$ where $[\mu_{\phi^i}, \sigma_{\phi^i}]$ is the output of the signal network

$$[\mu_{\phi^1}, \sigma_{\phi^1}, \mu_{\phi^2}, \sigma_{\phi^2}, \ldots, \mu_{\phi^N}, \sigma_{\phi^N}] = \Phi(s).$$

Algorithm 1: Multi-Agent Soft Signal-Actor (MASSA)

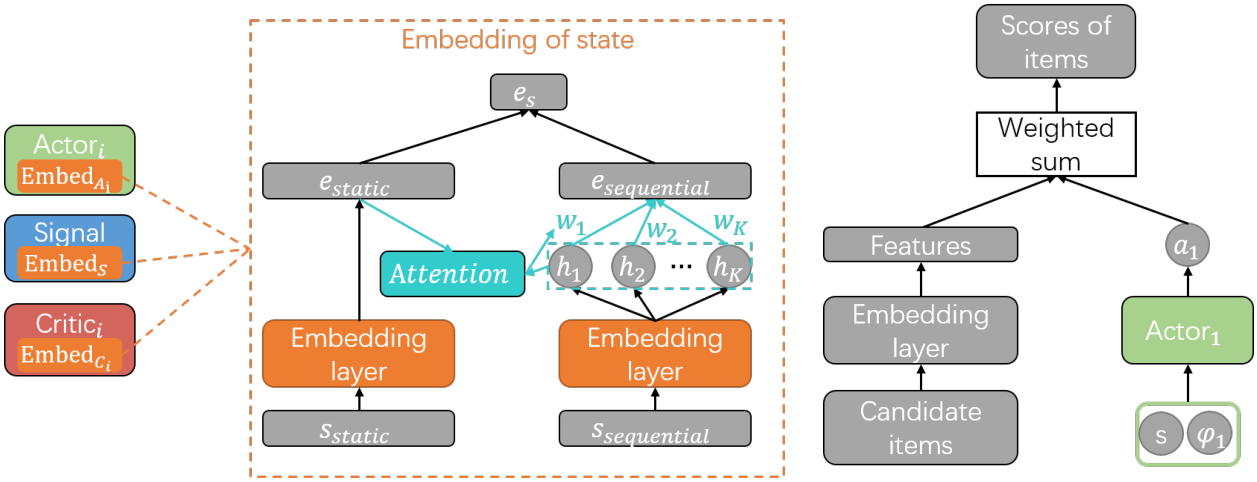
Initialize parameter vectors $(\theta, \eta, \tau, \xi, \delta)$, $\hat{\eta} = \eta$;
Initialize replay buffer D ;
for $t = 0, 1, \ldots$ **do**
  Observe state s_t;
  For each agent i, generate signal $\phi^i = \Phi^i(s_t)$ and select action $a_t^i = \pi^i(s_t, \phi_t^i)$;
  Execute action $a_t = [a_t^1, \ldots, a_t^N]$ and observe reward r_t and new state s_{t+1};
  Store (s_t, a_t, r_t, s_{t+1}) in the replay buffer D;
  Sample a batch of samples from D;
  for *each agent* i **do**
    Calculate $\nabla_\eta J_V^i(\eta)$ and update η;
    Calculate $\nabla_{\theta_j} J_Q^i(\theta_j)$ and update θ_j for $j \in \{1, 2\}$;
    Calculate $\nabla_\tau J_\pi^i(\tau)$ and update τ;
  Calculate $\nabla_\xi J_\Phi^i(\xi)$ and update ξ;
  Update the parameter of the target state value network $\hat{\eta}_{t+1} = (1 - \delta)\hat{\eta}_t + \delta\eta_t$;

Actor The structure of actors are illustrated in Fig. 4(b). Each actor π^i outputs an action given state s and signal ϕ^i. We concatenate the embedding of state and the signal as the input of a three-layer fully-connected network to generate action. We define that the action of each module is a vector whose dimension is the same as the dimension of candidate items' features. Following soft actor-critic [6], we adopt stochastic policy $[\mu_{a^i}, \sigma_{a^i}] = \pi^i(s, \phi^i)$ and the action a^i is sampled from $\mathcal{N}(\mu_{a^i}, diag(\sigma_{a^i}))$. To rank items, we leverage a linear model, in which the weighted sums of items' features and the action are treated as the scores of items. Candidate items are ranked and recommended according to their scores.

Critic Each agent maintains a critic network $Q^i(s, a)$ to estimate the expected cumulative reward of a state-action pair (s, a). The embedding of the state is concatenated with actions of all the agents as the input of a fully-connected network whose output is $Q^i(s, a)$. In our problem, users' historical activities are collected and stored after users leave the multi-module scenario. During the training period, agents can access the actions of other agents to reduce the uncertainty of the environment. Since we use the soft actor-critic structure [6], the double-Q technique is adopted and a state value network $V(s)$ is maintained to stabilize training. The double-Q technique reduces the variance and over-fitting of the Q value by maintaining two Q networks and choosing the minimal Q value as the estimate of the (s, a) pair in each time step. The value network $V(s)$ is used to approximate $\mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[\min Q_j(s, a)]$ for $j \in \{1, 2\}$ and update Q networks.

4.2 Policy Update with the Soft Signal Network

As discussed above, we use neural networks to approximate Q value, V value, represent policies and generate signals. For the i-th agent, we consider a parameterized state value function $V_\eta^i(s_t)$, two Q-functions $Q_{\theta_j}^i(s_t, a_t)$, $j \in \{1, 2\}$, a stochastic policy $\pi_\tau^i(s_t, \phi_t^i)$ and a



(a) The architecture of state embedding. The features of users are divided into two types: static feature, and sequential feature. The attention mechanism is used to decide the weights of items recorded in the sequential feature. The embedding of state is not shared among different networks (including actors, critics and the signal network).

(b) The architecture of actor and ranking. The action is a vector whose dimension is the same with the item's feature. The weighted sum of these two vectors is considered as the score of an item.

Figure 4: State embedding and the structure of ranking.

shared signal network $\Phi_\xi(s_t)$. The parameters of these networks are η, θ, τ, and ξ. The update rules will be introduced in this section.

We adopt Soft Actor-Critic (SAC) [6] for each agent. Differing from standard RL that maximizes the expected sum of rewards

$$\mathbb{E}_{s\sim\rho^\pi, a\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r_t\right],$$

the objective of SAC augments the objective with the expected entropy of the policy $\mathbb{E}_{s\sim\rho^\pi,a\sim\pi}\left[\sum_{t=0}^{T}\gamma^t(r_t+\mathcal{H}(\pi^i(\cdot|s_t,\phi_t^i)))\right]$, where ρ^π is the state distribution induced by π, γ^t is the t-th power of γ and $\mathcal{H}(\pi^i(\cdot|s_t,\phi_t^i)=\mathbb{E}_{a_t^i\sim\pi^i}\left[\pi_\tau^i(a_t^i|s_t,\phi_t^i)\right]$. The entropy term aims at encouraging exploration, while giving up on clearly unpromising avenues. We have

$$Q^i(s_t,a_t)=\mathbb{E}_{s\sim\rho^\pi,a^i\sim\pi^i}\left[r(s_t,a_t)+\gamma V^i(s_{t+1})\right], \tag{1}$$

where

$$V^i(s_t)=\mathbb{E}_{a^i\sim\pi^i}\left[Q^i(s_t,a_t)-\log\pi^i(a_t^i|s_t,\phi_t^i)\right].$$

Then, we update the parameters of Q and V according to [6].

Critic. The centralized critic is optimized according to the Bellmen function of soft actor-critic. For the value function $V_\eta^i(s_t)$, we have

$$J_V^i(\eta)=\mathbb{E}_{s_t\sim D}\left[\frac{1}{2}\left(V_\eta^i(s_t)-\mathbb{E}_{a_t\sim\pi_\tau}\left[Q_\theta^i(s_t,a_t)-\log\pi_\tau^i(a_t^i|s_t,\phi_t^i)\right]\right)^2\right], \tag{2}$$

where D is the distribution of samples, or a replay buffer. The gradient of i-th V value network can be estimated by an unbiased estimator:

$$\hat\nabla_\eta J_V^i(\eta)=\nabla_\eta V_\eta^i(s_t)\left(V_\eta^i(s_t)-Q_\theta^i(s_t,a_t)+\log\pi_\tau^i(a_t^i|s_t,\phi_t^i)\right), \tag{3}$$

where actions and signals are sampled from current networks and $Q_\theta^i=\min_{j\in\{1,2\}}Q_{\theta_j}^i(s_t,a_t)$. The Q value network is trained to minimize the Bellman residual

$$J_Q^i(\theta_j)=\mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_{\theta_j}^i(s_t,a_t)-\hat Q^i(s_t,a_t)\right)^2\right], \tag{4}$$

with

$$\hat Q^i(s_t,a_t)=r_t^i(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim P}\left[V_{\hat\eta}^i(s_{t+1})\right], \tag{5}$$

where $V_{\hat\eta}^i(s_{t+1})$ is a target network of V, where $\hat\eta$ is an exponentially average of η. More specifically, the update rule for $\hat\eta$ is $\hat\eta_{t+1}=(1-\delta)\hat\eta_t+\delta\eta_t$. We approximate the gradient for θ_j with

$$\hat\nabla_{\theta_j}J_Q^i(\theta_j)=\nabla_{\theta_j}Q_{\theta_j}^i(s_t,a_t)\left(Q_{\theta_j}^i(s_t,a_t)-\hat Q^i(s_t,a_t)\right). \tag{6}$$

Actor. For each actor, the objective is to maximize the Q value with the entropy term, since Q value introduced in Eq. (1) does not include $\mathcal{H}(\pi^i(\cdot|s_t,\phi_t^i))$:

$$J_\pi^i(\tau)=-\mathbb{E}_{s_t\sim D,a^i\sim\pi^i}\left[Q_\theta^i(s_t,a_t^{-i},a_t^i)-\log\pi_\tau^i(a_t^i|s_t,\phi_t^i)\right], \tag{7}$$

where a_t^{-i} is a vector including actions of all the agents except the i-th agent and a_t^i is generated by the current policy π_τ^i. We use reparameterization trick [11]

$$a_t^i=f_\tau(\epsilon_t;s_t,\phi_t^i)=f_\tau^\mu(s_t,\phi_t^i)+\epsilon f_\tau^\sigma(s_t,\phi_t^i),$$

where $\epsilon\sim\mathcal{N}(0,I)$ and I is identity matrix. f_τ is a neural network whose output is $[f_\tau^\mu,f_\tau^\sigma]$ and τ is the parameter of f_τ. The stochastic gradient is

$$\begin{aligned}\hat\nabla_\tau J_\pi^i(\tau)=&\nabla_\tau\log\pi_\tau^i(a_t^i|s_t,\phi_t^i)+\left(-\nabla_{a_t^i}Q_\theta^i(s_t,a_t^{-i},a_t^i)+\right.\\&\left.\nabla_{a_t^i}\log\pi_\tau^i(a_t^i|s_t,\phi_t^i)\right)\nabla_\tau f_\tau(\epsilon_t;s_t,\phi_t^i).\end{aligned} \tag{8}$$

These updates are extended from soft actor-critic algorithm [6].

The entropy-regularized signal network. Now we introduce the update of the signal network. Since the signal network aims at

Algorithm 2: Offline testing procedure.

1 Load parameters of actors, signal network and item embedding layer;
2 **for** $t = 0, 1, \ldots$ **do**
3 Read a record from testing dataset;
4 Observe state s_t;
5 Observe candidate set of items for two modules $L^i,\ i \in 1, 2$;
6 For each agent, rank these items and output a list;
7 Observe rewards r_t of recommended lists from the record;
8 Generate next state s_{t+1} (for training only);

maximizing the overall reward, the objective function is

$$J_\phi(\xi) = \frac{1}{N}\sum_i -\mathbb{E}_{s_t, a_t^{-i}\sim D}\left[Q_\theta^i(s_t, a_t^{-i}, a_t^i)\right]. \tag{9}$$

Inspired by the soft actor-critic, we augment an expected entropy of the signal network (soft signal network) and obtain a new objective. Intuitively, this term can encourage signal network to coordinate agents' exploration and find the optimal solution to maximize the global reward. Since the signal network outputs a signal ϕ^i for each agent i, we use the notation Φ^i to represent the part of the signal network for the i-th agent. Since

$$\mathcal{H}(\Phi^i(\cdot|s_t)) = \mathbb{E}_{\phi^i\sim\Phi^i}\log\Phi^i(\phi^i|s_t),$$

we have

$$\begin{aligned} J_\phi(\xi) = &\frac{1}{N}\sum_i \Big[\mathbb{E}_{s_t, a_t^{-i}\sim D, \phi^i\sim\Phi^i}\left[-Q_\theta^i(s_t, a_t^{-i}, \pi^i(s_t, \phi_t^i))+\right. \\ &\left.\alpha\log\Phi^i(\phi_t^i|s_t)\right]\Big]. \end{aligned} \tag{10}$$

According to [10], we derive the stochastic gradient using reparameterization trick again $\phi_t^i = g_\xi^i(\epsilon; s_t) = g_\xi^\mu(s_t) + \epsilon g_\xi^\sigma(s_t)$, where g is a neural network and $\epsilon \sim \mathcal{N}(0, I)$:

$$\begin{aligned} \hat{\nabla}_\xi J_\Phi(\xi) = \frac{1}{N}\sum_i \Big[&\alpha\nabla_\xi\log\Phi_\xi^i(\phi_t^i|s_t) + \Big(\alpha\nabla_{\phi_t^i}\log\Phi^i(\phi_t^i|s_t)- \\ &\nabla_{a_t^i}Q_\theta^i(s_t, a_t^{-i}, a_t^i)\nabla_{\phi_t^i}\pi_\tau^i(a_t^i|s_t, \phi_t^i)\Big)\nabla_\xi g_\xi^i(\epsilon_t; s_t)\Big]. \end{aligned} \tag{11}$$

The whole algorithm is shown in Algorithm 1. The algorithm can be divided into two stages, execution and training. In the execution part (Lines 4-7), policies of different agents are executed in the environment to collect data that is stored in the replay buffer. In the training part, all the parameters are updated according to their gradients derived in this section. In the end, the parameter of the target network is updated.

5 EXPERIMENT

We conduct extensive experiments to evaluate the performance of our algorithm based on Taobao. We first describe the details of the experimental setting. Then, some baselines are introduced. Finally, the performance of baselines and our algorithm are illustrated.

5.1 Dataset

Our dataset is collected from Taobao. The recommendation scenario contains two modules. For the training data, 14-day data is collected in March 2020 and about 1.5 million records (583076 items) are included in the dataset. Another 3-day data (about 200 thousand records) is used as the test dataset in offline testing. Each record includes a user's information, 10 recommended items for each module, the user's clicks, and the user's information after clicking. As we mentioned in the formulation section, users' information contains sequential and static features. Sequential features contain 50 items that the user clicked. The item ID, seller ID, and category ID of these historical clicked items are stored. If the number of historical clicked items of a user is less than 50, these features are set to 0 by default. For recommended items, features include price, sale, category and other information of items. After embedding, each item is represented by a 118-dimensional vector.

5.2 Experiment Setting

In the experiment, we use both offline and online (simulator) testing to illustrate the performance of our algorithm. In the offline training and evaluation, algorithms re-rank browsed items in each record and the clicked items should be ranked at the top of the list. The candidate set is limited to the recommended items stored in each record rather than all items collected from the dataset since we do not know the real reward of items that the user does not browse. The rewards of the recommended lists of our algorithm are directly obtained from historical data and used to evaluate the performance of algorithms. During training, since we need s_{t+1} to update parameters, the users' information s_t is updated by following rules. The static part of s is fixed and not changed no matter what the user clicks. If an item is clicked, the item is added into the sequential feature and the 50-th historical clicked item is removed from the sequential feature. If M item is clicked, we do M updates of the sequential feature. We assume that items in the first module are clicked firstly and the items with high ranks are clicked before those with low rank within a module. Then, the new s_{t+1} is generated and stored to train our algorithm. The offline testing algorithm in detail is presented in Algorithm 2.

For the online training and testing, due to the huge cost and risk caused by deploying different algorithms to the real-world scenario, we train a simulator to implement the online testing following [27]. The structure of the simulator is shown in Fig. 5. In order to consider the information on the whole page, the input of the simulator is all the recommended items of two modules (6 items). We obtain embedding of items by a shared embedding layer. Meanwhile, the user's information is processed by the embedding structure shown in Fig. 4(a). The features of items and a user are concatenated as the input of a four-layer fully-connected network. Then, the CTRs (Click Through Rate) of these items are predicted. The bias of position is considered by this design since the sequence of items in the input actually indicates the information of positions. We test the trained simulator in the test dataset (not used to train the simulator). The overall accuracy is over 90%, which suggests that the simulator can accurately simulate the real online environment.

For training and testing our algorithm, we collect 2000 items with the largest CTR for each module to expand the candidate set. In

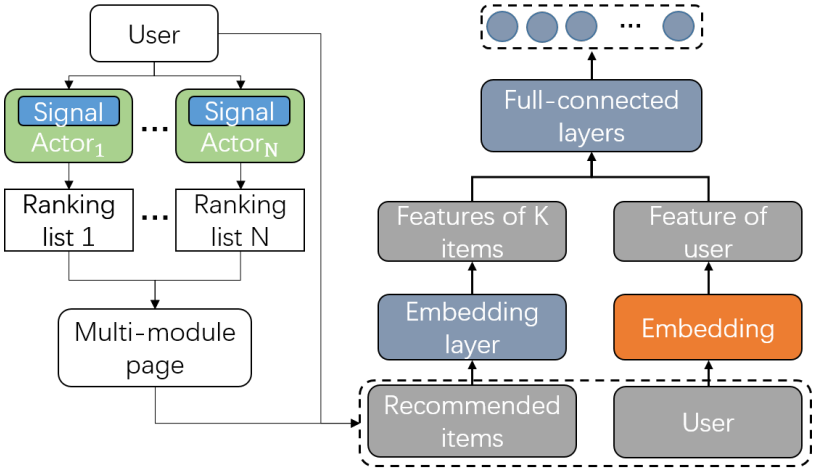


Figure 5: The structure of our simulator. The inputs are all the recommended items on one web page and the information of a user. The output is a vector including the probabilities that these items are clicked.

Table 1: Results of offline testing.

Method \ Metric	Precision			nDCG		
	Module 1	Module 2	Overall	Module 1	Module 2	Overall
L2R	0.193	0.047	0.24	0.196	0.042	0.238
DDPG	0.211	0.046	0.257	0.214	0.042	0.256
MADDPG	0.227	0.047	0.274	0.231	0.044	0.275
COMA	0.165	0.039	0.204	0.175	0.037	0.212
QMIX	0.396	0.056	0.452	0.368	0.055	0.423
COM	0.216	0.042	0.258	0.217	0.041	0.258
MASAC	0.367	0.055	0.422	0.337	0.051	0.389
COMA+SAC	0.206	0.048	0.254	0.205	0.045	0.25
QMIX+SAC	0.305	0.048	0.353	0.294	0.042	0.336
COM+SAC	0.292	0.047	0.341	0.29	0.045	0.335
MAAC	0.301	0.046	0.347	0.289	0.043	0.332
MASSA w/o att (ours)	0.397	0.050	0.447	0.433	0.052	0.485
MASSA w/o en (ours)	0.44	0.055	0.495	0.398	0.05	0.448
MASSA (ours)	**0.555**	**0.06**	**0.615**	**0.459**	**0.057**	**0.516**

our training and testing dataset, about 90% of clicks are contributed by these items. In each round, actors select a list of items and the simulator outputs rewards for these items. The training and testing procedure is similar to Algorithm 2 except the Line 7, where the rewards come from the simulator rather than historical data.

To evaluate the performance of various algorithms, we use clicks as rewards and introduce two metrics Precision [17] and nDCG [24]. The formulations are shown as follows.

- Precision:

$$Precision = \frac{\#\text{clicks in top-K items}}{K}.$$

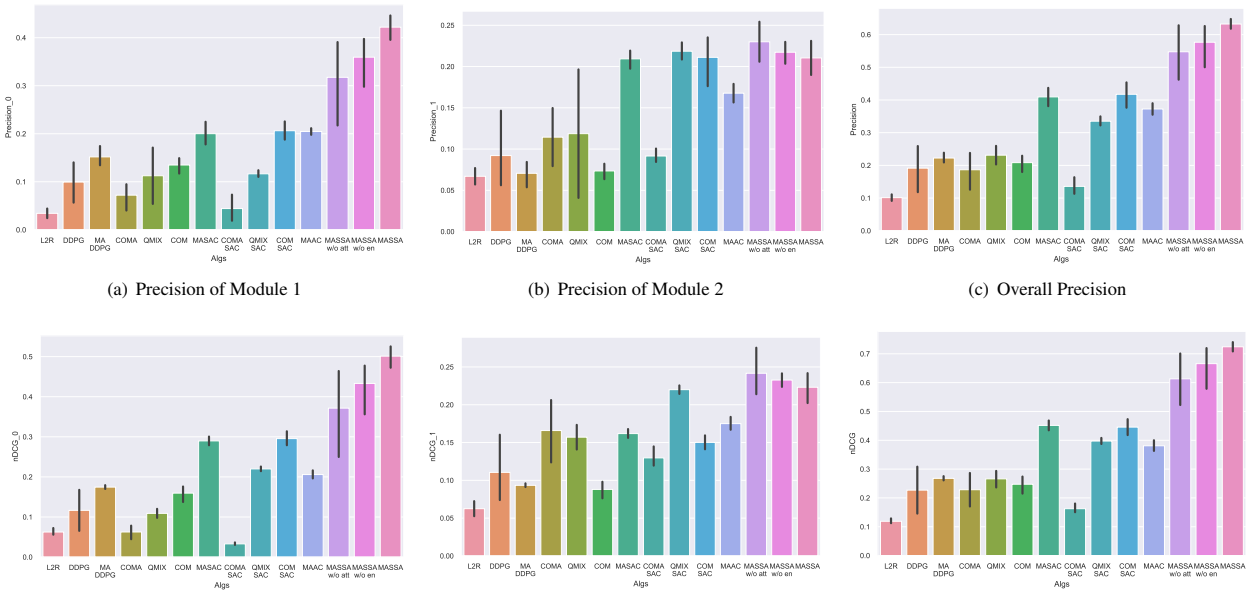
- nDCG:

$$nDCG = \sum_{k=1}^{K} \frac{r_k}{\log(1+k)},$$

where $r_k = 1$ if the k-th item is clicked, otherwise, $r_k = 0$.

For each module, the performance of a ranking policy is evaluated by these two metrics. The overall performance is the sum of each module's performance.

For components of our algorithm, we leverage a 4-layer neural network with the additional embedding structure introduced in Fig. 4(a). The activation function is *relu* for all fully-connected layers except output layers. The size of the replay buffer is $1e6$. The dimension of the items' embedding is 118. The length of each signal vector is 64. The discount factor is $\gamma = 0.99$. The learning rate for actor, critic, and signal networks is 0.01 and the weight for updating the target network is $\delta = 0.01$. The weight of entropy terms is $\alpha = 0.01$. We select these parameters via cross-validation and do parameter-tuning for baselines for a fair comparison.



(a) Precision of Module 1 (b) Precision of Module 2 (c) Overall Precision

(d) nDCG of Module 1 (e) nDCG of Module 2 (f) Overall nDCG

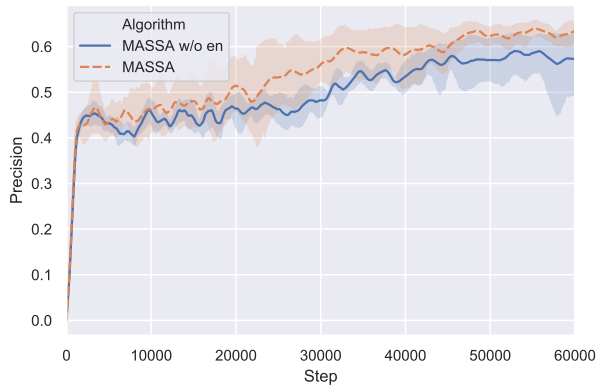
Figure 6: The results of the online experiment.

5.3 Baselines

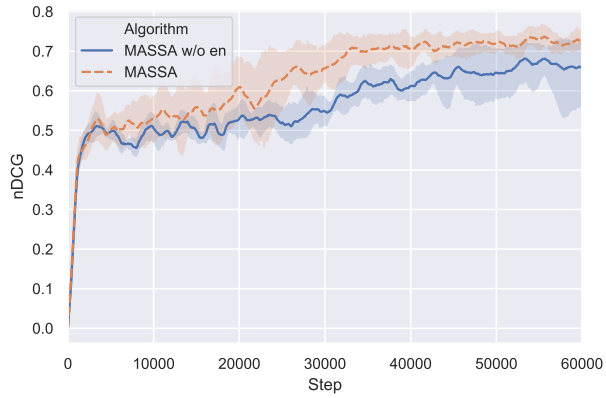
Our algorithm is compared with the following baselines:

- **L2R** [14]: This algorithm trains a point-wise learning-to-rank network by supervised learning. The network is the same as the simulator except for the input and the output. The input changes to users' information and one item. The network predicts the CTR of this item. We deploy an L2R algorithm for each module, which is trained to reduce the sigmoid cross-entropy loss for each module.
- **DDPG** [12]: Deep Deterministic Policy Gradient method is a single-agent RL algorithm that consists of an actor and a critic. The structure of actors is the same as MADDPG and L2R.
- **MADDPG** [16]: Multi-Agent Deep Deterministic Policy Gradient method is the multi-agent version of DDPG. Each agent maintains an actor and a critic. During training, critics can access other agents' actions and observations. While in the execution, actors select actions only depending on their own observation.
- **COMA** [4]: Counterfactual multi-agent policy gradients method is a cooperative multi-agent algorithm that leverages counterfactual rewards to train agents. The main idea is to change the action of an agent to a baseline action and use the gap of Q values of these two actions as the reward. Differing from MADDPG, all the agents share a critic to estimate the global reward.
- **QMIX** [20]: QMIX assumes that the global maximum reward is a weighted sum of local maximum rewards of agents and proposes a mixing network to explicitly decompose the global reward. The decomposed local rewards are treated as the contribution of each agent and used to train actors.
- **COM**: COM is a simple extension of the methods [3] by letting actors choose actions simultaneously. Actors send messages to others during execution. Although this algorithm violates the restriction that different modules cannot communicate, the comparison aims to illustrate the performance in environments that allow communication.
- **MASAC**: This algorithm is an extension of MADDPG by applying soft actor-critic [6], where an entropy term is augmented in the reward to encourage exploration. Different from our method, this algorithm does not have a signal network.
- **MAAC** [9]: Multi Actor-Attention-Critic algorithm maintains the structure of MASAC. The attention mechanism is adopted to handle messages sent by critics and extract useful information to each critic.
- **MASSA w/o en**: This method is proposed for the ablation study, in which the entropy terms of signals are removed from the loss function of the signal network ($\alpha = 0$). By comparing this method with ours, the importance of the entropy-regularized version of the loss function is indicated.
- **MASSA w/o att**: In this method, the attention mechanism is replaced by simple concatenation $e_s = [e_{static}, h]$ for ablation study.

Additionally, since our algorithm is based on MASAC, we combine COMA, QMIX, and COM with MASAC to obtain the other three baselines: COMA+SAC, QMIX+SAC, and COM+SAC. Notice that the method in [28] is a model-based version of COM and other baselines (including ours) are model-free methods. Thus, we only compare to COM considering fairness.



(a) Precision



(b) nDCG

Figure 7: The curves of precision and nDCG during online training. The solid curves correspond to the mean and the shaded region to the minimum and maximum values over the 10 runs. The difference between these two algorithms is the entropy term of the loss function for the signal network.

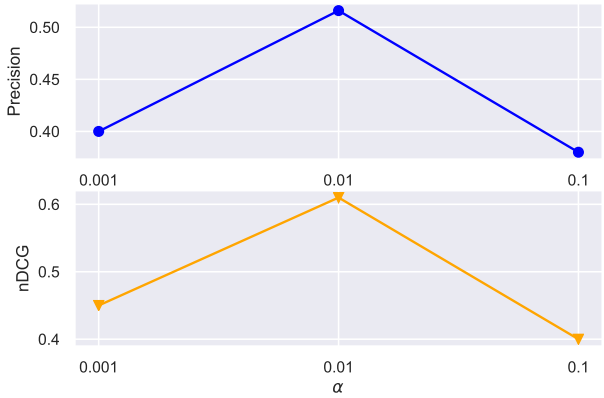
5.4 Result

In this subsection, we illustrate performance of different methods to indicate the improvement caused by the signal network and the entropy scheme.

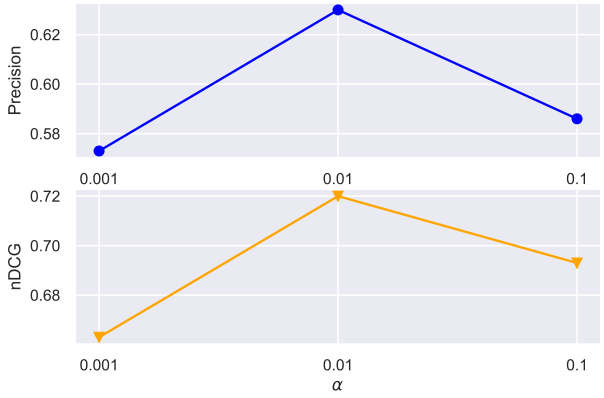
5.4.1 Offline Testing. The results of our offline evaluation are shown in Table 1. All the methods are trained by 14-day training data and tested by the 3-day testing data. There are a few interesting conclusions drawn from the results.

Firstly, the signal network and additional entropy terms can improve performance significantly. Since the only distinction between *MASAC* and *MASSA w/o entropy* is the signal network, the gap of the performance shows the effectiveness of the signal network. Besides, by adding the entropy term to encourage exploration, *MASSA* method outperforms all the other methods. Comparing to *MASSA*, the *MASSA w/o entropy* method is prone to converge to a sub-optimal policy in our scenarios. The entropy-regularized algorithm can explore in view of global performance.

Secondly, the metrics of module 1 are better than that of module 2, which is caused by different properties of these two modules.



(a) Offline



(b) Online

Figure 8: The performance with the change of α

As shown in Fig. 2, module 1 is at the top of the web page. Thus, users are more likely to be attracted by module 1 and ignore another module, especially when the items recommended by module 1 are good. In our dataset, the ratio of the number of clicks in these two modules is about 6:1.

Finally, *DDPG* performs worse comparing with *MADDPG* whose actors have a similar structure with *DDPG*. The main reason is that the ranking policies of the two modules are trained individually without any cooperation.

5.4.2 Online Testing. For the online experiment, Fig. 6 exhibits the performance of various algorithms. The performance is the mean of 10 runs. Our algorithms outperform others again in the online experiment.

Firstly, the performance of the methods based on *MASAC* is better than that based on *MADDPG* except for *COMA*. The reason is that the online environment is more complex than the offline setting in terms of the number of candidate items and the source of clicks. The number of candidate items increases from 10 to 2000 for each set and the clicks are from an online simulator. Exploration is more important to obtain a better policy in a complex environment. Thus, SAC-based approaches perform better.

Secondly, although *MASSA w/o entropy* is better than *MASSA* for the module 2, the overall performance of *MASSA w/o entropy* is

worse. It illustrates that in order to find a globally optimal solution, *MASSA* makes a small sacrifice of module 2 and obtains a huge improvement for the overall performance.

The effect of entropy The importance of the entropy term is indicated in Fig. 7. We can observe that two algorithms perform similar in the first 20 thousand steps and the overall performance seems to be constant if we ignore the perturbation, which means that the ranking policy falls into a sub-optimal solution. Due to the entropy term, *MASSA* constantly explores and escapes from the sub-optimal solution at around 30 thousand steps. Finally, a globally optimal solution is found. However, *MASSA w/o entropy* algorithm only finds a better sub-optimal solution slowly.

Another interesting fact is the change in the shaded region. For *MASSA*, the region is huge before 45 thousand steps and becomes smaller in the last 10 thousand steps. However, the region of *MASSA w/o entropy* becomes larger at the end of the training. It indicates that *MASSA* explores more at the beginning and converges to the optimal solution. However, due to the lack of exploration, *MASSA w/o entropy* falls into different sub-optimal solutions in the end.

5.4.3 The influence of α. Fig. 8 shows the performance with the change of α which is the weight of the entropy term for the loss function of the signal network. Our algorithm performs the best when $\alpha = 0.01$. Thus, we use this value in both online and offline experiments.

6 CONCLUSION

In this paper, we propose a novel multi-agent cooperative learning algorithm for the multi-module recommendation problem, in which a page contains multiple modules that recommend items processing different specific properties. To prompt cooperation and maximize the overall reward, we firstly design a signal network that sends additional signals to all the modules. Secondly, an entropy-regularized version of the signal network is proposed to coordinate agents' exploration. Finally, we conduct both offline and online experiments to verify that our proposed algorithm outperforms other state-of-the-art learning algorithms.

ACKNOWLEDGMENTS

This work was supported by Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Joint Research Institute (JRI), Nanyang Technological University, Singapore.

REFERENCES

[1] Robert J Aumann. 1974. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics* 1, 1 (1974), 67–96.
[2] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. 2018. Reinforcement learning based recommender system using biclustering technique. *arXiv preprint arXiv:1801.05532* (2018).
[3] 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 World Wide Web Conference*. 1939–1948.
[4] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual Multi-Agent Policy Gradients. *arXiv preprint cs.AI/1705.08926* (2017).
[5] Amy Greenwald, Keith Hall, and Roberto Serrano. 2003. Correlated Q-learning. In *Proceedings of the International Conference on Machine Learning*. 242.
[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290* (2018).
[7] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application. *arXiv preprint arXiv:1803.00710* (2018).
[8] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Navrekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. 2019. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767* (2019).
[9] Shariq Iqbal and Fei Sha. 2018. Actor-attention-critic for multi-agent reinforcement learning. *arXiv preprint arXiv:1810.02912* (2018).
[10] Martin Jankowiak and Fritz Obermeyer. 2018. Pathwise derivatives beyond the reparameterization trick. *arXiv preprint arXiv:1806.01851* (2018).
[11] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[12] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
[13] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning*. Elsevier, 157–163.
[14] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and trends in information retrieval* 3, 3 (2009), 225–331.
[15] Yong Liu, Yinan Zhang, Qiong Wu, Chunyan Miao, Lizhen Cui, Binqiang Zhao, Yin Zhao, and Lu Guan. 2019. Diversity-promoting deep reinforcement learning for interactive recommendation. *arXiv preprint arXiv:1903.07826* (2019).
[16] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*. 6379–6390.
[17] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. Introduction to information retrieval. *Natural Language Engineering* 16, 1 (2010), 100–103.
[18] Harrie Oosterhuis and Maarten de Rijke. 2018. Ranking for relevance and display preferences in complex presentation layouts. In *International ACM SIGIR Conference on Research & Development in Information Retrieval*. 845–854.
[19] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. 2019. Value-aware Recommendation based on Reinforced Profit Maximization in E-commerce Systems. *arXiv preprint arXiv:1902.00851* (2019).
[20] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485* (2018).
[21] Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold, Yori Zwols, Daniel Visentin, and Ben Coppin. 2015. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. *arXiv preprint arXiv:1512.01124* (2015).
[22] Ryuichi Takanobu, Tao Zhuang, Minlie Huang, Jun Feng, Haihong Tang, and Bo Zheng. 2019. Aggregating E-commerce Search Results from Heterogeneous Sources via Hierarchical Reinforcement Learning. In *Proceedings of the World Wide Web Conference*. 1771–1781.
[23] Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. 2020. Learning Efficient Multi-agent Communication: An Information Bottleneck Approach. *Proceedings of the International Conference on Machine Learning* (2020).
[24] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *Conference on Learning Theory*. 25–54.
[25] Mengchen Zhao, Zhao Li, Bo An, Haifeng Lu, Yifan Yang, and Chen Chu. 2018. Impression Allocation for Combating Fraud in E-commerce Via Deep Reinforcement Learning with Action Norm Penalty. *Proceedings of the International Joint Conferences on Artificial Intelligence Organization*, 3940–3946.
[26] 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), 4.
[27] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. *arXiv preprint arXiv:1805.02343* (2018).
[28] Xiangyu Zhao, Long Xia, Yihong Zhao, Dawei Yin, and Jiliang Tang. 2019. Model-Based Reinforcement Learning for Whole-Chain Recommendations. *arXiv preprint arXiv:1902.03987* (2019).
[29] 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 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 167–176.
[30] 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. *arXiv preprint arXiv:1902.05570* (2019).