Recommendation via Reinforcement Learning Methods

Xu He

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy (Ph.D.)

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

15/03/2021

.....
Date

Xu He

.....
Xu He

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

15/03/2021

.....
Date

.....
Bo An

Authorship Attribution Statement

This thesis contains material from four papers published in the following peer-reviewed conferences in which I am the first and corresponding author.

Chapter 3 is published as **Xu He**, Haipeng Chen, and Bo An, "Learning Behaviors with Uncertain Human Feedback". *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (**UAI'20**), pp. 131-140, 2020. The contributions of the authors are as follows:

- Prof. Bo An and I proposed the problem setting.
- I proposed improved methods, conducted the experiments, and wrote the main manuscript. Dr. Haipeng Chen wrote a part of the manuscript.
- Dr. Haipeng Chen and Prof. Bo An provided insightful comments and carefully revised the manuscript.

Chapter 4 is published as **Xu He**, Bo An, Yanghua Li, Haikai Chen, Qingyu Guo, Xin Li and Zhirong Wang, "Contextual user browsing bandits for large-scale online mobile recommendation". *Proceedings of the 2020 ACM Recommender Systems conference* (**RecSys'20**), pp.63-72, 2020. The contributions of the authors are as follows:

- I proposed the idea and the solution.
- I conducted the experiments. Haikai Chen collected dataset and finished data processing.
- I wrote the manuscript. Prof. Bo An discussed with me about how to properly present the key idea of the manuscript.
- I derived the associated mathematical equations for the analytical frameworks proposed in the manuscripts. Dr. Qingyu Guo discussed with me about the analytical frameworks.
- Yanghua Li, Haikai Chen, Xin Li, Zhirong Wang, Dr. Qingyu Guo and Prof. Bo An provided insightful comments and revised the manuscript.

Chapter 5 is published as **Xu He**, Bo An, Yanghua Li, Haikai Chen, Rundong Wang, Xinrun Wang, Runsheng Yu, Xin Li and Zhirong Wang, " Learning to Collaborate in Multi-Module Recommendation via Multi-Agent Reinforcement Learning without Communication". *Proceedings of the 2020 ACM Recommender Systems conference* (**RecSys'20**), pp.210-219, 2020. The contributions of the authors are as follows:

- Prof. Bo An and I proposed the idea.
- I conducted the experiments. Haikai Chen collected dataset and finished data processing.
- I wrote the manuscript. Prof. Bo An discussed with me about the framework of the manuscript.
- Yanghua Li, Haikai Chen, Rundong Wang, Dr. Xinrun Wang, Runsheng Yu, Xin Li, Zhirong Wang and Prof. Bo An provided critical comments and revised the manuscript.

15/03/2021 Xu He

....

Date Xu He

Abstract

Recommender system has been a persistent research goal for decades, which aims at recommending suitable items such as movies to users. Supervised learning methods are widely adopted by modelling recommendation problems as prediction tasks. However, with the rise of online e-commerce platforms, various scenarios appear, which allow users to make sequential decisions rather than one-time decisions. Therefore, reinforcement learning methods have attracted increasing attention in recent years to solve these problems.

This doctoral thesis is devoted to investigating some recommendation settings that can be solved by reinforcement learning methods, including *multi-arm bandit* and *multi-agent reinforcement learning*.

For the recommendation domain, most scenarios only involve a single agent that generates recommended items to users aiming at maximizing some metrics like click-through rate (CTR). Since candidate items change all the time in many online recommendation scenarios, one crucial issue is the trade-off between exploration and exploitation. Thus, we consider multi-arm bandit problems, a special topic in online learning and reinforcement learning to balance exploration and exploitation. We propose two methods to alleviate issues in recommendation problems.

Firstly, we consider how users give feedback to items or actions chosen by an agent. Previous works rarely consider the uncertainty when humans provide feedback, especially in cases that the optimal actions are not obvious to the users. For example, when similar items are recommended to a user, the user is likely to provide positive feedback to sub-optimal items, negative feedback to the optimal item and even do not provide feedback in some confusing situations. To involve uncertainties in the learning environment and human feedback, we introduce a feedback model. Moreover, a novel method is proposed to find the optimal policy and proper feedback model simultaneously.

Secondly, for the online recommendation in mobile devices, positions of items have a significant influence on clicks due to the limited screen size of mobile devices: 1) Higher positions lead to more clicks for one commodity. 2) The 'pseudo-exposure' issue: Only a few recommended items are shown at first glance and users need to slide the screen to browse other items. Therefore, some recommended items ranked behind are not viewed by users and it is not proper to treat these items as negative samples. To address these two issues, we model the online recommendation as a contextual combinatorial bandit problem and define the reward of a recommended set. Then, we propose a novel contextual combinatorial bandit method and provide a formal regret analysis. An online experiment is implemented in Taobao, one of the most popular e-commerce platforms in the world. Results on two metrics show that our algorithm outperforms the other contextual bandit algorithms.

For *multi-agent reinforcement learning* setting, we focus on a kind of recommendation scenario in online e-commerce platforms, which involves multiple modules to recommend items with different properties such as huge discounts. A web page often consists of some independent modules. The ranking policies of these modules are decided by different teams and optimized individually without cooperation, which would result in competition between modules. Thus, the global policy of the whole page could be sub-optimal. To address this issue, we propose a novel multi-agent cooperative reinforcement learning approach with the restriction that modules cannot communicate with others. Experiments based on real-world e-commerce data demonstrate that our algorithm obtains superior performance over baselines.

Acknowledgments

First and foremost, I am greatly indebted to my supervisor, Prof. Bo An, who, throughout my four-year journey at Nanyang Technological University, has provided me insightful discussions, a suitable research environment, as well as the freedom that allows me to explore interesting research problems. His great sense of research taste encourages and inspires me to work on important questions. Not only did I learn about how to be professional, but also I learn from his methods and thoughts to tackle research problems. I could not have imagined having a better advisor and mentor for my Ph.D. study.

I am also very fortunate to meet and learn from a group of talented people: Qingyu Guo, Wanyuan Wang, Haipeng Chen, Yanhai Xiong, Mengchen Zhao, Jiuchuan Jiang, Youzhi Zhang, Xinrun Wang, Lei Feng, Aye Phyu, Hongxin Wei, Rundong Wang, Wei Qiu, Yakub Cerny, Runsheng Yu, Yanchen Deng, Shuxin Li, Wanqi Xue, Fei Fei Lin, Zhuyun Yin, Xinyu Cai, Shuo Sun, Pengdeng Li, Shenggong Ji and Shufeng Kong. I would like to all of them for the joy, the support, the excitement that they gave me.

I would like to thank my Thesis Advisory Committee (TAC) members: Prof. Sinno Jialin Pan and Prof. Xiaohui Bei. They helped me a lot and made my research journey easier at NTU.

During my Ph.D., I had an exciting internship at Alibaba Group. I would like to thank Haikai Chen, Yanghua Li and other people I met at Hangzhou, who helped me a lot during the internship.

I would like to thank many of my friends: Zhuoyi Lin, YunFeng Zhai, Yuchao Zhao, Zhiyuan Cui, Haoliang Gou, Shihao Liu, Zheng Xu, Heng Zhao, Changda Xia, Peng Zhao, Chong Wu and so on. The bond built among us is one of the most cherished things in my life.

Most importantly, I would like to thank my families, especially my parents: Ruili Hou and Wei He, for their unconditional love and support to me. Needless to say, this thesis

would not have been possible without your encouragement along the way. This thesis is dedicated to all of you. I love you.

Contents

List of Figures

List of Tables

List of Publications

International Conferences

1. Wei Qiu, Xinrun Wang, Runsheng Yu, **Xu He**, Rundong Wang, Bo An, Svetlana Obraztsova, Zinovi Rabinovich. RMIX: Learning Risk-Sensitive Policies for Cooperative Reinforcement Learning Agents. *arXiv preprint arXiv:2102.08159* (2021) [73]

2. Runsheng Yu, Yu Gong, **Xu He**, Yu Zhu, Qingwen Liu, Wenwu Ou, Bo An. Personalized Adaptive Meta Learning for User Imbalanced Preference Prediction. *Proceedings of the 35th AAAI Conference on Artificial Intelligence* (**AAAI'21**) [99]

3. **Xu He**, Bo An, Yanghua Li, Haikai Chen, Qingyu Guo, Xin Li, Zhirong Wang. Contextual user browsing bandits for large-scale online mobile recommendation. *Proceedings of the 2020 ACM Recommender Systems conference* (**RecSys'20**) [24]

4. **Xu He**, Bo An, Yanghua Li, Haikai Chen, Rundong Wang, Xinrun Wang, Runsheng Yu, Xin Li, Zhirong Wang. Learning to collaborate in multi-module recommendation via multi-agent reinforcement learning without communication. *Proceedings of the 2020 ACM Recommender Systems conference* (**RecSys'20**) [25]

5. **Xu He**, Haipeng Chen, and Bo An. Learning Behaviors with Uncertain Human Feedback. *Proceedings of the 2020 Conference on Uncertainty in Artificial Intelligence* (**UAI'20**) [26]

6. Rundong Wang, **Xu He**, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. Learning Efficient Multi-agent Communication: An Information Bottleneck Approach. *Proceedings of the 37th International Conference on Machine Learning* (**ICML'20**) [92]

7. Feifei Lin, **Xu He**, Bo An. Context-aware multi-agent coordination with loose couplings and repeated interaction. *Proceedings of the 2nd International Conference on Distributed Artificial Intelligence* (**DAI'20**) (Best paper) [53]

Chapter 1

Introduction

1.1 Background

In many e-commerce platforms, users are faced with countless products. To overcome such information overload, the recommneder system [76] provides users with a list of items that are likely to interest them. By treating recommendation problems as prediction tasks, supervised learning methods are widely used in the last decades. However, user behaviours are sequential in many online e-commerce platforms. For instance, to buy a computer, users may browse and compare multiple products within one week. Then, the recommendation agent needs to make sequential decisions rather than one-time decisions about which items should be recommended.

To address the sequential decision problems, reinforcement learning (RL) [86] has attracted attention and been deployed in many e-commerce platforms. According to the number of agents involving in the recommender system, we can divide RL methods into two types: single-agent and multi-agent algorithms. Most of the traditional recommendation problems like search engines need a single agent to display a list of items on users' devices. Thus, these problems can be solved by deploying a single algorithm, namely one agent. However, with the rise of online e-commerce platforms, recommendation scenarios become complicated, in which one page usually contains different modules. Each module has its own function or focuses on recommending the items with a specific property like huge discounts. If we consider each module as an agent, multi-agent methods are proper to be deployed in such scenarios.

In this thesis, we focus on the *multi-arm bandit* setting [37, 38, 46, 49, 74] for single-agent RL and the *multi-agent reinforcement learning* setting [19, 32, 60, 75]. Multi-arm bandit setting is a simplified version of RL, in which an agent repeatedly interacts with users to estimate users' interest for some items. Since users and items vary all the time, one crucial issue is how to balance exploration and exploitation. Without exploration, the recommendation agent cannot adapt to the change of items and users. However, if the agent is prone to exploration, the probability that users like the recommended items would be small, which is contrary to the aim, providing users' preferred items. To address this issue, bandit algorithms are proposed to explore efficiently and learn rapidly, which are proper to apply into recommendation scenarios that items and users change rapidly. For the multi-agent reinforcement learning setting, we focus on the aforementioned multi-module problem that multiple modules appear on one page. Each module needs to independently decide what to recommend due to the restriction of the online recommendation system. One vital issue is how to prompt cooperation and avoid competition between modules under this constraint.

1.2 Motivations

Although many recommendation problems are appropriate to be modeled as RL problems, most existing RL methods are designed to play computer games or control robotics. Thus, we need to modify existing methods to address domain-specific issues of recommendation tasks. Although some works make effort to extend existing RL methods to the recommendation domain, many issues still need to be addressed.

For the *multi-arm bandit* setting, we first consider how to understand the feedback provided by users. Most existing papers assume that human feedback is accurate. However, in environments with uncertainties, users may not know exactly the optimal action themselves, and therefore could only provide feedback with uncertainty. For example, users could be confused by some items with similar pictures or names shown on web pages and click the item that they do not prefer. Users are likely to provide positive feedback to sub-optimal actions, negative feedback to the optimal actions and even do not provide feedback in some confusing situations. Existing works, which utilize the Expectation

Maximization (EM) algorithm and treat the feedback model as hidden parameters, do not consider uncertainties in the learning environment and human feedback. Secondly, positions of items have a significant influence on clicks. Online recommendation services recommend multiple commodities to users. Nowadays, a considerable proportion of users visit e-commerce platforms by mobile devices. Due to the limited screen size of mobile devices, positions of items have a significant influence on clicks: 1) Higher positions lead to more clicks for one commodity. 2) The 'pseudo-exposure' issue: Only a few recommended items are shown at first glance and users need to slide the screen to browse other items. Therefore, some recommended items ranked behind are not viewed by users and it is not proper to treat this kind of items as negative samples. While many works model the online recommendation as contextual bandit problems, they rarely take the influence of positions into consideration and thus the estimation of the reward function may be biased.

For the *multi-agent reinforcement learning* setting, we consider the aforementioned new task, namely the multi-module recommendation task. 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 usually are decided by different teams and optimized individually without cooperation, which might result in competition between modules. Existing approaches that solve a similar task rely on an underlying communication mechanism, which violates the constraint of our online system that each module should make decisions independently.

This thesis aims at effectively addressing the above-mentioned issues.

1.3 Major Contributions

The major contributions of this thesis are summarized below.

- We propose a novel probabilistic human feedback model, which considers the uncertainties in humans while giving feedback to agents in an uncertain environment. To handle the new computational challenges brought by the new feedback model, we propose a novel approximate EM approach, which integrates Expectation-Maximization

(EM) and Gradient Descent (GD). We conduct extensive experimental evaluations of our proposed approach on both synthetic scenarios and two different real-world scenarios with human participants. Robustness analysis illustrates that our algorithm performs well even if trainers do not follow our feedback model.

- We propose a novel contextual combinatorial bandit algorithm to address position bias issues. First, we model the online recommendation task as a contextual combinatorial bandit problem and define the reward of a recommended set. Second, we assume that the examination probability is determined by both the position of an item and the last click above the position. We estimate the examination probabilities of various positions based on the User Browsing Model and use the probabilities as weights of samples in a linear reward model. Third, we formally analyze the regret of our algorithm. For T rounds, K recommended arms, d-dimensional feature vectors, we prove an $\tilde{O}(d\sqrt{TK})$ regret bound. Finally, we propose an unbiased estimator to evaluate our algorithm and other baselines using real-world data. An online experiment is implemented in Taobao, one of the largest e-commerce platforms in the world. Results on two CTR metrics show that our algorithm significantly outperforms the extensions of some other contextual bandit algorithms both on the offline and online experiments.

- We propose a novel approach for the multi-module recommendation problem. The first key contribution is a novel multi-agent cooperative reinforcement learning structure. The structure is inspired by a solution concept in game theory called correlated equilibrium [4] in which the predefined signals received by the agents guide their actions. The second key contribution is an entropy-regularized version of the signal network to coordinate agents' exploration. 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.

1.4 Thesis Organization

The remaining part of this thesis is organized as follows. Chapter 2 introduces preliminary knowledge and related work of this thesis. Chapter 3 proposes a novel human

feedback model considering uncertainty and a new method to find the optimal parameters. Chapter 4 presents a novel bandit formulation to estimate the position bias and then recommend items to users. Chapter 5 proposes a novel multi-agent reinforcement learning method to solve the multi-module recommendation problem. Chapter 6 concludes the thesis and discusses possible directions of future work.

Chapter 2

Preliminaries

2.1 Markov Decision Process

Markov decision process [6] (MDP) is a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker, namely an agent. In reinforcement learning, we model the agent-environment interaction as an MDP, which can be represented by a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Fig. 2.1 illustrates the agent-environment interaction. An agent interacts with the environment at each of a sequence of discrete time steps, $t = 1, 2, 3, \ldots$. At each step, the agent obtains some information about the current environment's state $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set including all the possible states. Then, the agent will choose the action, $a_t \in \mathcal{A}$, taken in current time step t, where $\mathcal{A}$ is the set of possible actions. As a consequence of the action, the environment provides a numerical reward, $r_t \in \mathcal{R} \subset \mathbb{R}$, and a new state s_{t+1} based on the transition function $\mathcal{P}(s_t, a_t)$ to the agent in the next time step.

To choose actions,the agent usually maintains a policy π that maps states to actions, where $\pi(a|s)$ is the probability that $a_t = a$ if $s_t = s$. Reinforcement learning methods aim at finding the optimal policy to maximize the return which is the cumulative discounted reward defined as $G = \sum_t \gamma^t r_t$, where $\gamma \in (0, 1)$ is a constant determining the present value of future rewards.

For recommendation tasks, the agent is a recommender system and the environment mainly includes users as well as other components like page structures. The definitions of other elements are summarized as follows.

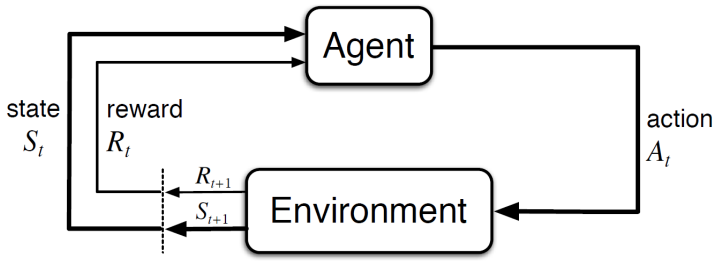


Fig. 2.1: The agent-environment interaction in reinforcement learning [86].

- The time step is usually defined as the number of requests that users send to obtain new recommended items. In other words, it can also be represented by the number of actions generated by the agent.

- The state space usually consists of features of users, which can be divided into static and dynamic features. Static features are relatively fixed within a long period including gender, age, address and so on. Dynamic features such as historical clicked or unclicked commodities of a user change with the time step.

- The action is defined as recommended commodities displayed to users. It can also be some parameters that can determine the recommended commodities. For example, the action is a vector whose dimension is the same as the length of the features of candidate items in practice when the number of candidate items is large. Then, the recommended items are determined by the scores calculated by the weighted sum of the action and the features of items.

- The definitions of rewards are customized according to the objectives of recommendation tasks. To improve the number of clicks, click through rate (CTR) is one of the most popular metrics in many works, which is defined as the ratio of clicked items to recommended items. Other metrics like conversation rate, gross merchandise value are also widely used in tasks that aim at improving the turnover of a company.

Fig. 2.2: Games can be solved by RL including Dota, StarCraft, Go and football.

2.2 Reinforcement Learning

Reinforcement learning is proposed to solve problems such as games illustrated in Fig. 2.2 that can be modeled as MDPs, which contains two main elements: a policy and a value function. A policy defines how to behave at a given time step or state. More formally, it is a mapping from states of the environment to available actions in these states, which can be represented by a table or function. A value function specifies which actions or states are good in an MDP. The value of a state or state-action pair usually is the cumulative discounted reward, which is the long-term reward that can be obtained after the state or state-action pair following a certain policy. Similar to the policy, the value function can also be stored in a table or approximated by a function. We give the mathematical expression in the following paragraphs.

To choose actions or obtain the optimal policies, value functions estimate how good a policy is for a given state or state-action pair. Thus, the agent usually maintains a value-function $V(s)$ or $Q(s,a)$. Given a policy π, we can define the state value function $V_\pi(s)$, which is the expected return starting at s and following π to choose actions:

$$V_\pi(s) = \mathbb{E}_\pi(G_t|s_t = s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{k+t}|s_t = s\right].$$

Then, the objective function can also be represented as $\mathcal{J}(\pi) = \mathbb{E}_{s_0\sim\rho_0} V_\pi(s_0)$, where ρ_0 is the initial distribution of states. The goal of RL is to find the optimal π to maximize $\mathcal{J}(\pi)$.

Similarly, we define the state-action value function $Q_\pi(s,a)$, the expected return taken action a at state s and following π thereafter:

$$Q_\pi(s,a) = \mathbb{E}(G_t|s_t = s, a_t = a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{k+t}|s_t = s, a_t = a\right].$$

According to the above two definitions, some properties of these two value functions can be given:

$$Q_\pi(s_t,a_t) = \mathbb{E}_{s_{t+1}\sim\mathcal{P}(s_t,a_t)}\left[r_{t+1} + \gamma V(s_{t+1})\right],$$

$$V_\pi(s_t) = \mathbb{E}_{s_{t+1}\sim\mathcal{P}(s_t,a_t),a_t\sim\pi(s_t)}\left[r_{t+1} + \gamma V(s_{t+1})\right],$$

$$Q_\pi(s_t,a_t) = \mathbb{E}_{s_{t+1}\sim\mathcal{P}(s_t,a_t),a_{t+1}\sim\pi(s_{t+1})}\left[r_{t+1} + \gamma Q(s_{t+1},a_{t+1})\right]$$

, which are *Bellman equations* that express the relationship between the value of current and successor states or state-action pairs. Based on Bellman equations, the *policy iteration* method is proposed to solve MDPs, which includes two parts: *policy evaluation* and *policy improvement*. By alternatively executing these two parts, we can obtain the optimal policy in tabular cases.

We first introduce the Banach fixed-point theorem, which supports the convergence of iteration methods.

Theorem 2.1 (Banach fixed-point theorem) *Let X be a complete metric space, and f be a contraction on X. Then there exists a unique x^* such that $f(x^*) = x^*$.*

The contraction of a mapping is defined as follows.

Definition 1 (Contraction) *Let X be a metric space, and $f : X \longrightarrow X$. We will say that f is a contraction if there exists some $0 < k < 1$ such that $d(f(x), f(y)) \geq kd(x,y)$ for all $x, y \in X$. The* inf *of such k's is called the contraction coefficient.*

In reinforcement learning settings, X usually is $\mathcal{R}$ and d is the norm. Then we can prove that the Bellman equation has a unique fixed point as follows:

$$\begin{aligned}\|\mathcal{T}^\pi V_1 - \mathcal{T}^\pi V_2\|_\infty &= \|r + \gamma\mathcal{P}_\pi V_1 - r - \gamma\mathcal{P}_\pi V_2\|_\infty \\ &= \gamma\,\|\mathcal{P}_\pi\,(V_1 - V_2)\|_\infty \\ &\leqslant \gamma\,\|V_1 - V_2\|_\infty\end{aligned}$$

, where $\mathcal{T}^{\pi}$ is the Bellman operator representing the left side of the Bellman equation and $\mathcal{P}_{\pi}$ is the transition probabilities following π.

Policy evaluation. Given a policy π, we can compute $\mathcal{J}(\pi)$ and V_{π} by the Bellman function:

$$\begin{aligned} V_{\pi}(s_t) &= \mathbb{E}_{s_{t+1}\sim\mathcal{P}(s_t,a_t),a_t\sim\pi(s_t)}\left[r_{t+1}+\gamma V(s_{t+1})\right] \\ &= \sum_{a}\pi(a|s)\sum_{s',r}\mathcal{P}(s',r|s,a)\left[r+\gamma V(s')\right]. \end{aligned}$$

If $\mathcal{P}$ is known, we can update $V(s)$ by $V(s')$ repeatedly for all the $s \in S$. In each time step, it replaces old values with new values obtained from the old value of the successor states. When the gap between the new and old values is small enough, we stop the iterations. According to the fixed-point theorem, $V(s)$ converges when the number of iterations becomes infinity.

Policy improvement. Since our goal is to find the optimal policy, policy evaluation is not enough and we need to find a way to improve the current policy. Firstly, we can compute $Q(s,a)$ based on the Bellman equation to estimate how good a is at state s. Then, at each state, we change the policy π by

$$\pi'(s) = \arg\max_{a} Q_{\pi}(s,a).$$

The new policy is at least as good as the old one according to the policy improvement theorem.

Theorem 2.2 (Policy improvement theorem [86]) *When* $\pi'(s) = \arg\max_a Q_{\pi}(s,a)$, *the return of* π' *will be no less than that of* π*:*

$$V_{\pi'}(s) \leq V_{\pi}(s).$$

Proof.

$$
\begin{aligned}
V_\pi(s) &\leq Q_\pi\left(s, \pi'(s)\right) \\
&= \mathbb{E}_{\pi'}\left[r_t + \gamma V_\pi\left(s_{t+1}\right) \mid s_t\right] \\
&\leq \mathbb{E}_{\pi'}\left[r_t + \gamma Q_\pi\left(s_{t+1}, \pi'\left(s_{t+1}\right)\right) \mid s_t\right] \\
&= \mathbb{E}_{\pi'}\left[r_t + \gamma \mathbb{E}_{\pi'}\left[r_{t+1} + \gamma V_\pi\left(s_{t+2}\right)\right] \mid s_t\right] \\
&= \mathbb{E}_{\pi'}\left[r_t + \gamma r_{t+1} + \gamma^2 V_\pi\left(s_{t+2}\right) \mid s_t\right] \\
&\leq \mathbb{E}_{\pi'}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V_\pi\left(s_{t+3}\right) \mid s_t\right] \\
&\vdots \\
&\leq \mathbb{E}_{\pi'}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots \mid s_t\right] \\
&= V_{\pi'}(s).
\end{aligned}
$$

If V_π is the same as $V_{\pi'}$, we find the fixed point of the Bellman equation. According to the fixed-point theorem, both π and π' must be the optimal policies.

Value iteration. Policy evaluation usually requires multiple sweeps through the state set to obtain accurate values, which is time-consuming. Value iteration simplifies the policy iteration method by restricting the number of sweeps to one. Formally,

$$
V_\pi(s_t) = \max_a \sum_{s',r} \mathcal{P}(s', r|s, a)\left[r + \gamma V(s')\right]
$$

, which does not lose the convergence guarantees of policy iteration.

Monte-Carlo method. The value iteration and policy iteration assume that the transition probability of the environment is completely known. For many real-world environments, this assumption is hard to be satisfied. Thus, the Monte-Carlo method is proposed to solve MDPs by experience, namely sample sequences of states, actions, and rewards from interaction with an environment. Note that the definitions of V and Q are expectation of the cumulative discounted rewards. One direct way is to take actions following the policy π in an environment and obtain the experience (s_t, a_t, r_t). For each state or state-action pair in the experience, we calculate $V(s_t)$ and $Q(s_t, a_t)$ by the average of $\sum_t r_t$.

Algorithm 1: Value-based RL

Initialize: $Q(s,a)$ for $s \in \mathcal{S},\ a \in \mathcal{A}$;

Reset the environment;
for $t = 1, \ldots, T$ **do**
    Choose a_t based on s_t using policy π derived from Q;
    Obtain s_{t+1} and r_t;
    `// For SARSA`
    Choose a_{t+1} following π (e.g., ϵ-greedy) ;
    $Q(s_t,a_t) = Q(s_t,a_t) + \alpha\,[r_t + Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)]$;
    `// For Q-learning`
    $Q(s_t,a_t) = Q(s_t,a_t) + \alpha\,[r_t + \max_a Q(s_{t+1},a) - Q(s_t,a_t)]$;

2.2.1 Value-Based Approaches

In the above-mentioned policy iteration or value iteration methods, they both need to obtain the expectation during the update. Thus, Temporal-Difference (TD) learning is proposed to estimate the expectation of V values. To estimate V values of a policy π, let π interact with the environment with T time steps. Then, we can update V values using sampled data by

$$V(s_t) = V(s_t) + \alpha\,[r_t + V(s_{t+1}) - V(s_t)]$$

, where α is a constant to decide the weight of new samples.

SARSA. We now consider how to use the TD method to obtain the optimal policy. Instead of estimating V, SARSA directly estimates Q and chooses the best action based on Q values. The update rule is

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha\,[r_t + Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)]$$

, where a_{t+1} is sampled from a policy derived from Q. One of the most widely used methods is $\epsilon - greedy$, where the agent chooses the action having maximal Q value with $1-\epsilon$ probability and randomly chooses an action with ϵ probability. ϵ is a constant to control the frequency of exploration.

Q-learning. Q-learning [96] is one of the most popular methods in RL. The main difference between Q-learning and SARSA is how to choose action in the next state s' during the update. The update rule is defined as

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right].$$

Algorithm 2: Deep Q-learning with Experience Replay [65]

Initialize: the replay buffer $\mathcal{D}$ to capacity N, the action-value function Q_θ with random weights and target action-value function $\hat{Q}_{\theta^-}$ with the same weight as Q.

for $t = 1, \ldots, T$ **do**
- Choose a_t by ϵ-greedy policy;
- Execute action a_t in the environment and observe r_t and s_{t+1};
- Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$;
- Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$;
- Calculate $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(s_{j+1}, a'; \theta^-\right) & \text{for non-terminal } s_{j+1} \end{cases}$;
- Update the weights of Q by minimizing $\left(y_j - Q(s_j, a^j; \theta)\right)$;
- Let $\hat{Q} = Q$ every C steps;

In this formula, the maximal Q value is used no matter what the policy is. For instance, a random policy is deployed in the environment while the action with the maximal Q value is selected during the update.

Deep Q-learning. Above-mention methods focus on tabular cases, where the number of states and actions is finite and limited. For MDPs that involve infinite or a large number of states and actions, approximation methods are proposed. The key idea is to use functions to estimate Q and V rather than storing them for each state and action. With the rise of deep learning, DQN [65] is proposed by adopting a deep neural network to approximate the Q value function. DQN uses a Q network to estimate $Q(s, a)$ by taking s and a as inputs and using stochastic gradient descent (SGD) on the following objective

$$L(\theta) = \mathbb{E}_{a_t, s_t, s_{t+1} \sim \mathcal{P}, \pi}\left[Q(s_t, a_t) - \left[r + \gamma \max_{a'} Q(s', a')\right]\right].$$

Additionally, replay buffer and target network are proposed to improve the stabilization of the policy. Replay buffer stores recently generated samples and provides training data to the Q network. The target network is cloned from the Q network and kept frozen as the optimization target every C steps, where C is a constant. The detailed algorithm is shown in Alg. 2.

Lots of methods are proposed to improve DQN. Dueling DQN [94] decomposes Q values to advantages A and V values, namely $Q(s, a) = A(s, a) + V(s)$, aiming at accelerating

convergence. Double DQN [89] changes the selection of a' to alleviate the overestimating problem by

$$r_j + \gamma \hat{Q}\left(s_{j+1}, \arg\max_{a'} Q(s_{j+1}, a'; \theta); \theta^-\right).$$

Distributional DQN [5] estimates the distribution of Q for each state-action pair rather than a real number to improve the accuracy of the estimated Q function, Prioritized experience replay [78] proposes to select minibatch from replay buffer with probability p_t according to the TD error rather than randomly. More specifically,

$$p_t \propto \left| r_t + \gamma \max_{a'} Q_{\theta^-}(s_{t+1}, a') - Q_\theta(s_t, a_t) \right|^\omega$$

, where ω is a constant to control the shape of the distribution. Rainbow DQN [27] combines above-mentioned methods to reach a better performance.

2.2.2 Policy-Based Approaches

Since value-based methods choose actions by comparing the Q values of available actions, they cannot work when the action space is large or continuous. Thus, policy-based methods are proposed to address this issue by introducing a policy function to make decisions taking a state as input.

Policy gradient theorem [86]. Note that the objective function of RL methods is

$$\mathcal{J}(\theta) = \sum_{s \in \mathcal{S}} \rho_\pi(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) Q_\pi(s,a)$$

, where θ is the parameter of the function representing π. To update the policy function by the gradient descent method, we calculate $\nabla \mathcal{J}(\theta)$ by parts $\pi(a|s;\theta)$ and $Q_\pi(s,a)$, which is too complex to obtain the gradient.

Theorem 2.3 (Policy gradient theorem) *The gradient of $\mathcal{J}$ is*

$$\nabla_\theta \mathcal{J}(\theta) = \sum_{s \in \mathcal{S}} \rho_\pi(s) \sum_{a \in \mathcal{A}} \nabla_\theta \pi(a|s;\theta) Q_\pi(s,a).$$

Proof.

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{s_0 \in \mathcal{S}} \rho(s_0) \cdot \sum_{a_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(a_0|s_0;\theta) \cdot Q_\pi(s_0, a_0) \\
&+ \sum_{s_0 \in \mathcal{S}} \rho(s_0) \cdot \sum_{a_0 \in \mathcal{A}} \pi(a_0|s_0;\theta) \cdot \nabla_\theta Q_\pi(s_0, a_0) \\
&= \sum_{s_0 \in \mathcal{S}} \rho(s_0) \cdot \sum_{a_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(a_0|s_0;\theta) \cdot Q_\pi(s_0, a_0) \\
&+ \sum_{s_0 \in \mathcal{S}} \rho(s_0) \cdot \sum_{a_0 \in \mathcal{A}} \pi(a_0|s_0;\theta) \cdot \left[r_0 + \sum_{s_1 \in \mathcal{S}} \gamma \rho(s_1 \mid s_0, a_0) \cdot \nabla_\theta V(s_1) \right] \\
&= \sum_{s_0 \in \mathcal{S}} \rho(s_0) \cdot \sum_{a_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(a_0|s_0;\theta) \cdot Q_\pi(s_0, a_0) \\
&+ \sum_{s_0 \in \mathcal{S}} \sum_{s_1 \in \mathcal{S}} \rho(s_0 \to s_1, 1, \pi) \cdot \left[\sum_{a_1 \in \mathcal{A}} \nabla_\theta \pi(a_1|s_1;\theta) Q_\pi(s_1, a_1) \right] \\
&= \sum_{s \in \mathcal{S}} \sum_{s_0 \in \mathcal{S}} \sum_t \gamma^t \rho(s_0) \rho(s_0 \to s, t, \pi) \cdot \sum_{a \in \mathcal{A}} \nabla_\theta \pi(a|s;\theta) \cdot Q_\pi(s, a) \\
&= \sum_{s \in \mathcal{S}} \rho_\pi(s) \cdot \sum_{a \in \mathcal{A}} \nabla_\theta \pi(a|s;\theta) \cdot Q_\pi(s, a)
\end{aligned}
$$

The last equation is due to the definition $\rho_\pi(s) = \sum_{s_0 \in \mathcal{S}} \sum_t \gamma^t \rho(s_0) \rho(s_0 \to s, t, \pi)$

REINFORCE. By simply applying the policy gradient theorem, the REINFORCE method estimates Q by the Monte-Carlo method. More specifically, trajectories (s_t, a_t, r_t) are obtained by following the policy function π_θ. The Q value or return is estimated by

$$Q = G = \sum_{t=0,1,\ldots} \gamma^t r_t.$$

Then, the policy π_θ can be updated by the gradient

$$\nabla_\theta \log \pi(a|s;\theta) Q_\pi(s, a).$$

We add the log sign because the samples we obtained following π and the term $\frac{1}{\pi(a|s;\theta)}$ is used to alleviate the bias. Formally,

$$\mathbb{E}_\pi \frac{\nabla_\theta \pi(a|s;\theta)}{\pi(a|s;\theta)} Q_\pi(s, a).$$

Actor-Critic. Similar to value-based methods, the Q values can also be approximate by a function to reduce variance during training. We then get the Actor-Critic

Algorithm 3: Deep deterministic policy gradient [52]

Initialize: the replay buffer $\mathcal{D}$ to capacity N, the critic network Q_θ and the actor π_w with random weights, and the target critic $\hat{Q}_{\theta^-}$ and actor $\hat{\pi}_{w^-}$ with the same weight as Q and π.

for $t = 1, \ldots, T$ **do**

- Choose a_t by actor with random noisy;
- Execute action a_t in the environment and observe r_t and s_{t+1};
- Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$;
- Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$;
- Calculate $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \hat{Q}\left(s_{j+1}, \hat{\pi}(s_{j+1}); \theta^-\right) & \text{for non-terminal } s_{j+1} \end{cases}$;
- Update the weights of π by the gradient $\nabla_a Q(s_j, a; \theta)_{a=\pi(s_j;w)} \nabla_w \pi(s_j; w)$;
- Update the weights of Q by minimizing $\left(y_j - Q(s_j, a^j; \theta)\right)$;
- Update the target networks:

$$\hat{Q} = \alpha Q + (1 - \alpha)\hat{Q}$$
$$\hat{\pi} = \alpha \pi + (1 - \alpha)\hat{\pi}$$

method. The policy is represented by the actor that selects actions and the estimated value function is the critic due to the role that criticizes the actions chosen by the actor. The actor can be trained by gradient descent method using the values provided by the critic.

Deep deterministic policy gradient. DDPG [52] represents the functions of actor and critic by deep neural networks. Due to the property of the deterministic policy, this method can be updated by samples obtained by other policies or historical policies. Thus, the replay buffer proposed in DQN is adopted to accelerate training. The details is shown in Alg. 3. α is a constant controlling the learning rate.

Soft actor-critic. SAC [23] modifies the objective function by adding an entropy term to promote exploration. The new objective function is

$$\mathcal{J}(\pi) = \sum_t \mathbb{E}_{s_t, a_t \sim \rho_\pi} \left[r_t + \alpha \mathcal{H}(\pi(\cdot \mid s_t)) \right]$$

, where α is a constant and $\mathcal{H}(\pi(\cdot \mid s_t))$ is the entropy of π. Differing from DDPG, the actor is to minimize the expected KL-divergence:

$$\mathcal{J}_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[D_{\mathrm{KL}} \left(\pi_\phi(\cdot \mid s_t) \,\|\, \frac{\exp\left(Q_\theta(s_t, \cdot)\right)}{Z_\theta(s_t)} \right) \right].$$

The partition function Z normalizes the distribution.

TRPO and PPO. Trust Region Policy Optimization and Proximal Policy Optimization [79, 80] methods are policy-based methods with an additional constraint

$$D_{\mathrm{KL}}(\pi^{old}, \pi^{new}) < \delta$$

, where π^{old} and π^{old} are policies before and after an update. δ is a hyper-parameter that controls the maximal gap between two policies. The difference between TRPO and PPO is how to obey the constraint during updates. TRPO uses the conjugate gradient method, which is complex and time-consuming. PPO proposes to transfer the problem to an unconstrained optimization problem by treating the constraint as a penalty term in the objective function. Then, the stochastic gradient descent method can be used to update.

2.2.3 Multi-Arm Bandit Approaches

Multi-arm bandit problems consider a simplified setting of MDPs, where the number of states is one. The objective of multi-arm bandit methods is to minimize the regret:

$$R(T) = \sum_{t=0}^{T} \left(r_t^* - r_t(a_\pi)\right)$$

, where r_t^* is the optimal reward obtained at each time step and $r_t(a_\pi)$ represents the reward obtained by a policy π. To find the optimal action or arm, one crucial issue is the trade-off of exploration and exploitation when selecting action at each time step. The upper confidence bound (UCB) method [2] is proposed by using the upper confidence bound of each arm to decide pulled arms. Formally,

$$a_t = \arg\max_a Q(a) + \sqrt{\frac{2\log t}{N_t(a)}}$$

, where $Q(a)$ is the mean of reward of action a and $N_t(a)$ is the number of times that action a has been chosen before t time step. The latter term of the equation is the upper confidence bound of $Q(a)$.

In order to handle the large action space, the contextual bandit method, LinUCB [49], is proposed by assuming that each arm processes a feature vector. It assumes that the

reward of an arm is linear in its feature and hence use a linear function to approximate the rewards of arms. Formally,

$$\mathbb{E}[r] = \theta^t x$$

, where θ is the unknown parameters and x is the feature of an arm. By decomposing θ to $A^{-1}b$, the algorithm updates by

$$A = A + xx^T$$

$$b = b + rx.$$

To select action, the upper confidence bound of each arm is calculated by

$$x^T\theta + \alpha\sqrt{x^T A^{-1} x}.$$

2.3 Reinforcement Learning for Recommender System

Many deep reinforcement learning methods are used in the recommender system domain. Existing 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 rewards such as return time [105]. An algorithm is proposed to use two individual LSTM modules for items with short-term and long-term rewards respectively [108]. Reward shaping is a straightforward method to train a policy end to end by defining various reward functions. The diversity of recommended sets is added to the reward function [57]. Transition probabilities from users' actions (such as click) to purchase are used as rewards [68]. A network is used to predict transition probabilities and the output is considered as rewards for actions, which is trained by historical data. Modified MDPs for recommendation are proposed by redefining the structure of reward function and the transition function respectively [29, 31]. A method is proposed to improve profit by detecting fraud transactions [100]. Transition function is divided into three sub-functions that indicate the probabilities that users leave, purchase, or click.

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 [102]. A hierarchical algorithm is proposed to aggregate topics, blog posts, and products on one web page [87]. Similarly, the Double-Rank Model is proposed to learn how to rank the display positions and with which documents to fill those positions [67]. The positions of topics, posts and products are firstly decided by the high-level policy and low-level policies determine what to recommend. The problem 'when and where should advertising be added?' is addressed for web pages that contain advertising [101]. With a given recommended list, the rewards of additional advertising in different positions are learned. Since the positive reward is sparse, a work treats negative samples as additional information in training [104]. The main idea is to leverage two networks to process negative samples and positive samples respectively.

Third, other works focusing on the large space of actions and states usually adopt clustering techniques to reduce the space [13, 85]. 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.

2.3.1 Combinatorial Multi-Arm Bandit

Since we usually need to recommend a set of items rather than one item to users, traditional bandit algorithms cannot be directly applied in many scenarios. Thus, we focus on a special kind of bandit algorithm called combinatorial bandit, which is proposed to pull a set of arms in each round. The formulation is first studied in [11] to handle the situation where the reward is decided by a set of arms, such as recommendation and advertising. The contextual version of this problem is proposed in which each arm possesses a contextual vector that determines the reward [72, 97].

Combinatorial bandit with click models. The idea of utilizing click models in bandit algorithms to model the influence of positions has been explored in prior research works [38, 44, 46–48, 51, 106]. The Cascade Model [15] merges with multi-armed bandit, combinatorial bandit, contextual bandit and contextual combinatorial bandit respectively [46, 47, 51, 107]. The Cascade Model assumes that a user scans items from top to bottom until they click one. Thus, their algorithms cannot be applied to the scenarios where users click more than one item. To address this disadvantage, Katariya et al. [38] combine a

multi-armed bandit algorithm with the Dependent Click Model, an extension of the Cascade Model for multiple clicks scenarios. This model assumes that after users clicked an item, they may still continue to examine other items, if they are not satisfied. However, all these models assume that users will not browse items behind the position of the last click and users' attention on each item before the last click is the same, which is not suitable for our problem. In contrast, the Position-Based Model [44,48] assume that click rates of items are affected by their displayed positions. Komiyama et al. [44] consider the influence of ads differing from traditional bandit problems.

2.3.2 Interactive Reinforcement Learning

Interactive reinforcement learning is a branch that aims at hastening the learning process or training an agent by human feedback in some environments that rewards are not available or sparse.

Many existing works [9,22,42,59] show the efficiency of learning from human trainers. Knox et al. [41–43] propose the TAMER framework in which human feedback is learned by a supervised learning method and then used to provide rewards to the agent. The TAMER framework is merged with the deep neural network to address more difficult problems [95]. MacGlashan et al. [61] propose a novel Actor-Critic algorithm considering that human feedback relates to agents' policy. Policy shaping algorithms [9, 22] utilize the difference between 'good' and 'bad' labels to determine the best action by a Bayesian approach. How a trainer implicitly expects a learner to interpret feedback is studied in [28]. Trainers are allowed to directly slightly change the action in paper [70]. Two works [12, 62] focus on training agents to learn the meaning of language or commands. However, most of these approaches do not consider the feedback model of trainers except Loftin et al. [58, 59, 62]. They consider trainers' feedback model and propose a strategy-aware algorithm to infer the meaning of no feedback via the trainer's feedback history.

2.3.3 Multi-agent Reinforcement Learning

In multi-agent reinforcement learning, the basic MDP model is extended to multi-player Markov game [54], which is represented by $(N, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. The main difference is the number of agents N. If we individually optimize each agent, the policy is hard to converge

due to the nonstationarity of other agents. To alleviate this problem, centralized training decentralized execution (CTDE) methods propose to use the information of all agents when estimating the Q values.

MADDPG. The most famous algorithm is MADDPG [60], in which each agent maintains a critic and an actor. Each critic is a centralized action-value function that takes as input the actions of all agents in addition to state information, and outputs the $Q^i(s, a_1, \ldots, a_N)$ values for the i-th agent. Then, similar to DDPG, the Q value is used to train the actor.

COMA. In many environments like football game, the reward is for a team rather than each person. Thus, we need to consider how to assign rewards for each agent, which is called credit assignment problem. Counterfactual multi-agent (COMA) policy gradients [20] method proposes a counterfactual function $A(s, a) = Q(s, a) - \sum_{a'} \pi^i(a' \mid s)Q(s, (a', a^{-i}))$, where a^{-i} is the action without the i-th agent. The function estimates the effect when the i-th agent executes other actions to determine the contribution of the action a^i. The counterfactual value is used to update the actor of each agent.

QMIX. To extend DQN to multi-agent environment, the credit assignment problem also exists. Different from COMA, QMIX [75] provides an explicit way to determine the Q values of all agents. A mixing network is proposed to estimate the global Q values taken Q values of all the agent as inputs, which satisfies monotonicity: the total Q value should increase with the rise of individual Q values. More specifically, the weights of the mixing network is forced to be non-negative by the absolute value sign. Each agent executes depending on their Q value functions trained together with the mixing network.

Communication. In real world, many scenarios allow communication between agents such as robot football. Many papers propose multi-agent reinforcement learning methods with a communication channel. Although these methods violate the centralized training decentralized execution rule slightly, they work in specific environments. Various structures of the communication channel are proposed [18, 34, 83, 84].

Recommendation via multi-agent RL. Some works use multi-agent frameworks to promote cooperation between different pages in recommendation scenarios. Inspired by RL methods involving communication like [92], a multi-agent RL framework is proposed where agents can communicate by messages [17]. A model-based RL algorithm

is proposed by using neural networks to predict transition probability and shape reward [103].

2.4 Chapter Summary

This chapter reviews the background of this thesis, which consists of basic reinforcement learning methods, deep reinforcement learning methods and some works that apply reinforcement learning to recommendation problems. We demonstrate both tabular and approximation methods to solve different kinds of MDPs, including value-based methods and policy-based methods. Additionally, the convergence and optimality of some methods are illustrated. We also introduce reinforcement learning methods that involve neural networks to approximate value functions or representing the policy, which will play an important role in Chapter 5 where we propose a novel multi-agent reinforcement learning method for a recommendation task. Moreover, we introduce some works that apply multi-arm bandit methods and two kinds of reinforcement learning methods into the recommendation domain, which can be considered as the literature review for the next three chapters.

Chapter 3

Learning Behaviors with Uncertain Human Feedback

3.1 Introduction

Training agents for tasks with human feedback has many application scenarios such as games and recommendation systems. Previous works [58, 59] recognize the fact that humans as trainers may have different training strategies, i.e., different habits of giving positive/negative feedback, or not giving feedback in some confusing cases. They usually solve the problem using Expectation Maximization (EM) by treating human feedback model as hidden parameters. However, in environments with uncertainties, trainers may not know exactly the optimal action themselves, and therefore could only provide feedback with uncertainty. For example, in many games like Texas hold'em poker, the rewards of all the actions are not deterministic. A sub-optimal action may get huge reward in some cases. Then, trainers are likely to provide positive feedback to the sub-optimal action or not sure which kind of feedback they should give. Moreover, existing works do not differentiate non-optimal actions, i.e., they assume the expected rewards of different sub-optimal actions are the same, whereas in practice, different actions might have different levels of optimality, and the expected rewards of different sub-optimal actions are usually dependent on their distances to the optimal one.

In this chapter[1], we address these two issues by proposing a novel human feedback model, which takes the uncertainty of trainer's feedback into account. Instead of assuming that humans will only give positive feedback when the action taken by the agent

[1]This chapter has been published in [26].

is optimal, we model the way that humans give feedback as a probability distribution (e.g., Gaussian), where the probability of giving positive (negative) feedback increases (decreases) w.r.t. the distance of action towards the optimal action. It brings new challenges, in the sense that the newly introduced probabilistic human feedback model makes the calculation of the integration intractable. To this end, we propose a novel approximate EM algorithm, which decomposes the original EM algorithm into two procedures. First, we use an approximated EM step where we assume a subset of the parameters in the human feedback model (e.g., standard deviation of the Gaussian) are fixed and known *a-priori*. In the second step, we use the Gradient Descent (GD) method to compute the most likely parameter values that we fix in the first step.

Our key contributions are summarized as follows. First, we propose a novel probabilistic human feedback model, which considers the uncertainties in humans while giving feedback to agents in an uncertain environment. Second, to handle the new computational challenges brought by the new feedback model, we propose a novel approximate EM approach, which integrates Expectation-Maximization (EM) and Gradient Descent (GD). Last, We conduct extensive experimental evaluations of our proposed approach on both synthetic scenarios and two different real-world scenarios with human participants. 1) We show that our approach outperforms existing works under various settings and metrics. 2) Ablation study shows the effectiveness of our proposed techniques. 3) Robustness analysis demonstrates that our algorithm still performs well even if trainers do not follow our feedback model.

3.2 Modelling Uncertain Trainer Feedback

In one time period, we define state $s \in S$, where S is the set of states. After observing current state s, the agent chooses an action $a \in A$. Following [58], we define a discrete action space $A = \{a_1, ..., a_K\}$. After the agent chooses an action, the trainer could provide feedback $f \in F = \{f^+, f^-, f^0\}$ to describe his/her degree of satisfaction, where $\{f^+, f^-, f^0\}$ corresponds to positive, negative and no feedback respectively. Positive feedback indicates 'good' and negative feedback is 'bad', while no feedback could be used to indicate other information such as confusion or 'not bad' (but also not good). All of

the trainer's feedback will be recorded in history $h = \{[s_t, a_t, f_t]|t = 1, \ldots, T\}$, where T is current time period. $s_t \in S$, $a_t \in A$ and $f_t \in F$ are state, action and feedback obtained in the time period t.

To model the uncertainty when trainers give feedback, we propose a novel probabilistic human feedback model, the probabilities of obtaining different kinds of feedback are represented by $p(a, \lambda(s); \sigma, \mu, f)$ in state s, where a is the current action, $\lambda(s)$ is the trainer's preferred action in state s. σ and $\mu = \{\mu^+, \mu^-\}$ are unknown parameters of the probability function. Formally,

$$p(a, \lambda(s); \sigma, \mu, f) = \begin{cases} p^+(a, \lambda(s); \sigma, \mu^+) & f = f^+ \\ p^-(a, \lambda(s); \sigma, \mu^-) & f = f^- \\ 1 - p^+(a, \lambda(s); \sigma, \mu^+) & \\ \quad -p^-(a, \lambda(s); \sigma, \mu^-) & f = f^0 \end{cases}$$

where σ and $\mu = \{\mu^+, \mu^-\}$ control variance and the maximum probability of providing different kinds of feedback (positive feedback and negative feedback) respectively. The parameters of the model could be adjusted to approximate different trainers' habits of providing feedback. Many popular distributions can be written in this form. For example, the Gaussian function

$$\mu e^{-\frac{(a-\lambda(s))^2}{2\sigma^2}}$$

and a simple extension of the density function of the Logistic distribution

$$\frac{\mu}{\sigma(1 + e^{-\frac{(a-\lambda(s))}{\sigma}})}.$$

We can divide the functions p^+ and p^- into $\mu \in [0, 1]$ and another term $\hat{e}_\sigma \in [0, 1]$, formally,

$$p^+ = \mu^+ \hat{e}_\sigma$$

and

$$p^- = \mu^- \left[1 - (1 - \epsilon)\hat{e}_\sigma\right]$$

, where ϵ is a positive constant whose value is close to 0 in order to ensure that the probability of obtaining negative feedback is not 0 at $a = \lambda(s)$. We constrain that the upper bound of the sum of $p^+(a, \lambda(s); \sigma, \mu^+)$ and $p^-(a, \lambda(s); \sigma, \mu^-)$ cannot be larger than 1 for any s and a. Then, we have

$$\begin{aligned} &\mu^+ \hat{e}_\sigma + \mu^- \left[1 - (1 - \epsilon)\hat{e}_\sigma\right] \leq 1 \\ \Rightarrow &1 - \mu^- - \hat{e}_\sigma \left[\mu^+ + \mu^-(1 - \epsilon)\right] \geq 0. \end{aligned}$$

where $\hat{e}_\sigma \in [0, 1]$. Notice that the function on the left side is monotonous for the term $\hat{e}_\sigma$, which means that the minimum of the function is obtained when $\hat{e}_\sigma = 1$ since $\mu \geq 0$ and $\epsilon \to 0^+$. Thus, we have

$$1 - \mu^+ - \epsilon\mu^- \geq 0 \Rightarrow \mu^+ + \epsilon\mu^- \leq 1.$$

By using the property $\mu^- \in [0, 1]$, since the maximal value of the function in the left side should be less than 1, we let $\mu^- = 1$ and get the upper bound of μ^+:

$$\mu^+ + \epsilon\mu^- \leq 1 \Rightarrow \mu^+ \leq 1 - \epsilon.$$

The intuitive interpretation of μ^+ and μ^- is that they are the maximum probabilities of obtaining positive and negative feedback respectively. That is, μ^+ is the probability of getting positive feedback when the current action is the trainer's most preferred one and μ^- is the probability of getting negative feedback at the trainer's most disliked action. When the action a is different from $\lambda(s)$, the probability of obtaining positive feedback declines while more negative feedback will be received.

3.3 Approximate Expectation Maximization

Since Bayesian approaches have shown the efficiency in inferring from a small quantity of data, we adopt the Maximum Likelihood Estimation (MLE) to estimate the policy function λ with two unknown parameters μ and σ in the probabilistic human feedback model, where the objective function is represented as:

$$\arg\max_\lambda P(h|\lambda; \mu, \sigma), \tag{3.1}$$

where $h = \{[s_t, a_t, f_t]|t = 1, \ldots, T\}$ is the history of state, action and feedback. Note that this MLE cannot be calculated directly since there are two sets of unknown variables, policy λ and parameters (μ,σ), in this equation and there is no explicit bound for σ. One classical way of handling this kind of problems is to use the EM algorithm where the i-th EM update step is represented as:

$$\lambda_{i+1} = \arg\max_\lambda \iiint P(\mu^+, \mu^-|h, \lambda_i) \cdot \ln[P(h, \mu^+, \mu^-|\lambda)] d\mu^+ d\mu^- d\sigma.$$

However, it is easy to see that this integral is intractable, as 1) there is no explicit upper bound for σ and 2) the large dimension of the hidden variables significantly increases the computational complexity of calculating the integral (which is usually calculated using numerical methods as there are no analytical solutions). As a result, we propose a novel approximate EM algorithm, in which an alternating optimization method is used to solve this MLE by two steps: 1) Update λ with σ fixed and treat μ as a hidden variable by EM algorithm. 2) Update σ with λ fixed by GD algorithm and use a trick to ignore μ. [2]

3.3.1 Updating λ with σ Fixed

Since μ^+ and μ^- mean the maximum probabilities of obtaining positive and negative feedback, we know that the range of μ^- is $[0, 1]$ and $0 \leq \mu^+ \leq 1 - \epsilon$. By treating μ^+ and μ^- as bounded latent variables, λ can be updated by the Expectation-Maximization (EM) algorithm when σ is fixed, which is inspired by [59]. In the expectation step, we compute the expectation of a log-likelihood estimate with respect to a bounded latent variable. In the maximization step, the expectation is maximized with respect to another unknown variable. Treating μ^+, μ^- as latent variables, the i-th EM update step is:

$$\lambda_{i+1} = \arg\max_\lambda \int_0^1 \int_0^{1-\epsilon} P(\mu^+, \mu^-|h, \lambda_i) \cdot \ln[P(h, \mu^+, \mu^-|\lambda)] d\mu^+ d\mu^-, \tag{3.2}$$

where λ_i is the inferred preference in i-th step. To ensure that the probability of receiving positive and negative feedback is not larger than 1, the upper bound of μ^+ is defined by $1 - \mu^-$.

Using the Bayes' theorem and the property of logarithm, we obtain

$$P(\mu^+, \mu^-|h, \lambda_i) = \frac{P(h|\mu^+, \mu^-, \lambda_i) P(\mu^+, \mu^-|\lambda_i)}{P(h|\lambda_i)}, \tag{3.3}$$

and

$$\ln[P(h, \mu^+, \mu^-|\lambda)] = \ln[P(h|\mu^+, \mu^-, \lambda)] + \ln[P(\mu^+, \mu^-|\lambda)]. \tag{3.4}$$

Notice that $P(h|\lambda_i)$ is the marginal probability and does not involve any variable. Thus we can treat it as constant and remove it from Eq.(3.3). Since μ^+ and μ^- define the way that a trainer provides feedback and λ is the set including the optimal actions in various states, $\{\mu^+, \mu^-\}$ is independent of λ and thus

$$P(\mu^+, \mu^-|\lambda) = P(\mu^+, \mu^-|\lambda_i) = P(\mu^+, \mu^-).$$

[2] An alternative is to use gradient-based methods at both steps [7]. However, this is infeasible due to the complex problem structure and the difficulty in deriving gradients at both steps.

We assume that μ^+ and μ^- are uniformly distributed due to the lack of the prior knowledge about μ^+ and μ^-. Therefore, we get $p(\mu^+, \mu^-) = 2$ by solving the equation

$$\int_0^1 \int_0^{1-\epsilon} p(\mu^+, \mu^-) d\mu^+ d\mu^- = 1.$$

Thus, the term $P(\mu^+, \mu^-|\lambda_i)$ can be removed from Eq.(3.3). For the second logarithmic term of Eq.(3.4), since

$$\int_0^1 \int_0^{1-\epsilon} P(\mu^+, \mu^-|h, \lambda_i) \ln[2] d\mu^+ d\mu^- \tag{3.5}$$

is not related to λ, we can ignore it. Finally, the objective could be simplified to

$$\lambda_{i+1}(s) = \arg\max_{\lambda} \int_0^1 \int_0^{1-\epsilon} P(h|\mu^+, \mu^-, \lambda_i) \cdot \ln[P(h^s|\mu^+, \mu^-, \lambda(s))] d\mu^+ d\mu^-, \tag{3.6}$$

where h^s is the history containing the state s. Utilizing $h = \{[s_t, a_t, f_t]|t = 1, \ldots, T\}$, we can calculate the integral, since the probability of obtaining h given μ and λ is:

$$\begin{aligned}
P(h|\mu^+, \mu^-, \lambda) &= \prod_T P(f_t|a_t, s_t, \lambda(s_t), \mu^+, \mu^-) \\
&= \prod_T p(a_t, \lambda(s_t); \sigma, \mu, f_t) \\
&= \prod_{s_h \in S} p^+(a_h, \lambda(s_h); \sigma^+, \mu^+)^{|n^+_{s_h}|} \cdot p^-(a_h, \lambda(s_h); \sigma^+, \mu^+)^{|n^-_{s_h}|} \\
&\quad \cdot p^0(a_h, \lambda(s_h); \sigma^+, \mu^+)^{|n^0_{s_h}|},
\end{aligned}$$

where $|n^+_{s_h}|$, $|n^-_{s_h}|$, $|n^0_{s_h}|$ are the numbers of three types of feedback in the state $s_h \in h$. For p^+ and p^-, notice that $\ln[\mu \hat{e}_\sigma] = \ln[\mu] + \ln[\hat{e}_\sigma]$. Since $\ln[\mu]$ is not related to λ, we ignore it during the calculation to prevent divergence. In practice, we compute expectations for all the actions that are available in a sate s and select the action with the maximal expectation as the policy $\lambda(s)$ for the state s.

Eq.(3.6) shows the natural way to utilize feedback from other states. When we compute the optimal action in state s, the first term $P(h|\mu^+, \mu^-, \lambda_i)$ considers all the historical feedback and affects the result of the Eq.(3.6). In this way, both feedback model and historical feedback are taken into consideration to infer the best action.

Algorithm 4: Adaptive Bayesian Learning with Uncertain Feedback (ABLUF)

Initialize: $\lambda = randomPolicy(), \epsilon = 0.01,$
$h = [\ ], t = 0, done = 0;$

while $done \neq 1$ **do**
  $s_t = getState();$
  $a_t = \lambda(s);$
  $takeAction(a_t);$
  $f_t = getFeedback();$
  $h = [h; [s_t, a_t, f_t]];$
  repeat
    $\lambda' = \lambda;$
    $\lambda = \arg\max_\lambda \int_0^1 \int_0^{1-\epsilon} P(h|\mu^+, \mu^-, \lambda') \cdot \ln[P(h^s|\mu^+, \mu^-, \lambda(s))] d\mu^+ d\mu^-;$
  until $\lambda' = \lambda;$
  $\sigma = \sigma - \alpha \frac{1}{n} \sum_s \sum_a \nabla_\sigma L(\sigma; f, \lambda);$
  $done = getDone();$
  $t = t + 1;$

3.3.2 Updating σ with λ Fixed

Since our objective function is calculated based on probabilities provided by the feedback model, the accuracy of the feedback model affects the result significantly. To obtain an accurate human feedback model, we use GD method to minimize the square loss function between the inferred feedback model $p(a, \lambda(s); \sigma, \mu, f)$ and the real one, i.e.,

$$[p(a, \lambda(s); \sigma, \mu, f) - p(f|a, s)]^2. \tag{3.7}$$

However, since we calculate the integral under μ, the value of μ is unknown. We cannot compute the gradient of the loss function. since $p(a, \lambda(s); \sigma, \mu, f)$ is $\mu^+ \hat{e}_\sigma$ and $\mu^-[1 - (1 - \epsilon)\hat{e}_\sigma]$, the second term with σ actually is the estimation of the ratio between the probability at action a and the maximum probability of receiving positive or negative feedback. That is,

$$\begin{aligned} ratio_p^+ =& \hat{e}_\sigma \approx \frac{p(f^+|a, s)}{p(f^+|\lambda(s), s)} = ratio_a(f^+) \\ ratio_p^- =& [1 - (1 - \epsilon)\hat{e}_\sigma] \approx \frac{p(f^-|a, s)}{p(f^-|a_s^-, s)} = ratio_a(f^-), \end{aligned} \tag{3.8}$$

where $a_s^- = \arg\max_a d(a, \lambda(s))$ is the action that a trainer dislikes the most, since this action has the maximum probability of receiving negative feedback. We transform the loss function to

$$L(\sigma; f, \lambda) = [ratio_p - ratio_a(f)]^2. \tag{3.9}$$

We omit μ in L, since it does not appear in the new formulation of the loss function.

The gradient $\nabla_\sigma L(\sigma; f, \lambda)$ of the new loss function could be computed as follows. For f^+, we have

$$[ratio_p^+ - ratio_a(f^+)]\nabla_\sigma ratio_p^+. \tag{3.10}$$

For f^-, we have

$$[ratio_p^- - ratio_a(f^-)]\nabla_\sigma ratio_p^-. \tag{3.11}$$

Thus,

$$\begin{aligned}\nabla_\sigma L(\sigma; f, \lambda) =&[ratio_p^+ - ratio_a(f^+)]\nabla_\sigma ratio_p^+ \\ &+ [ratio_p^- - ratio_a(f^-)]\nabla_\sigma ratio_p^-,\end{aligned}$$

where $ratio_a(f^+)$ and $ratio_a(f^-)$ could be obtained from the historical feedback. For example, if the probabilities of offering positive feedback are 0.5 and 0.9 in action a and the optimal action $\lambda_i(s)$ respectively, $ratio_a(f^+) = \frac{0.5}{0.9}$. We implement the Gradient Descent method to update the parameters and all the historical feedback of all actions is used to compute the gradient and then update σ:

$$\sigma = \sigma - \alpha \frac{1}{n} \sum_s \sum_a \nabla_\sigma L(\sigma; f, \lambda), \tag{3.12}$$

where $n = |S| \times |A|$ is the total number of states-action pairs.

The Adaptive Bayesian Learning with Uncertain Feedback (ABLUF) algorithm is shown in Algorithm 4. The value of step size α is given in the experiment section based on the property of the gradient. The variable *done* is obtained from the environment and the trainer, which is also introduced in the next section. After selecting and taking the action in Lines 3-4, the system will obtain the trainer's feedback and record it in h. In Lines 7-10, the EM algorithm calculates the preferences of the trainer by Eq.(3.6) based on the records stored in h. After updating λ, the parameter σ is updated at Line 11 by Eq.(3.12).

3.4 Experiment Evaluations

We implement two sets of experiments for two environments with human subjects: training a virtual dog to catch rats and learning users' preferences of lighting. We also evaluate the performance, convergence, and robustness of our proposed algorithm in simulated experiments. Code can be found at `https://github.com/hlhllh/Learning-Behaviors-with-Uncertain-Human-Feedback`.

3.4.1 Choice of Human Feedback Model

In general, a good human feedback model is expected to have the following good properties: 1) It captures the uncertainty of a trainer to give feedback. 2) The optimal action is unique in each state. 3) When the action becomes far away from the optimal by a same amount, trainer's satisfaction will decrease by a similar amount.

For these concerns, we use Gaussian functions as the human feedback model due to its simplicity. [3] Thus, we define

$$p^+(a, \lambda(s); \sigma, \mu^+) = \mu^+ e^{-\frac{(a-\lambda(s))^2}{2\sigma^2}} \tag{3.13}$$

$$p^-(a, \lambda(s); \sigma, \mu^-) = \mu^-[1 - (1-\epsilon)e^{-\frac{(a-\lambda(s))^2}{2\sigma^2}}] \tag{3.14}$$

and the gradients of $L(\sigma; f, \lambda)$ are

$$2\frac{(a-\lambda_i(s))^2}{\sigma^3}[ratio_p^+ - ratio_a(f^+)]ratio_p^+$$

and

$$-2(1-\epsilon)\frac{(a-\lambda_i(s))^2}{\sigma^3}[ratio_p^- - ratio_a(f^-)]ratio_p^-$$

for f^+ and f^- respectively. The learning rate α is set to be $0.4\sigma^3$. The σ^3 term aims at eliminating the denominator to avoid a very small update of the parameter σ when the value of σ is large.

[3] Our robustness analysis shows that the Gaussian functions model well adapts to situations where the way trainers give feedback is different from the Gaussian functions. Other forms of human feedback model could also be considered, such as the probability density functions of Cauchy distribution, Logistic distribution, etc. We leave the exploration of various forms of human feedback model as future work.

3.4.2 Baselines

We compare with two state-of-the-art algorithms: Inferring Strategy-Aware Bayesian Learning (ISABL) algorithm [58] and Upper Confidence Bound (UCB) algorithm [3].

- ISABL [58] is a Bayesian learning algorithm aiming at utilizing human feedback to train an agent. Expectation-Maximization method is used to calculate the best action. This algorithm outperforms traditional reinforcement approaches like Q-learning, which is a Bayesian learning algorithm aiming at utilizing human feedback to train an agent. They assume that the human trainer only provides positive feedback when the agent selects an optimal action. If the agent chooses other actions, the trainer will give negative feedback. No feedback is also considered to determine the training policy. The error rate of the ISABL algorithm in our experiment is 0.1 following the original setting.

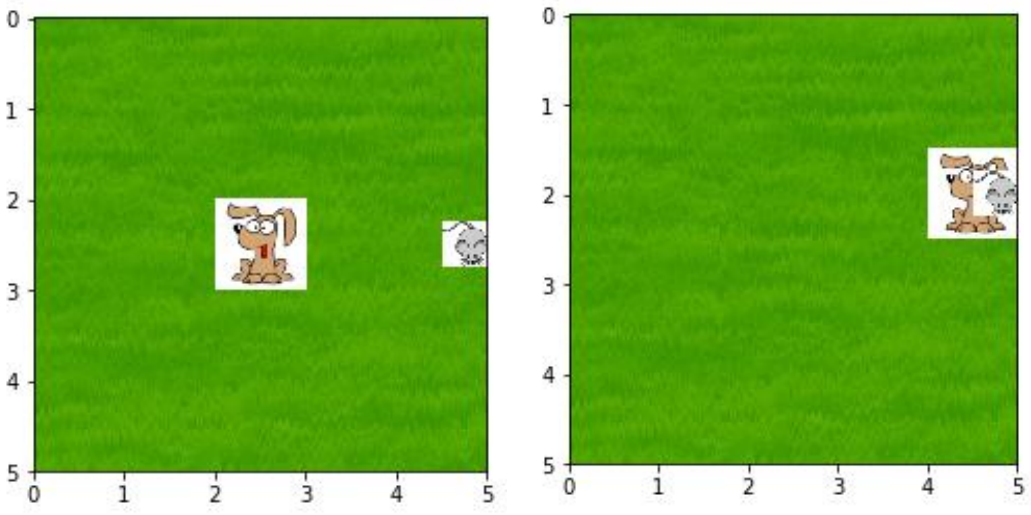
- The upper confidence bound (UCB) algorithm [3] calculates the upper confidence bound of the expected reward for each action and chooses an action with maximum UCB value. In our experiment, we assign values for different kinds of feedback, that is, $[f^+, f^-, f^0] \rightarrow [1, -1, 0]$. After receiving the feedback, the upper confidence bound for an action will be computed by

$$UCB(s,a) = \mathbb{E}[r_{s,a}] + \sqrt{\frac{2\log t_{s,a}}{t_s}},$$

  where the $\mathbb{E}[r_{s,a}]$ is the expected feedback value conditioning on the feedback for action a in state s, $t_{s,a}$ means the number of times that the algorithm chooses a in state s, and t_s is the total number of times that state s appears.

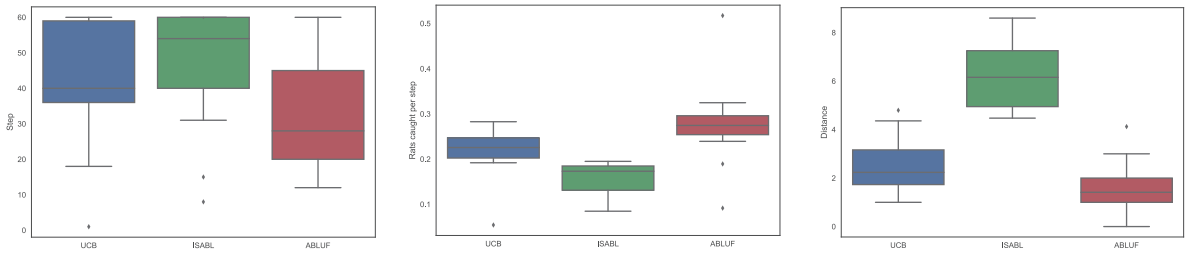
3.4.3 Training a Virtual Dog

In our first experiment, trainers are required to train a virtual dog to catch rats. At the beginning of each step, the dog is at the center of the figure and a rat appears at one of four edges of the figure. Each edge can be considered as a state in this experiment. Then, the rat moves to one of 6 points on the same edge following a distribution that is fixed for each edge and not influenced by the dog. The dog chooses one point from 6 candidate



(a) State (b) Action

Fig. 3.1: The state and action of training a dog.



(a) Step (b) Rats caught per step (c) The gap between the learned policy and the optimal policy

Fig. 3.2: The performance of different algorithms to train a dog.

points to go as the action in this state. The probability that the dog catches the rat is inverse to the distance between the dog and the rat. After observing the locations of the dog and the rat, trainers provide feedback to current action of the dog. Then, the next step begins. For example, Figure 3.1a is a state and Figure 3.1b shows the action of dog and the location the rat appears. The optimal actions for all states are generated randomly before the experiment begins. For each state, we use a standard Gaussian function per state to decide the probability that the rat appears at different points, which is $e^{-\frac{(a-\lambda(s))^2}{2}}$, $\forall a$. Normalization is executed to ensure the sum of probabilities is 1. Similarly, the probability that the rat is caught is $e^{-\frac{(a_{rat}-a_{dog})^2}{2}}$. We invited 40 trainers

to participate in this experiment. Four states are randomly ordered and appears one by one.

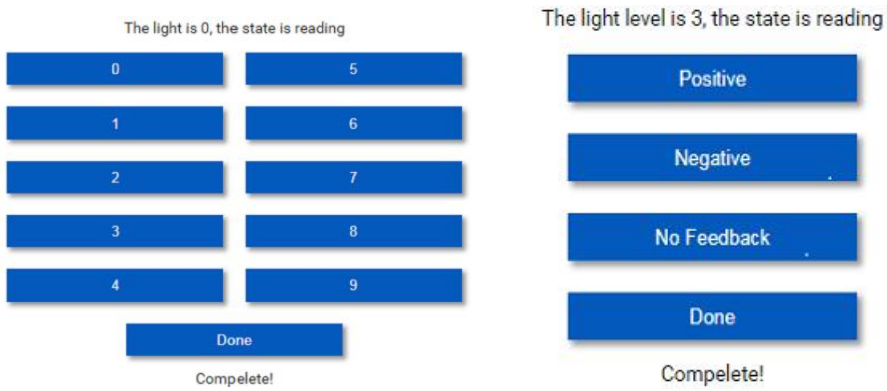
When to stop? We limit the number of steps that trainers can interact with the agent to be 15 in each state in order to constrain the time of the experiment. We will show that this limitation is reasonable since the probability that our algorithm learns the optimal actions under this constraint is high. The state changes when 1) the trainer thinks that the dog's performance is good enough or 2) the number of steps exceeds 15. The experiment ends if all the states satisfy the above mentioned conditions. Our algorithm and two baselines introduced in the last section are executed one by one with random order for each trainer. Trainers are not informed of the order.

Metrics. Three metrics are applied to judge the performance of various algorithms: 1) The number of steps used to finish the experiment. 2) The average number of caught rats over steps. 3) The 2-norm between the learned policy and the optimal policy. A good algorithm can catch more rats during training and finally learn the optimal policy with less steps.

Results. Results under these metrics are shown in Figure 3.2. Figure 3.2a shows that most of trainers finish the training before 15 steps per state. However, for other baselines, more steps are required to satisfy trainers' criteria of ending, which indicates that other algorithms perform relatively badly after training. Figure 3.2b shows the number of rats caught per step. The larger the value is, the faster an algorithm converges to the optimal policy during the training. The gap between the optimal policy and the learned policy is illustrated in Figure 3.2c. The gap is calculated by the 2-norm between these two policies. These three figures indicate that our algorithm performs the best considering the speed of convergence and the quality of the learned policy. For all the three metrics, the ABLUF method is statistically superior to others using Wilcoxon two-sided signed-rank test ($p < 0.01$).

3.4.4 Learning Users' Preference of Lighting

The second experiment aims at learning users' preference of lighting in different situations. Users provide different kinds of feedback after an action corresponding to a light level is chosen by an algorithm. Since users' utility functions are very difficult for users



(a) QUERY method. (b) Other methods.

Fig. 3.3: GUI designed for human experiment. (a) Users can change the light level directly by clicking the buttons. (b) Users can transfer their satisfaction by clicking the first three buttons.

to access and human sense is inaccurate [8, 40], human feedback is uncertain and they are likely to provide positive feedback in some sub-optimal action especially when the difference of two nearby light levels is tiny. To verify this assumption, we introduce the QUERY method as a baseline, which asks users to directly choose light levels.

We invited 40 students to participate in our experiment and built the experiment environment by a room with a computer and a YeeLight[4] bulb that is able to be controlled remotely. The experiment is divided into four parts to evaluate four different algorithms (ABLUF, UCB, ISABL, QUERY) which are arranged randomly during the experiment. In each part, participants' preferred light levels in three states (reading, watching videos and talking with others) are learned and the order of these states is fixed. The number of optional light levels is 10 ($0\%, 11\%, 22\%, \ldots, 99\%$ of the bulb's maximum lightness), which is a balance of two concerns. 1) It is difficult for human beings to detect the difference between two adjacent light levels when the number is larger. 2) A smaller number of light levels would potentially degrade the comfort of users. In the QUERY method, we directly ask users to adjust light levels rather than providing feedback.

Users can interact with our system by GUIs shown in Figure 3.3. The current light level and state are shown on the top of the web site and the 'Done' button is used to transform the current state to the next one. For QUERY method, users can directly

[4] `https://www.yeelight.com`

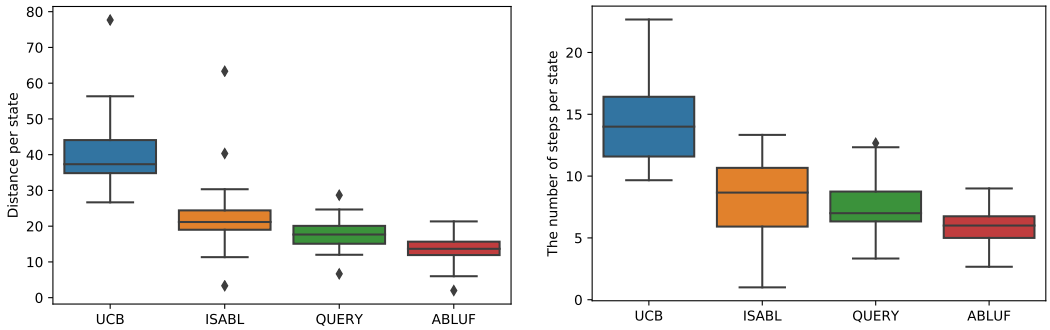
Table 3.1: The result of human experiment (*mean* ± *std. deviation*) under two metrics. The performance of our algorithm is statistically superior to others.

Algorithm	distance/state	#steps/state
UCB	40.48 ± 11.09	14.72 ± 3.49
ISABL	23.22 ± 11.82	8.21 ± 3.37
QUERY	17.75 ± 4.68	7.85 ± 2.32
ABLUF	**13.33 ± 4.59**	**5.96 ± 1.50**

click the buttons with numbers to change the light level in Figure 3.3a. A larger number corresponds to a brighter light level. For other methods, we design a similar GUI shown in Figure 3.3b. The first three buttons are designed for collecting different kinds of feedback. To collect feedback accurately, the participants can press the 'No Feedback' button to provide 'no feedback'.

When to stop? For these methods, the algorithm chooses a light level from 10 candidates according to the participant's feedback. Then, participants are asked to provide feedback. The maximum number of steps and the time for each algorithm are not constrained. After participants think that the system has learned their preferences, they are also asked to stay in the current state for 1 minute and then interact with two more steps. If the participant does not want to change the light level and the algorithm maintains his/her favorite light level when the participant interacts with the agent in the two additional steps, we think that the algorithm has converged and learned participants' preference. Then, participants can click the 'Done' button and the state will be transferred to the next one until all of the states are traversed. For the QUERY method, participants can explore and select their favorite light levels in each state without any constraint. In each state, they can choose other light levels through the GUI if they feel uncomfortable. After they think a light level is comfortable, we ask them to stay in the current lighting situation for 1 minute to ensure that their feeling is stable. If their feeling does not change, they can click 'done' button to transfer to the next state. Otherwise, they could continue to choose another light level.

Metrics. For this experiment, the step required to finish the training could be used as the metric. The other two metrics introduced in the last section is not applicable

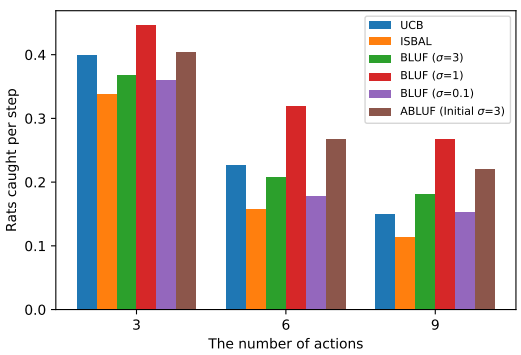


(a) Distance/state (b) #steps/state

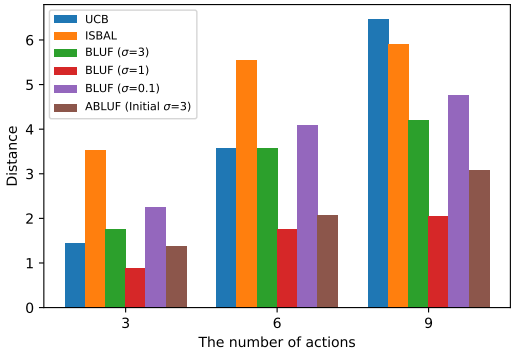
Fig. 3.4: The result of human experiment under accumulative distance and steps.

since the setting of this experiment is different. Since users only end the experiment when they feel comfortable, the learned policy is the optimal one. And it is difficult to know the probability that a user feels comfortable when the light level is not his preferred one. Alternatively, we use accumulative distance $d = \sum_{t=0}^{T}[a_t - \lambda(s)]^2$ between the current action and the optimal action as a metric to evaluate the degree of discomfort. Intuitively, if the distance is large, the user is not likely to feel comfortable. In practice, if one light level is selected in the last few steps repeatedly, we will ignore these steps since what we are concerned about is the total number of steps required for learning rather than verifying.

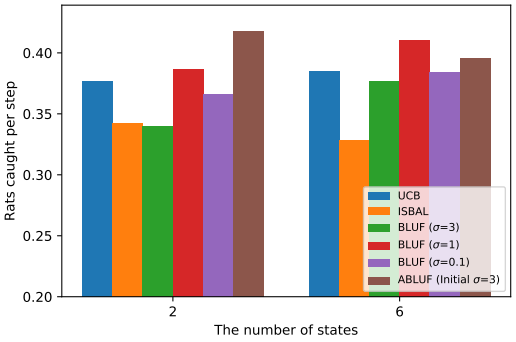
Results. The result of our human experiment is shown in Figure 3.4 and Table 3.1. The ABLUF method is statistically superior to the comparing algorithms using Wilcoxon two-sided signed-rank test ($p < 0.01$). In the QUERY method, participants cannot select their favorite light levels with a small number of steps, since they need to compare similar light levels to find their favorite ones and their strategies of finding preferred light levels are radical. For example, when the light level is too bright, users are prone to select a very dark one and then use more steps to find the optimal one. This observation is supported by [8, 21] which show that utility functions are very difficult for users to access, which indicates that they cannot tell their preferred light levels accurately without comparison. Our algorithm designs a more efficient way to explore various light levels according to participants' feedback.



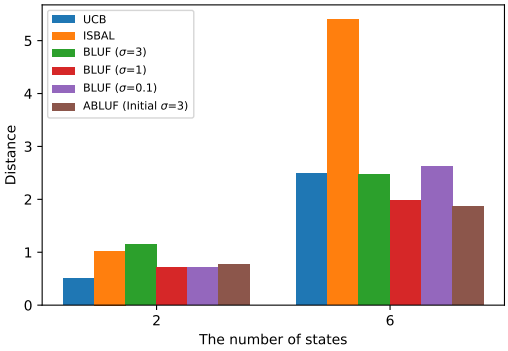
(a) Rats caught per step when $|s| = 4$



(b) The gap between policies when $|s| = 4$



(c) Rats caught per step when $|a| = 3$



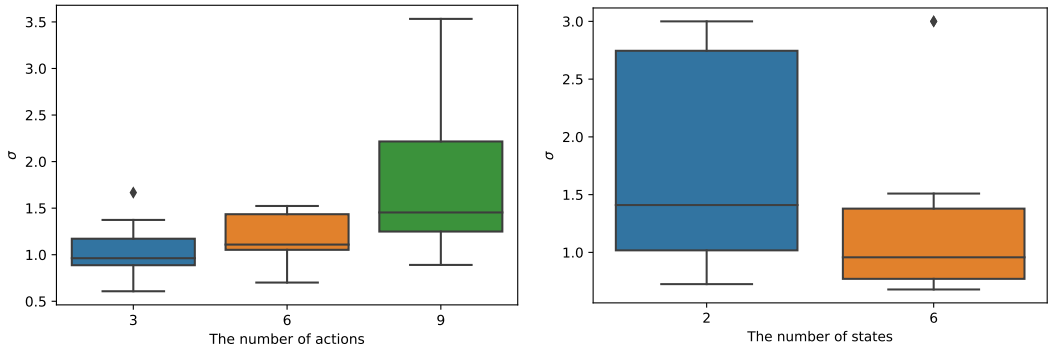
(d) The gap between policies when $|a| = 3$

Fig. 3.5: The result of dog training experiment with different numbers of actions and states.

3.4.5 Performance in Various Synthetic Environmental Settings

In this subsection, we implement experiments with simulated trainers/users in the above mentioned environments to answer three questions: 1) How does the number of actions and states influence the performance of different algorithms. 2) Can our algorithm learn the true value of σ precisely? 3) Can our algorithm perform well when trainers does not follow our feedback model? The experimental results show that the ABLUF algorithm is better than two baselines significantly ($p < 0.01$ by t-test) and can learn the value of σ accurately in most cases.

Ablation. To illustrate the importance of updating the feedback model, we also compare with Bayesian Learning for Uncertain Feedback (BLUF) algorithm, a simplified version of our ABLUF algorithm, where we treat the parameter σ as a constant input



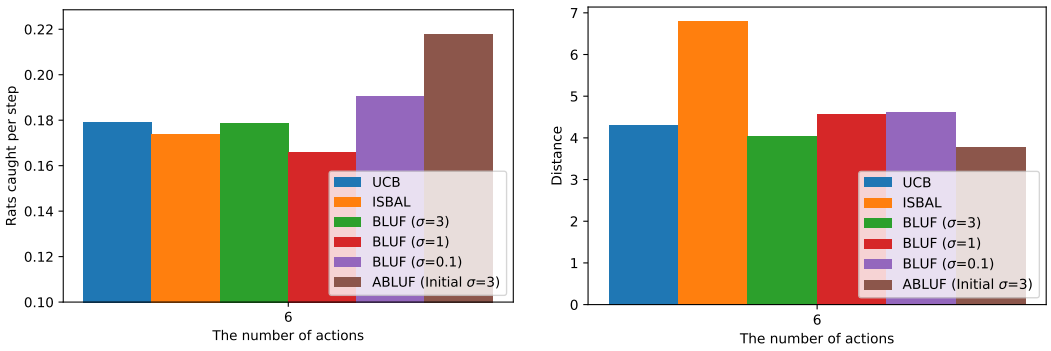
(a) Fixed #states: 4 (b) Fixed #actions: 3

Fig. 3.6: The values of σ after training.

($\sigma \in \{0.1, 1, 3\}$ in our experiments). This means that we have prior knowledge about the changed ratio of the probabilities at action a given the distance $d(a, a_s^*)$. However, it is hard to get sufficient data to obtain such prior knowledge in the real world for each trainer. Thus, the BLUF algorithm is not feasible in our human experiment. We only compare with it to illustrate the impact of an accurate σ in this simulated experiment.

Answer for the first question. For the synthetic data, the probabilities of obtaining different kinds of feedback follow our feedback model in which $\sigma = 1$ and μ is randomly generated. The optimal actions are generated randomly for various states. For both environments, we set the number of actions to 3, 6 and 9 when the number of states is 4. When the number of actions is 3, we vary the number of states to 2 and 6. Trainers are allowed to interact with algorithms for 15 steps per state, which is similar to our human experiment.

For the dog training environment, Figure 3.5 shows the performance of all the algorithms in different settings. The sub figures indicate that the change of the number of actions affects performance more significantly due to two reasons: 1) There are more sub-optimal actions when the number of actions is large, which increases the difficulty to figure out the optimal actions. 2) The worst case for two metrics becomes worse. The maximum distance of the optimal action and another action increases from $\sqrt{2}$ to $\sqrt{8}$ when the number of actions varies from 3 to 9. Similarly, the minimal probability of catching a rat declines from e^{-2} to e^{-32}. Therefore, performance of all algorithms descends. Our ABLUF method outperforms all the baselines except the BLUF algorithm



(a) Rats caught per step (b) The gap between policies

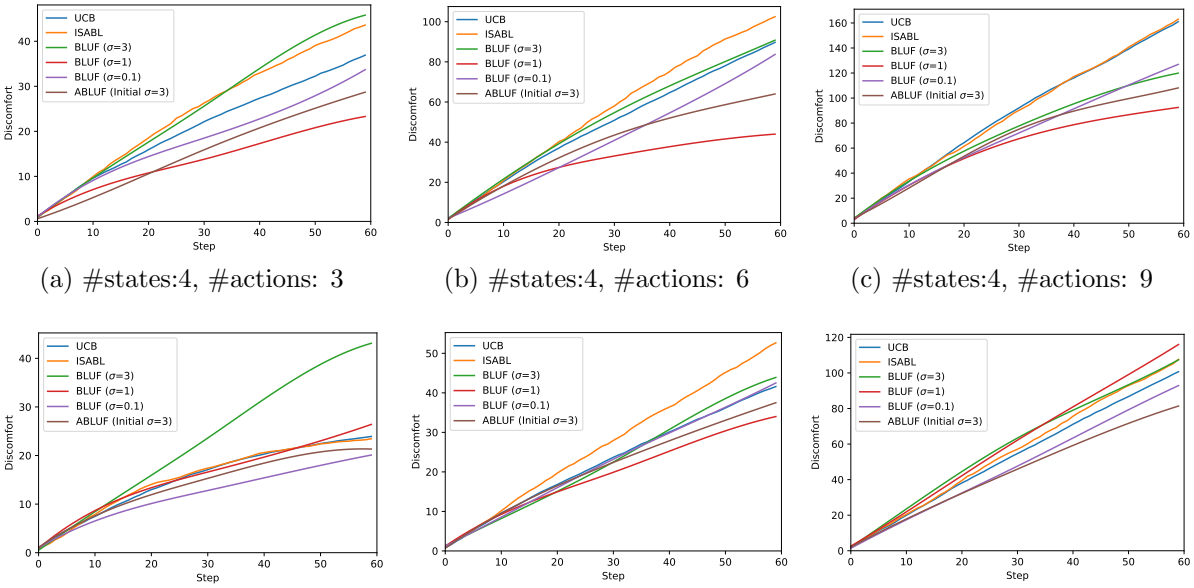
Fig. 3.7: The result of dog training experiment with randomly generated trainers when $|s| = 4$ and $|a| = 6$.

with accurate $\sigma = 1$ known in advance. For other ablation methods with biased σ, our adaptive method achieves better performance, which shows that an accurate σ is crucial.

However, the variation of states does not impact the number of the rats caught per step, since the probabilities of catching rats are similar for both the optimal action and a sub-optimal one. This conclusion is verified by the last figure, where the gap increases with respect to the number of states. The figure illustrates that the learned policy is worse when $|s| = 6$. However, the rat caught per step does not decrease, which indicates that a sub-optimal policy does not affect much in this setting. The performances of ABLUF and BLUF are similar when $|s| = 2$. The reason would be that a simple setting could be solved by an inaccurate model such as ISABL.

For the lighting control environment, similar results are displayed in Figures 3.8a to 3.8e. Since the maximum value of $[a - \lambda(s)]^2$ increases dramatically when we change the number of actions form 3 to 9. The accumulative distances of all the algorithms vary rapidly form Figure 3.8a to Figure 3.8c, while the increase is relatively small when the number of states becomes larger, which is the same as the dog training experiment.

Answer for the second question. Figure 3.6 illustrates the learning results of the parameter σ summarised from two environments. The worst case appears when $|s| = 4, |a| = 9$ and $|s| = 2, |a| = 3$. In the simplest setting, the accuracy of σ does not have a significant influence on convergence and the ABLUF algorithm figures out the optimal actions quickly without collecting enough feedback. Thus, σ cannot be learned



(a) #states:4, #actions: 3 (b) #states:4, #actions: 6 (c) #states:4, #actions: 9

(d) #states:2, #actions: 3 (e) #states:6, #actions: 3 (f) Randomly generated User (#states:4, #actions: 6)

Fig. 3.8: Accumulative distance trend with the change of the number of states and actions.

precisely. When $|s| = 4$ and $|a| = 9$, since the limitation of steps is 15 per state, it is not enough for our algorithm to converge to the optimal policy and thus the estimate of the ratio $ratio_a(f)$ is biased leading to the update towards wrong directions. This phenomenon would be eliminated with more steps and feedback, since for other settings, our algorithm can learn a relatively precise σ with little variance.

Answer for the third question (Robustness Analysis). Additionally, we evaluate our algorithm in the worst case, in which the trainer does not follow our feedback model and the probabilities of providing different kinds of feedback are generated randomly. We set $|s| = 4$ and $|a| = 6$, which is the same as the human experiment for training virtual dogs. We randomly generate simulated trainers and the only constraint is that the optimal action at each state has the highest probability of receiving positive feedback and the lowest probability of receiving negative feedback. The restriction is mild because of the definition of 'the optimal action'. Figure 3.7 and Figure 3.8f demonstrate that even for the worst case, our algorithm outperforms the other two baselines. The ABLUF method performs the best in this case where we do not have any prior knowledge about human feedback models. Since we fix the value of σ for BLUF algorithm, it performs not

well when the true value of σ differs from the prior knowledge given to BLUF. However, the ABLUF method can learn an approximated model even if these trainers do not follow our feedback model and thus is robust.

3.5 Chapter Summary and Discussion

In this chapter, we consider the uncertainty when humans provide feedback. More specifically, trainers are likely to provide positive feedback, negative feedback and no feedback to any action no matter if the action is optimal or not. To address this issue, we propose a novel feedback model that has two sets of parameters to control the shape of functions in the model. An approximated Expectation Maximization (EM) algorithm combined with Gradient Descent (GD) method is proposed to learn an optimal policy and update the feedback model simultaneously. To illustrate the performance of the novel method, we implement experiments on both synthetic scenarios and two different real-world scenarios with human participants. Experimental results indicate that our algorithm outperforms baselines under multiple metrics. Moreover, robustness analysis shows that our algorithm performs well even if trainers do not follow our feedback model.

The proposed human feedback model can be applied to recommender systems as what we show in the second experiment, which is learning preferences according to human feedback. One possible application is understanding the ratings in online e-commerce platforms. Since different users have different habits of providing ratings, human feedback models can help us to understand the real meanings of the ratings. For example, if a user always gives high ratings to whatever he purchased, the low rating given by this user to an item would indicate a more serious criticism than another user, who always gives low ratings.

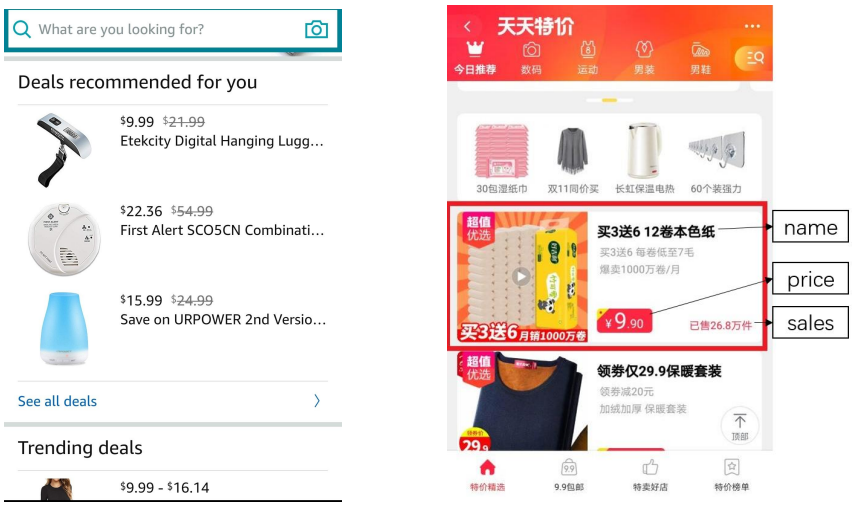
Chapter 4

Contextual User Browsing Bandits for Large-Scale Online Mobile Recommendation

4.1 Introduction

With the popularization of e-commerce platforms, a considerable proportion of users visit e-commerce platforms like Amazon and TaoBao by mobile devices. In typical recommendation scenarios of these platforms, a list of items is recommended online based on the features of items and the profiles of users. Due to the limited screen size of mobile devices, only the first few items of the list are displayed on users' mobile devices at first glance. To view the rest of the list, a user needs to slide the screen to go to the next page. In this process, the user can click an item that attracts him. After viewing the details of the item, he can return to the list to browse other items and make multiple clicks. If none of these items attracts the user in one or two pages, he is likely to quit from the recommendation scenario or the application.

In the aforementioned process, positions of items have a significant influence on clicks, since the probability of examination (a user views an item) is normally larger if the rank of an item is higher, which is called position bias. Moreover, it is possible that the user leaves the scenario without browsing all items and this phenomenon is called pseudo-exposure. For example, if we recommend a list of items to a user, he clicks the first item to view the details of it, and then returns to the recommended list to browse two commodities at position 2 and 3. The rest of commodities are not viewed. However, due



(a) (b)

Fig. 4.1: Pages in some e-commerce platforms

to the cost of data transmission and limited computation in mobile devices, it is tricky to accurately know which items are viewed. In many cases, we only know that he clicks the first one. Items behind position 3 are viewed as negative samples leading to biased rewards, since most of the existing bandit researches assume that all the recommended items are browsed by users.

This chapter [1] aims at addressing the position bias and the pseudo-exposure problem to improve the performance in the online recommendation. We adopt the contextual bandit framework, which is widely used for online recommendation. There are a few challenges that should be addressed to apply the bandit framework in our problem. First, a set of arms should be selected each round since a list of items is recommended in the online recommendation. However, traditional contextual multi-armed bandit algorithms, such as LinUCB [49], only choose one arm each round and thus cannot be applied in our problem directly. Second, the position bias and pseudo-exposure issue impact our dataset. If we directly learn from such data, clicks are biased due to the influence of positions and some items that are not viewed by users are treated as negative samples. Thus, the learned user preferences could be inaccurate. Although position bias is widely studied in recommender systems, only a few works consider it in the bandit domain. Though

[1] This chapter has been published in [24].

combinatorial bandit algorithms [37, 38, 46, 74] have been proposed to pull a set of arms in each round, existing works rarely consider the position bias issue in contextual combinatorial bandit methods. There are some attempts to model the influence of positions on clicks, such as the Cascade Model [15] and the Position-Based Model [14]. However, the Cascade Model and its extension directly assume that users will not browse the items behind the last click and ignore them, which is not accurate. The existing researches [44,48] that apply the Position-Based Model in bandit algorithms ignore the features of items, which leads to poor performance in large-scale recommendation problems.

In this chapter, we propose a novel contextual combinatorial bandit algorithm to address these challenges. Our contributions are four-fold. First, we model the online recommendation task as a novel contextual combinatorial bandit problem and define the reward of a recommended set. Second, aiming at addressing the position bias and the pseudo-exposure issue, we assume that the examination probability is determined by both the position of an item and the last click above the position. We estimate the examination probabilities of various positions based on the User Browsing Model and use them as weights of samples in a linear reward model. Third, we formally analyze the regret of our algorithm. For T rounds, K recommended arms, d-dimensional feature vectors, we prove an $\tilde{O}(d\sqrt{TK})$ regret bound. Finally, we propose an unbiased estimator to evaluate our algorithm and other baselines using real-world data. An online experiment is implemented in Taobao, one of the largest e-commerce platforms in the world. Results on two CTR metrics show that our algorithm significantly outperforms the extensions of some other contextual bandit algorithms both on the offline and online experiments.

4.2 Problem Statement and Formulation

In this section, we first provide a problem statement and describe the challenge encountered by mobile e-commerce platforms. We then formally formulate contextual combinatorial bandit problem and give the definition of reward.

With the popularity of smartphones, more and more consumers are prone to visiting e-commerce platforms by mobile applications. Different from web pages, the space of a page on mobile applications is limited and usually only a few commodities can be displayed. For example, Fig. 4.1a and 4.1b display 3 and 2 commodities respectively. Our

recommendation scenario is from Taobao, in which 12 commodities are recommended when a user enters the scenario. However, only one item can be seen at the first glance because of the limited screen size of mobile devices. Due to this, the data we obtained is affected by positions. More specifically, the probability that a user examines an item depends heavily on its position. 1) For an item, a higher position leads to a higher probability of click [36]. 2) Since we only obtain the items that are recommended and clicked after a user leaves our recommendation scenario, whether the items whose positions are behind the last click are viewed or not is unknown, which leads to the pseudo-exposure issue. An idea for addressing these problems is evaluating the influence of positions based on clicks and positions. The estimated influences are utilized as the weights of samples to learn users' preferences more accurately in this chapter.

Recommending multiple commodities to a user based on context information can be naturally modeled as a contextual combinatorial bandit problem. Formally, a contextual combinatorial bandit method M proceeds in discrete rounds $t = 1, 2, 3, \ldots, T$. In round t, M observes the current user u_t and a candidate set $\mathcal{A}_t$ including m arms and a set of context X_t that includes d-dimensional context vectors x_a for each arm $a \in \mathcal{A}_t$. The algorithm M will choose a subset S_t including K arms from $\mathcal{A}_t$ based on the observed information and then receive a reward vector R_{S_t} containing the rewards $r_{a_{k,t}} \in \{0, 1\}$, $\forall a_{k,t} \in S_t$, which is the semi-bandit feedback used in existing researches. The algorithm then updates the strategy of choosing arms based on the tuple (R_{S_t}, X_t).

Now we define the reward functions for the recommended set S_t following [38, 46, 56, 107]. To avoid the abandon of users, we hope that at least one item in the set S_t attracts users and can be clicked. In view of this, the reward of the selected set is defined to be 1, if at least one of the K items is clicked, formally,

$$F(S_t) = \begin{cases} 1 & \text{if } \sum_{r \in R_{S_t}} r > 0 \\ 0 & \text{otherwise} \end{cases}$$

In our recommendation problem, we view the commodities in the candidate set as arms. When at least one displayed commodity is clicked, the reward r is 1, otherwise 0, which is the click through rate (CTR) of a recommended set. For bandit algorithms, the objective usually is to minimize the regret, formally $R(T) = \mathbb{E}\left[\sum_{t=1}^{T} F(S_t^*) - F(S_t)\right]$, where S_t^* is the optimal set for round t. Thus, the goal of minimizing the regret can be transferred to maximize the expected CTR $\mathbb{E}[F(S_t)]$ in our problem [49].

Algorithm 5: LinUCB algorithm with User Browsing Model (UBM-LinUCB)

Input: Constant $\lambda \geq \phi'_w,\ \beta \geq \|\theta^*\|_2^2$ (We set $\lambda = \phi'_w,\ \beta = d$ in experiments), weights of different positions $\{w_{k,k'} | k = 1, \ldots, K,\ k' = 1, \ldots, k-1\}$,the number of items in a page K, the set of arms $\mathcal{A}$, the set of context X;

$A_0 = \lambda I_d$ (d-dimensional identity matrix);
$b_0 = 0_{d \times 1}$ (d-dimensional zero vector);
for $t = 1, \ldots, T$ **do**
  $\alpha = \sqrt{d \ln\left(1 + \frac{\phi'_w t}{d\lambda}\right) + 2\ln(tK)} + \sqrt{\lambda\beta}$;
  $\theta = A_t^{-1} b_t$;
  $p_a = \theta^T x_{a_t} + \alpha\sqrt{x_{a_t}^T A_t^{-1} x_{a_t}}$;
  $S_t = \varnothing$;
  for $k = 1, \ldots, K$ **do**
    $a_{k,t} = \arg\max_{a \in \mathcal{A}_t \setminus S_t} p_a$;
    $S_t = S_t \cup \{a_{k,t}\}$;
  Display S_t to a user;
  Get reward vector $R_{S_t} = [r_{a_{t,1}}, \ldots, r_{a_{K,t}}]$ of S_t;
  Compute k' for all positions based on R_{S_t};
  $A_t = A_{t-1} + \sum_{k=1}^{K} w_{k,k'}^2 x_{a_{k,t}} x_{a_{k,t}}^T$;
  $b_t = b_{t-1} + \sum_{k=1}^{K} w_{k,k'} r_{a_{k,t}} x_{a_{k,t}}$;

4.3 LinUCB with User Browsing Model

In this section, we propose our contextual bandit algorithm UBM-LinUCB (LinUCB with User Browsing Model), illustrated in Algorithm 5, which addresses the position bias and pseudo-exposure issue and is able to recommend a list of commodities with the theoretical guarantee.

After users view some items shown in the screen, they decide to slide the screen or leave, if these items are not attractive. Intuitively, the probability that users leave increases after seeing a long sequence of unattractive items. Thus, the probability of examination decreases when the position of an item becomes lower [36]. However, if an item is clicked, this item must be displayed in the screen. Since the screen usually can display more than one items, the nearby item (behind the clicked one) are more likely to be displayed and the probability of examining increases. Therefore, we introduce position weights relating to both positions and clicks to address the position bias and

the pseudo-exposure issue. For all the items, we estimate the probability of examination rather than treating them as negative samples or ignoring them directly. Inspired by the User Browsing Model (UBM) [14], we assume that the click through rate $r_{t,a}$ of an arm is determined by the examination probability w and attractiveness $\gamma(a)$ of arm a, namely, $r_{t,a} = w_{k,k'}\gamma(a)$, where $\gamma(a)$ is the attractiveness of an item a and $w_{k,k'}$ is the examination probability meaning the probability that a user views an item. We assume that $w_{k,k'}$ depends not only on the rank of an item k, but also on the rank of the previously clicked item k'. By assuming that the attractiveness of items follows a linear function, we propose a new linear reward model:

$$\mathbb{E}[r_{a_k}] = \theta^T(w_{k,k'}x_{a_k}) \tag{4.1}$$

where θ is an unknown coefficient vector whose length is the same as x_{a_k}. $w_{k,k'}$ is the examination probability for the rank k when the position of the last click is k'. We assume that $w_{k,k'}$ is a fixed constant, since it only depends on the structure of a page and can be learned in advance [48]. We introduce how to obtain $w_{k,k'}$ in the experiment part and focus on our main contribution, bandit algorithm, in this section.

We use ridge regression to solve the linear model [49]. The objective of ridge regression is to minimize the penalized residual sum of squares (PRSS):

$$PRSS(\theta) = \sum_{t=1}^{T}\sum_{k=K}^{T}[r_{a_{k,t}} - \theta^T(w_{k,k'}x_{a_{k,t}})]^2 + \sum_{j=1}^{d}{\theta_j}^2 \tag{4.2}$$

where θ_j is the j-th element of θ. To simplify notation, we use $w_{k,k'}$ to denote $w_{k,k',a_{k,t}}$, which is the examination probability of the item $a_{k,t}$ at round t where the position of $a_{k,t}$ and the last click before $a_{k,t}$ is k and k' respectively.

Let X be a $TK \times d$-dimensional matrix whose rows correspond to the context $x_{a_{k,t}}$ of the arm $a_{k,t}$ in round t and position k. W is also a $TK \times d$-dimensional matrix whose rows are $w_{k,k'}\mathbb{1}_{1\times d}$, weights of $a_{k,t}$ in round t and position-pair (k, k'). R contains rewards $r_{a_{k,t}}$ for the item $a_{k,t}$ each round t and position k. The PRSS function can be transformed to a compact representation:

$$PRSS(\theta) = [R - \theta^T(W \circ X)]^T[R - \theta^T(W \circ X)] + \lambda\|\theta\|_2^2 \tag{4.3}$$

where λ is a constant to control the weight of the regularizer and $\circ$ is the Hadamard product and $(W \circ X)_{i,j} = W_{i,j}X_{i,j}$. Thus, each row of $W \circ X$ is $w_{k,k'}x_{a_{k,t}}$.

To minimize $PRSS(\theta)$, the derivation $\frac{\partial PRSS(\theta)}{\partial \theta}$ should be zero. Then, the solution of θ is:

$$\theta = [(W \circ X)^T (W \circ X) + \lambda I_d]^{-1} (W \circ X)^T R \tag{4.4}$$

where I_d is the $d \times d$ identity matrix. Let $A = (W \circ X)^T (W \circ X) + \lambda I_d$ and $b = (W \circ X)^T R$. Applying the online version of ridge regression [49], the update formulations of A_t and b_t in round t are shown as follows:

$$A_{t+1} = A_t + \sum_{k=1}^{K} w_{k,k'}^2 x_{a_{k,t}} x_{a_{k,t}}^T \tag{4.5}$$

$$b_{t+1} = b_t + \sum_{k=1}^{K} w_{k,k'} x_{a_{k,t}}^T r_{a_{k,t}} \tag{4.6}$$

The initialization of A and b is shown in Line 1 and 2. Then, we use Eq. (4.5) and (4.6) in Line 14 and 15 respectively to update these two coefficients in our algorithm.

The standard deviation of the ridge regression [91] for any a and a given pair $\{k, k'\}$ is

$$\sqrt{w_{k,k'} x_a^T A_t^{-1} w_{k,k'} x_a}$$

and the upper confidence bound used to select the best recommended set is

$$\begin{aligned} p_a &= \theta^T w_{k,k'} x_a + \alpha \sqrt{w_{k,k'} x_a^T A_t^{-1} w_{k,k'} x_a} \\ &= w_{k,k'} (\theta^T x_a + \alpha \sqrt{x_a^T A_t^{-1} x_a}) \end{aligned} \tag{4.7}$$

where α is a parameter related to t, which is defined in the next section. Since the parameter $w_{k,k'}$ for a fixed pair $\{k, k'\}$ is a constant and is not related to a, we can ignore it and use the simplified equation in Line 6:

$$p_a = \theta^T x_a + \alpha \sqrt{x_a^T A_t^{-1} x_a} \tag{4.8}$$

In the next section, Lemma 4.1 indicates how to select the optimal set based on p_a and Theorem 4.1 defines α and constants used in the algorithm.

4.4 Theoretical Analysis

In this section, we give the theoretical analysis and prove that UBM-LinUCB achieves the regret bound $\tilde{O}(d\sqrt{TK})$ with respect to the aforementioned formulation of reward,

where the $\tilde{O}$ notation hides logarithmic factors. **Our key contributions are 1) the proof for Lemma 4.1, and 2) considering $w_{k,k'}$, which depends on both position k and the position of the last click k' in Theorem 4.1.** We define the expected cumulative regret at round T formally:

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T} F(S_t^*) - F(S_t)\right] \tag{4.9}$$

where S_t^* is the optimal set.

We first show that with respect to our novel linear reward model, the optimal set S_t^* at each round t selects the arms with top-K values of $x_{a_t}^T \theta^*$, which holds for R based on the rearrangement inequality. Then, the upper bound is proved.

Let $\gamma(a) = x_a^T \theta^*$. We assume that, without loss of generality,

$$w_{j+1,j} \geq w_{j+2,j} \geq \cdots \geq w_{K,j}$$

and

$$w_{k,k-1} \geq w_{k,k-2} \geq \cdots \geq w_{k,0}.$$

These two assumptions can be explained intuitively: 1) If a user clicks the j-th position, the probability that he observes k-th ($k > j$) position is inversely related to the distance $k - j$. 2) For a fixed position k, the probability of examination is larger when the position of the last click k' is closer to k.

The following lemma verifies that the optimal set S_t^* simply selects the arm a with the k-th highest value $\gamma(a)$ at the k-th slot for F by the rearrangement inequality. Let $S_t^* = \{a_{1,t}^*, ..., a_{K,t}^*\}$ where $a_{k,t}^*$ is the optimal arm recommended at the k-th position. We have:

Lemma 4.1 *S_t^* maximizing $\mathbb{E}[F(S_t)]$ consists of $a_{k,t}^*$ being the arm with the k-th highest value $\gamma(a)$.*

Now we transform the upper bound of R and give the proof.

Lemma 4.2 ([38]) *The upper bound of R is*

$$\sum_{t=1}^{T} \sum_{k=1}^{K} w_{k,0} \left[\gamma(a_{k,t}^*) - \gamma(a_{k,t})\right]$$

The proofs of two Lemmas are in the Appendix. Finally, we prove the upper bound of $R(T)$ for UBM-LinUCB algorithm.

Theorem 4.1 *Let* $\phi'_w = \sum_{k=1}^{K} w_{k,k-1}^2$. *When* $\lambda \geq \phi'_w \geq 1$, $\|\theta^*\|_2^2 \leq \beta$ *and*

$$\alpha \geq \sqrt{d \ln\left(1 + \frac{\phi'_w T}{d\lambda}\right) + 2\ln(TK)} + \sqrt{\lambda\beta},$$

if we run UBM-LinUCB algorithm, then

$$R(T) \leq 2\alpha\sqrt{2TKd \ln\left(1 + \frac{\phi'_w T}{\lambda d}\right)} + 1$$

Proof. (sketch) The structure of proof follows [1, 97]. The main contribution is considering $w_{k,k'}$, which depends on both position k and the position of the last click k'. We first define an event to judge the distance between the true attractiveness and the estimated attractiveness of each item $a \in A_t$:

$$E = \{|\langle x_{a_{k,t-1}}, \theta^* - \theta_{t-1}\rangle| \leq \alpha\sqrt{x_{a_{k,t-1}}^T A_{t-1}^{-1} x_{a_{k,t-1}}}, \forall a_{k,t} \in A_t, \forall t \leq T, \forall k \leq K\}$$

If event E happens, the regret can be bounded by the variance

$$\begin{aligned} R(T) &\leq 2\alpha\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,0}\sqrt{x_{a_{k,t}}^T A_{t-1}^{-1} x_{a_{k,t}}}\right] \\ &\leq 2\alpha\sqrt{2TKd \ln\left(1 + \frac{\phi'_w T}{\lambda d}\right)} \end{aligned}$$

Then, we prove that the E happens with the probability $1 - \delta$ when α satisfies the condition shown in the Theorem. And the bound of regret is $TK\delta$ when E does not happen. Let $\delta = \frac{1}{TK}$ and combining these two parts together, we finish the proof.

The full version of the proof is in the Appendix. We prove that our algorithm can achieve $\tilde{O}(d\sqrt{TK})$ bound and the $\tilde{O}$ notation hides logarithmic factors. Our bound is improved compared to related works [56, 107], which proved $\tilde{O}(dK\sqrt{TK})$ and $\tilde{O}(dK\sqrt{T})$ bounds respectively with the same reward function.

4.5 Experimental Evaluation

In this section, we describe the details of our experiments. First, we define the metrics of performance and the evaluation method. An unbiased estimator is introduced. Second, benchmark algorithms are listed and introduced briefly. Then, we introduce a public dataset, Yandex Personalized Web Search dataset[2], used in a simulated experiment and a real-world dataset provided by Taobao. We also implement an online experiment in the e-commerce platform. The details of data collection and processing are presented. Finally, the results of both offline and **online** experiments are provided and analyzed.

4.5.1 Metrics and Evaluation Method

Corresponding to the formulation of reward, we use the CTR_{set}, the expectation of $F(S_t)$, as the metric:

$$CTR_{set} = \frac{\sum_{t=1}^{T} F(S_t)}{T}$$

Additional Metric. In practice, the CTR of all the recommended items is also widely used. Thus, we define the accumulative reward CTR_{sum} of the set S_t, which is the expected total clicks of S_t:

$$CTR_{sum} = \frac{\sum_{t=1}^{T} \sum_{r \in R_{S_t}} r}{T}$$

Offline unbiased estimator. Since the historical logged data is generated by a logging production policy, we propose an unbiased offline estimator, User Browsing Inverse Propensity Scoring (UBM-IPS) estimator, to evaluate the performance inspired by Li et al. [50]. The idea is to estimate users' clicks on various positions based on UBM model and the detail of reduction is in the Appendix. The formulation of the UBM-IPS estimator is shown as follows.

$$V_{UBM}(\Phi) = \mathbb{E}_X \left[\mathbb{E}_{\substack{S \sim \pi(\cdot|X) \\ r \sim D(\cdot|X)}} \left[\sum_{k=1}^{K} \sum_{k'=0}^{k} r(a_k, k, k'|X) \cdot \frac{\langle \tilde{W}, \Phi(a_k, \cdot, \cdot|X) \rangle}{\langle \tilde{W}, \pi(a_k, \cdot, \cdot|X) \rangle} \right] \right] \tag{4.10}$$

where D is the logged dataset, $a_k \in \mathcal{A}$, π is the policy that generates D. Φ is the estimated policy, $\tilde{W} = [w_{1,0}, w_{2,0}, \ldots, w_{K,K-1}]$ is a vector including position weights.

[2] https://www.kaggle.com/c/yandex-personalized-web-search-challenge

Given the features of a candidate set and a user X, $\pi(a_k, \cdot, \cdot|X)$ consists of the probabilities that a_k appears at different k and k' under π corresponding to $\tilde{W}$ which is

$$[\pi(a_k, 1, 0|X), \pi(a_k, 2, 0|X), \ldots, \pi(a_k, K, K-1|X)].$$

$\langle \cdot, \cdot \rangle$ means the dot product of two vectors.

The estimator is unbiased when 1) the term $\frac{\langle \tilde{W}, \Phi(a_k, \cdot, \cdot|X) \rangle}{\langle \tilde{W}, \pi(a_k, \cdot, \cdot|X) \rangle}$ is not infinite and 2) users' behaviors follow the UBM model. For 1), in our offline experiment, we treat commodities recommended by π in one record as candidate set. Thus, all the items chosen by Φ have been selected by π for a specific X, that is $\Phi(a_k, \cdot, \cdot|X) \neq 0_{K(K+1)/2}$ and $\pi(a_k, \cdot, \cdot|X) \neq 0_{K(K+1)/2}$. Thus, the UBM-IPS term is not infinite. For 2), we use the evaluation method proposed in [10] to empirically estimate the accuracy of the UBM model in our recommendation dataset. This evaluation method removes the position bias by using the data in the first position as the test set and the rest as the training set. The experiment result in our e-commerce dataset shows that the UBM-IPS estimator is better than traditional IPS estimator that does not involve the UBM model (MSE: UBM-IPS: 0.1754, IPS: 0.1943), although the MSE is not 0 (totally unbiased). Thus, we use UBM-IPS to evaluate different algorithms empirically in the offline experiments.

Since k' depends on previous clicks, $\Phi(a_k, \cdot, \cdot|X)$ cannot be obtained directly. In practice, we first generate a recommended set by Φ. Since Φ is deterministic in our case, the recommended set determines $\Phi(a_k, \cdot, \cdot|X)$. Then, we obtain k' sequentially. More specifically, when the policy Φ chooses the set S_Φ after sampling a context set X from dataset, we first collect $|D(\cdot|X)|$ records $D(\cdot|X)$ with the same context set X. Then, for the first commodity $a_1 \in S_\Phi$, delete the terms that do not include a_1 in the Eq. (4.10):

$$\frac{1}{|D(\cdot|X)|} \sum_{a_1 \in D(\cdot|X)} r(a_1, k, k'|X) \frac{\langle \tilde{W}, \Phi(a_1, \cdot, \cdot|X) \rangle}{\langle \tilde{W}, \pi(a_1, \cdot, \cdot|X) \rangle} \tag{4.11}$$

This equation is used to simulate the reward $r(a_1, 1, 0)$, where k and k' are the position of a_1 and the last click before k in records respectively. Since Φ is deterministic,

$$\Phi(a_1, \cdot, \cdot|X) = \begin{cases} 1 & for\ k = 1, k' = 0 \\ 0 & others \end{cases}.$$

Notice that the simulated reward would be larger than one since $\langle \tilde{W}, \Phi(a_1, \cdot, \cdot|X) \rangle$ would be larger than $\langle \tilde{W}, \pi(a_1, \cdot, \cdot|X) \rangle$. Then, we obtain the k' for the second commodity a_2

based on reward $r(a_1, 1, 0)$:

$$k' = \begin{cases} 1 & r(a_1, 1, 0) \geq 1 \ or \ \mathcal{B}(r(a_1, 1, 0)) = 1 \\ 0 & \mathcal{B}(r(a_1, 1, 0)) = 0 \end{cases}$$

where $\mathcal{B}(\cdot)$ is the reward sampled from the Bernoulli distribution with mean $r(a_1, 1, 0)$. Given a fixed k', we can use the above method to compute the reward of a_2. Repeating this process, the rewards of all commodities in set S_Φ can be obtained. Then, the CTR_{sum} and CTR_{set} can be calculated. For the CTR_{sum}, we directly use the average of the unbiased rewards over T. For the CTR_{set}, we use the the Bernoulli distribution to sample a reward (0 or 1) if $r \in [0, 1]$. If the reward of a recommended item is not less than 1, $F(S_t) = 1$.

Influence of Positions. The influence of position is mainly determined by the structure of a page. Since a page's structure does not change in the long term, the position influence $w_{k,k'}$ can be considered as constants following [48]. We use the EM algorithm suggested by [14, 16] to estimate $w_{k,k'}$ in advance. The max and min values are 0.55 and 0.17 respectively for our recommendation scenario. For the Yandex dataset, they are 0.96 and 0.015. The difference is caused by the distribution of clicks in these two datasets. Since the Yandex dataset is collected from a web search scenario, 79.3% of users click the first web page shown in the result, which is usually the most relevant to the keyword they queried. Clicks are very rare for low-ranking web pages. Therefore, the gap between the max and min position influences is more significant. However, for our recommendation scenario, users usually are not purposeful and prone to browse more items (see Fig. 4.3). Thus, the gap of position influences is relatively small.

4.5.2 Benchmark Algorithms

We evaluate 4 baseline methods to illustrate the performance of our proposed algorithm. We only compare with combinatorial bandit algorithms that are based on LinUCB considering fairness and the problem setting. The first one is a context-free combinatorial bandit algorithm merged with the PBM model. The other three algorithms are contextual combinatorial bandit algorithms. Theoretical optimal parameters are used in experiments.

- **PBM-UCB** [48]: The PBM-UCB algorithm combines the PBM model with the UCB algorithm. Positions bias is considered to compute the mean and variance

of an arm's reward by applying the position-based model (PBM), which assumes that the position bias is only related to the rank of an item. The position bias is obtained by the EM algorithm mentioned in [48]. The upper confidence bound of this algorithm is used to choose the best arm in round t:

$$PBMUCB_a(t) = \frac{R_{sum}(t)}{\tilde{N_a}^2(t)} + \sqrt{\frac{N_a(t)}{\tilde{N_a}(t)}}\sqrt{\frac{\delta}{2\tilde{N_a}(t)}}$$

where $R_{sum}(t)$ is accumulated reward obtained by the item a, $N_a(t) = \sum \mathbb{1}\{a \text{ is selected}\}$ and $\tilde{N_a}(t) = \sum w_{k_a} \mathbb{1}\{a \text{ is selected}\}$.

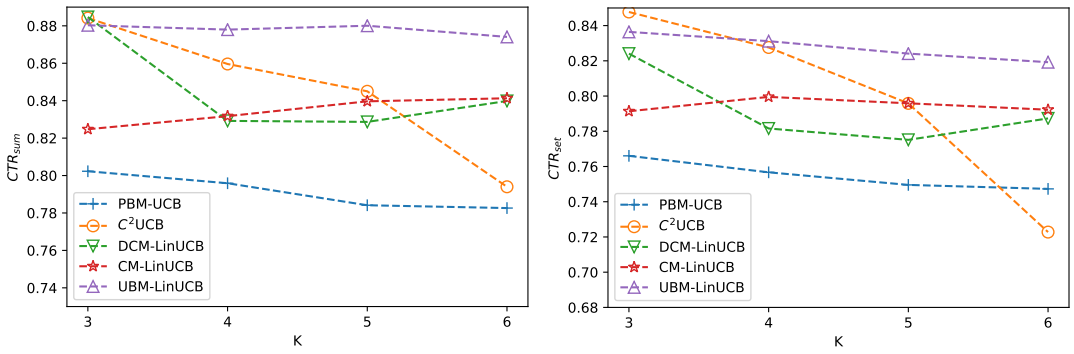
- **C^{2}UCB** [72]: C^{2}UCB algorithm is a combinatorial extension of the LinUCB algorithm. In each round, the algorithm selects K items with top-K upper confidence bounds. After displaying them to a user, the algorithm can observe the rewards for these K items and execute the update equations for each of these items.

- **CM-LinUCB**: CM-LinUCB is a simple extension of [51, 107] based on the Cascading Model. This model assumes that a user scans items from top to bottom until they find a relevant item. After they click the relevant item, they will leave without viewing other items behind. Thus, we ignore items behind the first click and only use the rest of samples to update parameters in this extension, which is the same as the First-Click algorithm used in [38].

- **DCM-LinUCB** [56]: DCM-LinUCB is an contextual bandit algorithm based on Dependent Click Model. DCM is an extension of CM by assuming that after a user clicked an item, they may still continue to examine other items. The model introduces a satisfaction variable to determine whether the user will continue to view other items or not. If the satisfaction variable equals 1, the user will leave. Otherwise, he will continue to scan other items.

4.5.3 Web Search Recommendation

The Yandex Personalized Web Search dataset contains 35 million search sessions. Each session contains a user ID, a query ID, IDs of web pages shown as result, and clicks. The



(a) CTR_{sum} (b) CTR_{set}

Fig. 4.2: The performances of algorithms when K changes.

top 3 most frequent queries are used for evaluation. The position influences are estimated by the PyClick library [14]. Since the web pages in the dataset do not have corresponding features, we build features for all the web pages. We first construct a user-website matrix $M_{u\times m}$ based on the records in the dataset, where $u = 194749$ and $m = 2701$ is the number of users and web pages respectively. The value of the element $M(i, j)$ is determined by the attractiveness of a web page j for a user i estimated by a part of Eq. 4.11:

$$\frac{1}{|D(\cdot|X)|}\sum\nolimits_{a_j\in D(\cdot|X)\rangle} r(a_j, k, k'|X)\cdot\frac{1}{\langle\tilde{W}, \pi(a_j, \cdot, \cdot|X)\rangle} \tag{4.12}$$

where $X = i$ represents the user i. Thus, the simulated reward can be obtained by multiplying the position weight $\langle\tilde{W}, \Phi(a_1, \cdot, \cdot|X)\rangle$ given any policy Φ. The features are generated by the truncated randomized SVD provided by Scikit-Learn library[3], i.e.,

$$M \approx USV^T.$$

We set the dimension of S to 10 and use the vector $[U(i), V(j)]$ as the feature of web page j for the i-th user. Notice that $|S| = 10$ is a balance between accuracy and complexity. Since the truncated randomized SVD is an approximated decomposition method, a smaller $|S|$ leads to worse accuracy. For example, all the contextual bandit algorithms cannot perform normally when $|S| = 5$. And a larger $|S|$ increases the computational complexity since the inverse of A_k need $O(d^3)$ time in each round, where d is the dimension of the feature vector.

[3] `https://scikit-learn.org/stable/`

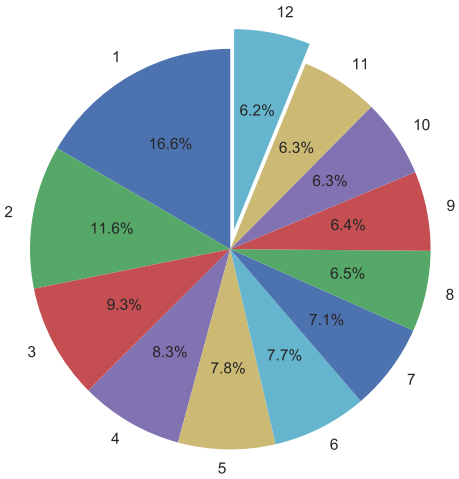
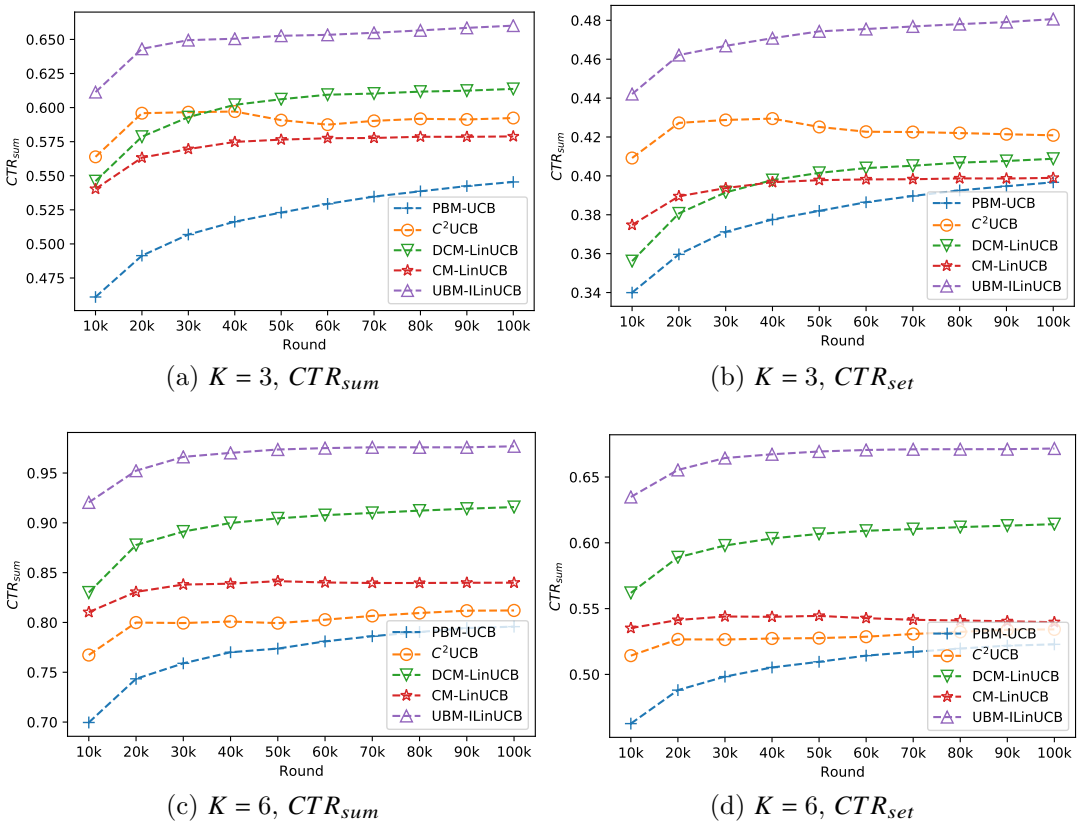


Fig. 4.3: The distribution of the last clicks' positions. The numbers outside the pie are positions of the last clicks. For example, 1 means that the last clicks of users are at the first position.

For each round, a user is randomly chosen to decide the attractiveness of different web pages. We only use the web pages that are shown to the user as the candidate set, since we cannot know the attractiveness of a web page if it is never displayed to the user. The simulated reward is estimated by the estimator we mentioned above using Eq. 4.11.

Result

The result after 5000 rounds is shown in Fig. 4.2 (average of 10 runs). When $K = 3$, contextual bandit algorithms perform similarly except CM-LinUCB. The reasons are: 1) The influence of positions is relatively small when $K = 3$. Thus, the effectiveness of different models cannot be revealed well. 2) Due to the hypothesis of the Cascading Model, CM-LinUCB only uses samples not behind the first click to update, which leads to insufficient training. When K becomes larger, the performance of C^{2}UCB declines dramatically and other contextual bandit algorithms outperform it, since C^{2}UCB does not consider the impact of positions. When K is larger than 4, the CTRs change slightly and do not increase anymore, since 91.6% of users click one web and 98.4% of users clicks 2 web pages or less. Moreover, the ratio of positive samples descends rapidly and has a negative impact on the learning results, especially for the method not involving click models.



(a) $K = 3$, CTR_{sum} (b) $K = 3$, CTR_{set}

(c) $K = 6$, CTR_{sum} (d) $K = 6$, CTR_{set}

Fig. 4.4: Performance of each algorithm under two CTR metrics with the increase of the number of rounds.

4.5.4 E-commerce Recommendation

Our real-world dataset is provided by Taobao. We utilize the data from a recommendation scenario in the platform, where 12 items are selected and then displayed to users. There are two special characteristics of this scenario. First, the number of commodities is fixed to 12. The mobile app of this platform will not recommend new items after all of these 12 items are browsed. Another characteristic is that users can only view the first item and part of the second item when they enter this scenario. To observe the rest of the items, users need to slide the screen.

We collect data following the method of the KDD Cup 2012 track 2 dataset[4], which contains logs of a search engine and is used in researches about multiple-play bandit problem [44, 48]. The items in our dataset correspond to the Ads in the KDD Cup 2012

[4] `https://www.kaggle.com/c/kddcup2012-track2`

track 2 dataset. The description of Ads is replaced by the feature vector of items. We directly use the logs of the e-commerce platform, which is the same as the KDD Cup 2012 track 2 dataset. For each user, the recommender system of the e-commerce platform selected 12 items from about 900 candidates based on both items' and users' features. Compared with the KDD Cup 2012 track 2 dataset, our dataset can be directly utilized in experiments without any transformation.

Since we cannot know a user's real interest if he does not click any item, we remove the data without clicks from the original log data. After processing, our dataset contains 180k records, while each record contains a user ID, IDs of 12 commodities, context vectors of 12 commodities and a click vector corresponding to the 12 commodities. The context vector is a 56×1 array including some features of both a commodity and a user, such as the commodity's price and the user's purchasing power. In our offline experiment, we treat 12 commodities in each record as the candidate set since whether users will click other commodities is unknown.

To illustrate the position bias and the pseudo-exposure issue better, Fig. 4.3 demonstrates the distribution of the last clicks' positions. The number around the circle is the position of the last click. Only 6.2% of the last items are clicked by users in our data. If we assume that the items at the positions behind the last click are affected by the pseudo-exposure, $\frac{11}{12}$ of items are biased for the records whose last click's position is 1, which accounts for 16.6% of the data. By calculating the weighted sum

$$\frac{11}{12} \times 16.6\% + \frac{10}{12} \times 11.6\% + \dots$$

, about 54% of rewards are affected by the pseudo-exposure issue. Notice that this percentage is a rough estimate, since it is possible that some items at positions behind the last click are viewed by users.

Feature Reduction

Additionally, we reduce the dimension of features inspired by [49], since the time of computation is closely related to the dimension of context. The update equation of A_k takes $O(d^2)$ time and the inverse of A_k need $O(d^3)$ time in each round. Thus, in order to improve the scalability of the algorithm, we apply a pre-trained denoising

Table 4.1: CTRs with the change of K from 3 to 6 when α is optimal. The numbers with a percentage is the CTR lift.

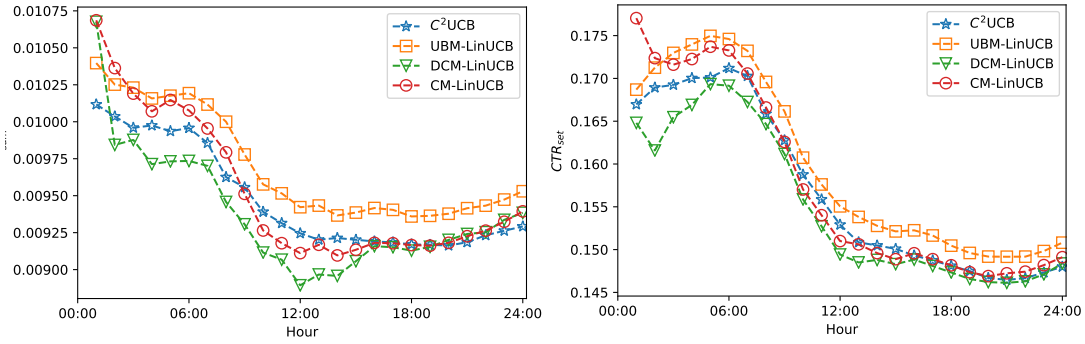
Algorithm	$K = 3$		$K = 4$		$K = 5$		$K = 6$	
	CTR_{sum}	CTR_{set}	CTR_{sum}	CTR_{set}	CTR_{sum}	CTR_{set}	CTR_{sum}	CTR_{set}
C^2UCB	0.592 0%	0.420 0%	0.643 0%	0.440 0%	0.747 0%	0.510 0%	0.812 0%	0.534 0%
PBM-UCB	0.545 -7.9%	0.396 -5.7%	0.655 1.8%	0.453 2.9%	0.744 -0.4%	0.519 1.7%	0.795 -2.1%	0.522 -2.1%
CM-LinUCB	0.578 -2.3%	0.398 -5.2%	0.674 4.8%	0.464 5.4%	0.771 3.2%	0.548 7.4%	0.839 3.3%	0.539 9.3%
DCM-LinUCB	0.613 3.5%	0.408 -2.8%	0.704 9.5%	0.474 7.7%	0.820 9.7%	0.570 11.7%	0.915 12.7%	0.614 14.9%
UBM-LinUCB	**0.660** **11.4%**	**0.480** **14.2%**	**0.781** **21.4%**	**0.538** **22.2%**	**0.869** **16.3%**	**0.604** **18.4%**	**0.976** **20.2%**	**0.671** **25.6%**

autoencoder [90] to reduce the dimension of the feature space from 56 to 10. The encoder contains 4 hidden layers with 25,10,10,25 neural units respectively and an output layer with 56 units. The ReLU activation function is applied to the outputs of the hidden layers. The context vectors x are perturbed by the vectors randomly drawn from a uniform distribution between 0 and 1, formally, $input = 0.95 * x + 0.05 * noise$. The objective of this autoencoder is to minimize the MSE loss $\|x - output\|_2^2$ between the context and the output. We adopt the output of the second hidden layer as extracted features, whose dimension is 10. The extracted features are utilized as the context in our algorithm and other compared methods.

Result

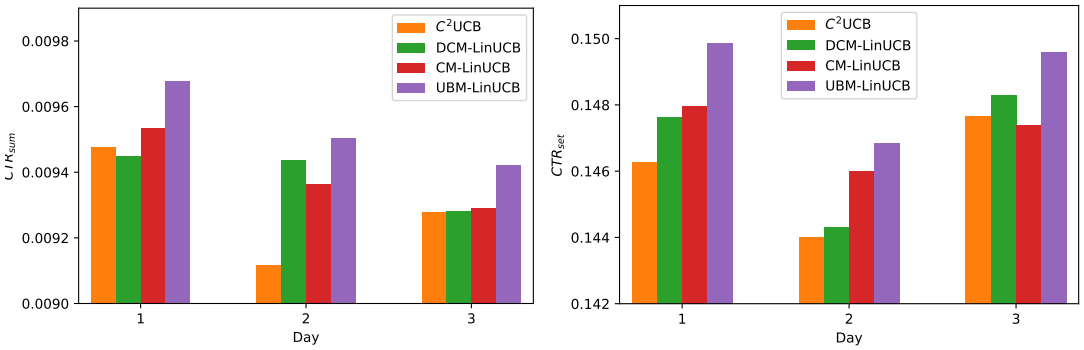
We conduct offline and online experiments to evaluate various algorithms. The first part of experiments illustrates the curves of algorithms with the increase of the number of rounds when $K = 3$ and 6. The second set of experiments shows the performance of different algorithms with the change in the number of recommended commodities. Finally, we implement the online experiment for three days in a real platform and report the performance in different days. The UBM-LinUCB algorithm performs the best in most cases.

Performance with the increase of the number of rounds. Fig. 4.4 demonstrates the performance of our algorithm under two metrics. Sub-figures 4.4a and 4.4c



(a) CTR_{sum} (b) CTR_{set}

Fig. 4.5: Trends in two metrics in the online experiment.



(a) CTR_{sum} (b) CTR_{set}

Fig. 4.6: CTRs of 3 days in the online experiment.

are the trends under the CTR_{sum} metric when $K = 3$ and 6 respectively. The other two sub-figures show the performance under the CTR_{set} metric. The UBM-LinUCB algorithm performs the best in all of these figures. Since the PBM-UCB algorithm does not consider the features of items, it performs worse than other methods. The speed of convergence is similar for the contextual bandit algorithms, while PBM-UCB is relatively slow. This is coincident to theoretical analysis that the regret of contextual bandit algorithms is independent with the number of items.

When $K = 6$, the speed of convergence is faster for almost all the algorithms. For example, for our UBM-LinUCB algorithm, it takes about 30k rounds to converge when $K = 6$ while its performance increases slowly until 60k or 70k rounds when $K = 3$. The reason would be related to the number of samples obtained in each round. More specifically, $K = 6$ means that our algorithm can receive 6 samples per round, which is

double compared to the case when $K = 3$. The more samples received in each round, the faster the speed of convergence is. One exception would be the CM-LinUCB. Since it only considers the samples before the first click, not all the samples are used for the update. Thus, the learning rate of CM-LinUCB would be insensitive to the increase of K.

The reason is that the sub-optimal policy can perform similarly with the optimal one when K is large. For example, if the recommendation policy is sub-optimal and the best response is ranked at position 4 or 5, this item can also be displayed when $K = 6$. This sub-optimal policy will receive the same reward as the optimal one. Then, the objectives of algorithms turn to learn a sub-optimal policy, which is easier than obtaining an optimal policy. Therefore, benchmark algorithms perform better when K is larger and the performance of our algorithm is slightly affected by K.

Experiments for the Number of Recommended Items K. We run algorithms to recommend various number of items and summarize the result in Table 4.1. The C^2UCB method is considered as a baseline in the table and the number with a percentage is the CTR lift compared to C^2UCB. A larger CTR of an algorithm indicates its effective use of data and features. We alter the value of K to demonstrate the influence of the recommended set's size. The UBM-LinUCB algorithm improves two CTR metrics by 20.2% and 25.6% when $K = 6$.

The advantage of our algorithm becomes more significant with the increase of K. The reason is that the influence of positions is not significant when K is small. When $K = 3$, the difference of the value of position weight is small. Thus, the CTR of an item is similar in the first three positions, which indicates that 1) a sub-optimal policy can also perform well especially when the recommended items of this sub-optimal policy is the same as the optimal one and the only gap is the order; 2) the bias introduced in learning is also relatively small. Then, baseline algorithms without considering the behavior model of users or using other inaccurate models would obtain good performance (for example C^2UCB). However, when K becomes larger, the performance of C^2UCB is exceeded by other algorithms that involve click models. Moreover, the performances of different click models can be illustrated more straightforwardly. The performance of CM-LinUCB becomes worse compared with DCM-LinUCB and UBM-LinUCB, since it only considers one click.

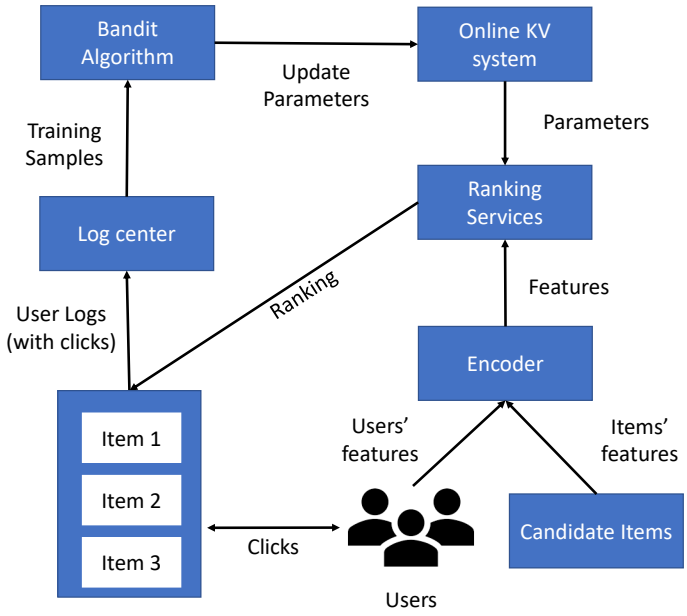


Fig. 4.7: The ranking system of our online experiment. The parameters of our algorithm are stored in the online KV system and used to rank items when users enter our scenario.

This result is similar with that of the paper [49], in which the authors indicate that contextual bandit algorithms perform well when the ratio $\frac{K}{m}$ is small.

4.5.5 Online experiment

We implement an online experiment in Taobao. In our scenario, 12 items should be selected from about 40,000 commodities and displayed on users' mobile devices. Bucket test or called A/B test is adopted. More specifically, users that enter our scenario will be randomly divided into different buckets. Each algorithm is deployed to one bucket and recommends about 5 millions of communities per day in the large-scale online evaluation. We use the same metrics as the offline experiment and report the performance of four algorithms. We do not evaluate PBM-UCB since it is infeasible to store the means and variances of all the 40,000 commodities in our online platform.

Fig. 4.7 illustrates the structure of our online platform. When users enter our scenario, their information will be concatenated with each item's feature. Then, an encoder mentioned in the previous section is used to reduce the dimension of features. In the ranking services, items are ranked according to the features and parameters. More specifically,

users are divided randomly to different buckets which correspond to different algorithms. According to the ids of buckets, parameters restored in the online KV system are read and used to compute the rank given features. The results are displayed in a mobile application and users give their feedback by interaction. The log center receives the raw logs and transfers them to samples to train different bandit algorithms. Each algorithm only obtains samples from its own bucket to update parameters.

Fig. 4.5 shows the trends of cumulative mean of CTRs in three days with the increase of time. Since the number of samples is relatively small in the first few hours, the performance of different algorithms is unstable. Except the perturbation on the first two hours, our algorithm constantly performs the best and the improvement is stable. Notice that all the algorithms performs well before 6 a.m, which is normal in our platform. The reason would be that active users shift with time. More specifically, people who use our app from 0 a.m to 6 a.m usually are keen to shop and thus more likely to click recommended items. DCM-LinUCB and CM-LinUCB perform similarly, possibly because a portion of users only click one item. Since the portion is not fixed, their performances are fluctuated each day. Fig. 4.6 summarizes the total CTRs in each day. Our algorithm can improve $\boldsymbol{CTR}_{sum}$ and $\boldsymbol{CTR}_{set}$ steadily compared to benchmark algorithms, although each algorithm's CTRs change in different dates. The result is reliable and meaningful considering the complex online environment and the huge volume of the recommended items (about 60 million). Since CTR is usually positively correlated with the turnover and profit of an e-commerce platform, our algorithm can improve the profit directly.

4.6 Chapter Summary

In this chapter, we focus on addressing the position bias and pseudo-exposure problem in a large-scale mobile e-commerce platform. First, we model the online recommendation as a contextual combinatorial bandit problem. Second, to address the pseudo-exposure problem, we utilize the UBM model to estimate probabilities that users view items at different ranks and propose a novel method UBM-LinUCB to take advantage of the estimated probabilities in our linear reward model. Third, we prove a sublinear expected

cumulative regret bound $\tilde{O}(d\sqrt{TK})$ for our regret function. Finally, we conduct experiments to evaluate our algorithm in two real-world datasets by an unbiased estimator and an online scenario provided by Taobao, one of the most popular e-commerce platforms in the world. Results indicate that our algorithm improves CTRs under two metrics in both offline and online experiments and outperforms other algorithms.

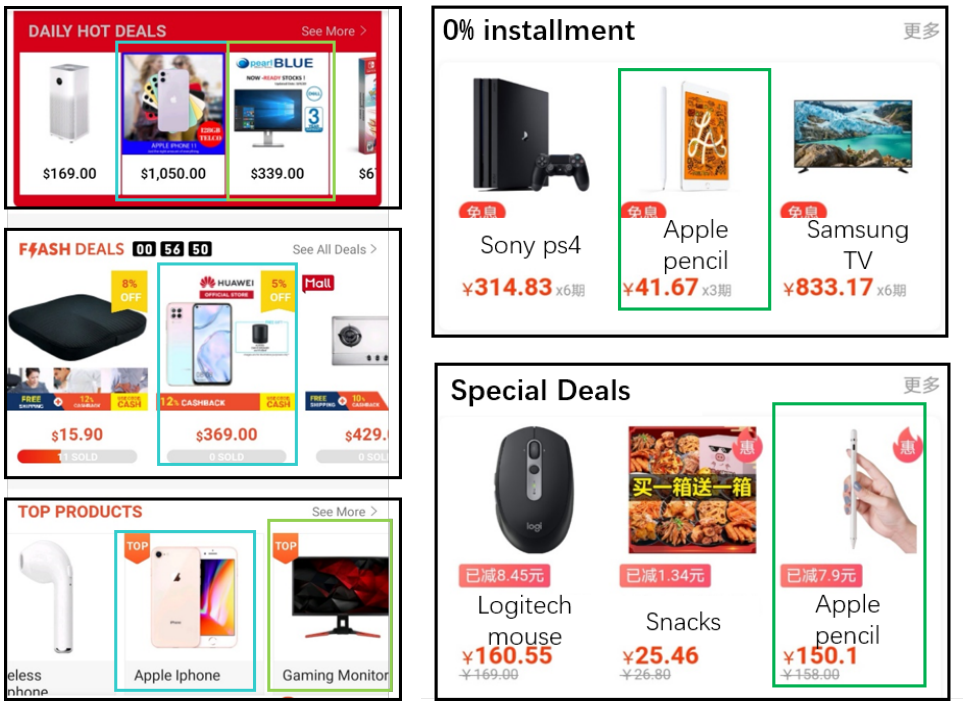
Chapter 5

Learning to Collaborate in Multi-Module Recommendation via Multi-Agent Reinforcement Learning

5.1 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. 5.1. 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. 5.1a and the apple pencil is recommended by two modules in Fig. 5.1b.



(a) Example 1 (b) Example 2

Fig. 5.1: Two examples of the multi-module recommendation scenarios. The black boxes represent modules. Boxes in different colors mark similar items in different modules. In sub-figure 5.1a, phones and monitors appear more than once. Meanwhile, apple pencils are recommended by two modules in sub-figure 5.1b.

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 [17, 103]. 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 chapter[1], we propose a novel approach for the multi-module recommendation problem. The first key contribution of this chapter is a novel multi-agent

[1]This chapter has been published in [25].

cooperative reinforcement learning structure. The structure is inspired by a solution concept in game theory called correlated equilibrium [4] in which the predefined signals received by the agents guide their actions. In this way, a better solution can be obtained comparing to Nash equilibrium, which does not involve the signal. 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.

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 [32] 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.

5.2 Problem Statement and Formulation

We firstly introduce the details of the multi-module recommendation problem. Fig. 5.2 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 page* shows a list of recommended

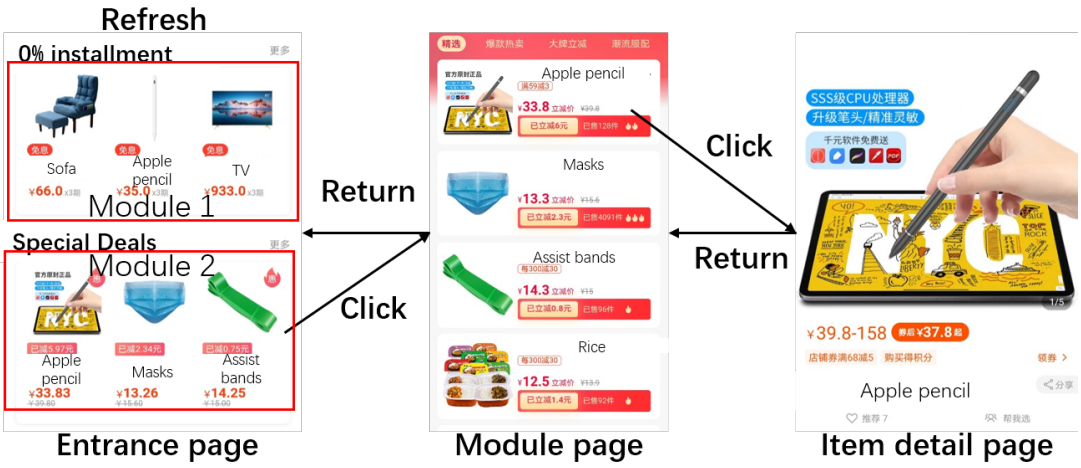


Fig. 5.2: The flow of a multi-module recommendation system. Pages shown in this figure are the entrance page, the module page, and the item detail page. The entrance page contains two modules.

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 recommend 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) [54]. 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 [17, 103].

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 $a^i = [a^i_1, \dots, a^i_j, \dots]$ 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 $R = [R^1, \dots, R^N]$, 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 $r = [r^1, \dots, r^N]$ represents rewards for N agents.

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

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

rather than the independent reward of each agent r^i_t, 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.

5.3 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. 5.3 illustrates the structure of our algorithm, which is based on MADDPG [60]. 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

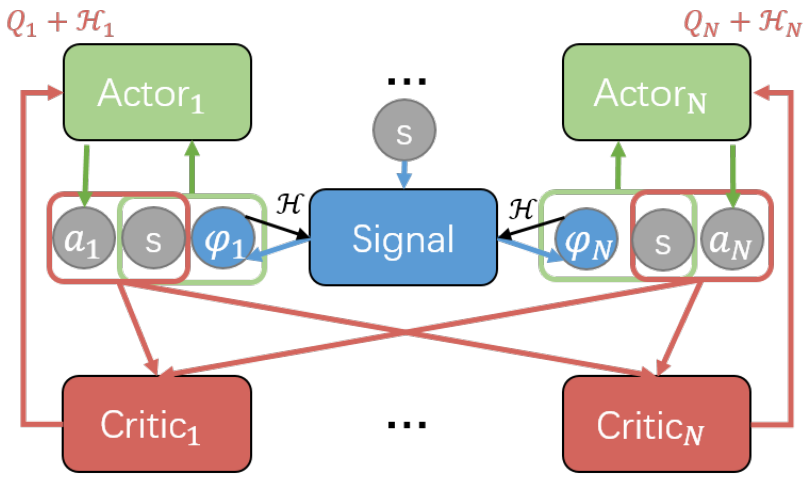
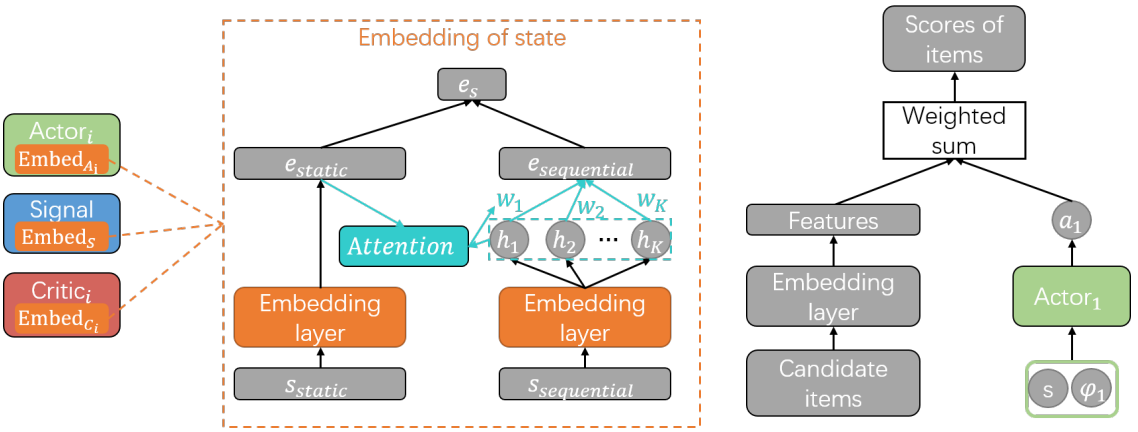


Fig. 5.3: The architecture of our approach. During the training, critics leverage other agents' actions to output the estimate of Q value. For the execution, each agent does not communicate with each other.

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.

5.3.1 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. 5.4a. As mentioned in Section 5.2, 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



(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.

Fig. 5.4: State embedding and the structure of ranking.

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 $0 \leq w_k \leq 1$ 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

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

Actor The structure of actors are illustrated in Fig. 5.4b. 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 [23], 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 [23], 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.

5.3.2 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 $V^i_\eta(s_t)$, two Q-functions $Q^i_{\theta_j}(s_t, a_t),\ j \in \{1, 2\}$, a stochastic policy $\pi^i_\tau(s_t, \phi^i_t)$ and a 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) [23] for each agent. Differing from standard RL that maximizes the expected sum of rewards

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

Algorithm 6: 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 $\hat{\eta}_{t+1} = (1 - \delta)\hat{\eta}_t + \delta\eta_t$;

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

$$Q^i(s_t, a_t) = \mathbb{E}_{s\sim\rho^\pi, a^i\sim\pi^i}\left[r(s_t, a_t) + \gamma V^i(s_{t+1})\right], \tag{5.1}$$

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 [23].

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

$$J_V^i(\eta) = \mathbb{E}_{s_t\sim D}\left[\frac{1}{2}\left(V_\eta^i(s_t) - \mathbb{E}_{a_t\sim\pi_\tau}\left[Q_\theta^i(s_t, a_t) - \log\pi_\tau^i(a_t^i|s_t, \phi_t^i)\right]\right)^2\right], \tag{5.2}$$

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:

$$\hat{\nabla}_\eta J_V^i(\eta) = \nabla_\eta V_\eta^i(s_t) \left(V_\eta^i(s_t) - Q_\theta^i(s_t, a_t) + \log \pi_\tau^i(a_t^i | s_t, \phi_t^i) \right), \tag{5.3}$$

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

$$J_Q^i(\theta_j) = \mathbb{E}_{(s_t, a_t) \sim D} \left[\frac{1}{2} \left(Q_{\theta_j}^i(s_t, a_t) - \hat{Q}^i(s_t, a_t) \right)^2 \right], \tag{5.4}$$

with

$$\hat{Q}^i(s_t, a_t) = r_t^i(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P} \left[V_{\hat{\eta}}^i(s_{t+1}) \right], \tag{5.5}$$

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

$$\hat{\nabla}_{\theta_j} J_Q^i(\theta_j) = \nabla_{\theta_j} Q_{\theta_j}^i(s_t, a_t) \left(Q_{\theta_j}^i(s_t, a_t) - \hat{Q}^i(s_t, a_t) \right). \tag{5.6}$$

Actor. For each actor, the objective is to maximize the Q value with the entropy term, since Q value introduced in Eq. (5.1) does not include $\mathcal{H}(\pi^i(\cdot | s_t, \phi_t^i)$:

$$J_\pi^i(\tau) = -\mathbb{E}_{s_t \sim D, a^i \sim \pi^i} \left[Q_\theta^i(s_t, a_t^{-i}, a_t^i) - \log \pi_\tau^i(a_t^i | s_t, \phi_t^i) \right], \tag{5.7}$$

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 [39]

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

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

$$\begin{aligned} \hat{\nabla}_\tau J_\pi^i(\tau) =& \nabla_\tau \log \pi_\tau^i(a_t^i | s_t, \phi_t^i) + \left(-\nabla_{a_t^i} Q_\theta^i(s_t, a_t^{-i}, a_t^i) + \right. \\ & \left. \nabla_{a_t^i} \log \pi_\tau^i(a_t^i | s_t, \phi_t^i) \right) \nabla_\tau f_\tau(\epsilon_t; s_t, \phi_t^i). \end{aligned} \tag{5.8}$$

These updates are extended from soft actor-critic algorithm [23].

The entropy-regularized signal network. Now we introduce the update of the signal network. Since the signal network aims at maximizing the overall reward, the objective function is

$$J_\phi(\xi) = \frac{1}{N}\sum_i -\mathbb{E}_{s_t, a_t^{-i}\sim D}\left[Q_\theta^i(s_t, a_t^{-i}, a_t^i)\right]. \tag{5.9}$$

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. According to the definition

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

we add it into the objective function of the signal network with a hyper-parameter α controlling the weight of the entropy term:

$$J_\phi(\xi) = \frac{1}{N}\sum_i\left[\mathbb{E}_{s_t, a_t^{-i}\sim D, \phi^i\sim\Phi^i}\left[-Q_\theta^i(s_t, a_t^{-i}, \pi^i(s_t, \phi_t^i)) + \alpha\log\Phi^i(\phi_t^i|s_t)\right]\right]. \tag{5.10}$$

According to [33], 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 $\epsilon \sim \mathcal{N}(0, I)$:

$$\begin{aligned}\hat{\nabla}_\xi J_\Phi(\xi) = \frac{1}{N}\sum_i\Big[&\alpha\nabla_\xi\log\Phi_\xi^i(\phi_t^i|s_t) + \Big(\alpha\nabla_{\phi_t^i}\log\Phi^i(\phi_t^i|s_t) - \\ &\nabla_{a_t^i}Q_\theta^i(s_t, a_t^{-i}, a_t^i)\nabla_{\phi_t^i}\pi_\tau^i(a_t^i|s_t, \phi_t^i)\Big)\nabla_\xi g_\xi^i(\epsilon_t; s_t)\Big].\end{aligned} \tag{5.11}$$

The whole algorithm is shown in Algorithm 6. 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.

Algorithm 7: Offline testing procedure.

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

5.4 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.

5.4.1 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.

5.4.2 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

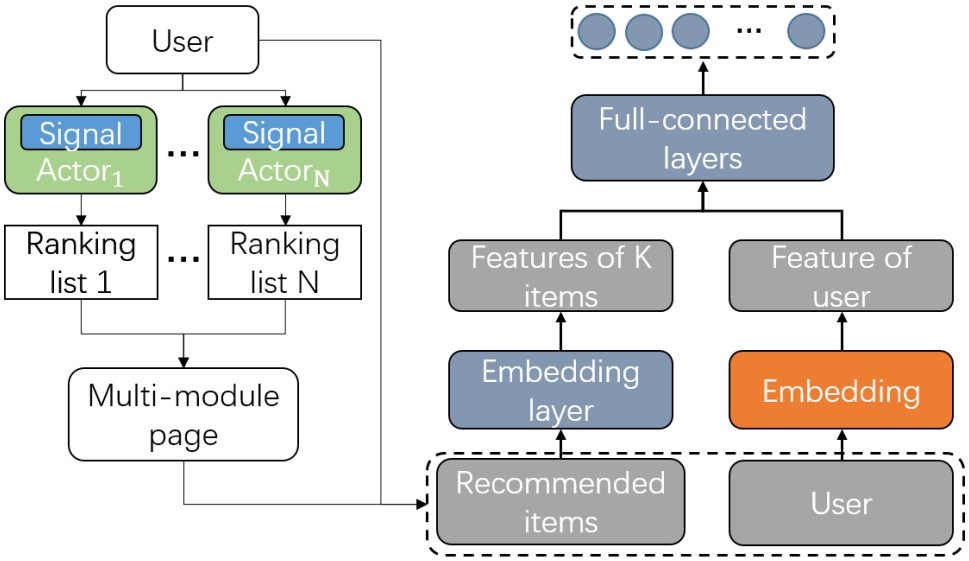


Fig. 5.5: The structure of our simulator. The inputs are all the recommended items on one web page and the information of a user. The output is a vector including the probabilities that these items are clicked.

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 items are 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 ranks 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 7.

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 [102]. The structure of the simulator is shown in Fig. 5.5. 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. 5.4a. 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 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 7 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 [63] and nDCG [93]. 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. 5.4a. 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

Table 5.1: 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)	0.433	0.052	0.485	0.397	0.050	0.447
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**

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.

5.4.3 Baselines

Our algorithm is compared with the following baselines:

- **L2R** [55]: 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** [52]: 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** [60]: 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** [19]: 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** [75]: 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.

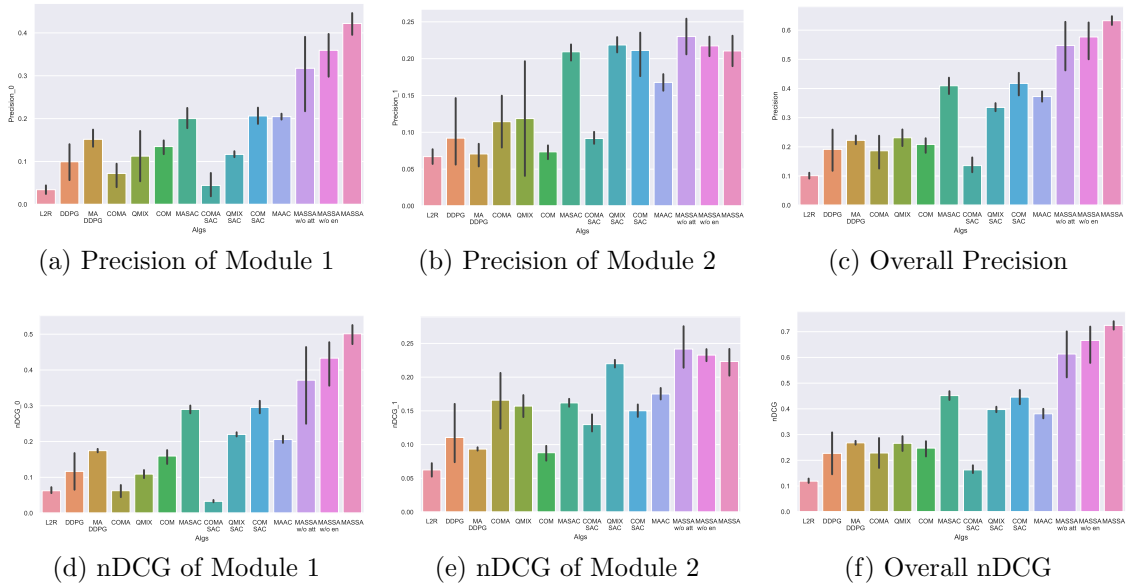
- **COM**: COM is a simple extension of the methods [17] by letting actors choose actions simultaneously. Actors send messages to others during execution. Although this algorithm violates the restriction that different modules cannot communicate, the comparison aims to illustrate the performance in environments that allow communication.

- **MASAC**: This algorithm is an extension of MADDPG by applying soft actor-critic [23], 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** [32]: 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.



(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

Fig. 5.6: The results of the online experiment.

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 [103] is a model-based version of COM and other baselines (including ours) are model-free methods. Thus, we only compare to COM considering fairness.

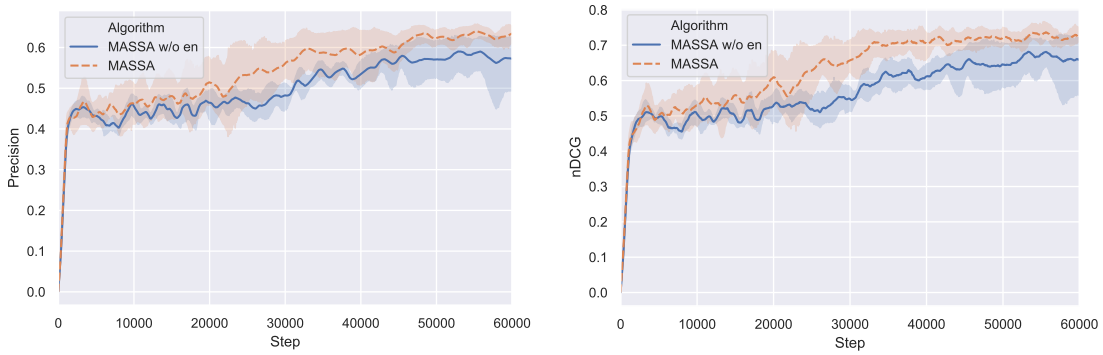
5.4.4 Result

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

Offline Testing

The results of our offline evaluation are shown in Table 5.1. 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



(a) Precision (b) nDCG

Fig. 5.7: 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.

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.

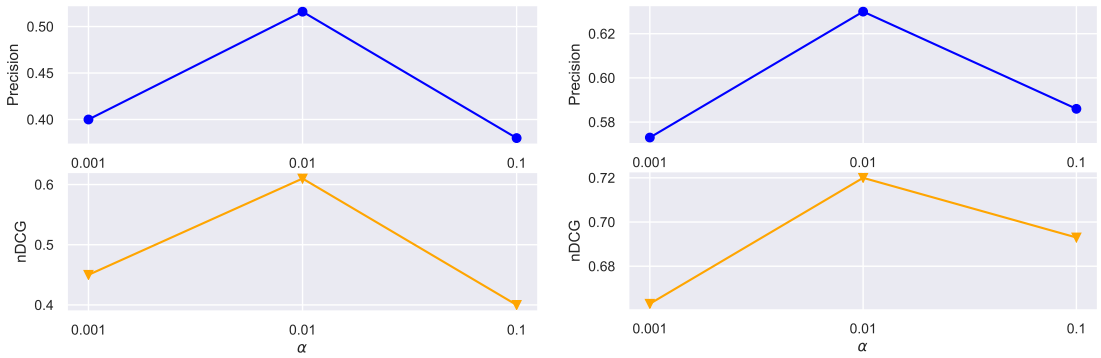
Secondly, the metrics of module 1 are better than that of module 2, which is caused by different properties of these two modules. As shown in Fig. 5.2, module 1 is at the top of the web page. Thus, users are more likely to be attracted by module 1 and ignore another module, especially when the items 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.

Online Testing

For the online experiment, Fig. 5.6 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



(a) Offline (b) Online

Fig. 5.8: The performance with the change of α

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 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. 5.7. 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.

The influence of α

Fig. 5.8 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.

5.5 Chapter Summary

In this chapter, 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.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

More and more attempts have been made to apply reinforcement learning methods to the recommendation domain in recent years. In this thesis, we focus on two settings to model recommendation problems, i.e., multi-arm bandit setting and multi-agent reinforcement learning setting.

In chapter 3, we consider how to understand users' feedback and obtain the optimal policy that is able to select users' preferred actions. The key issue is to model the uncertainty of human feedback. More concretely, users would provide any kind of feedback to either optimal or non-optimal actions, which include positive, negative and no feedback. Hence, we propose a novel human feedback model that consists of three functions representing the probability distributions of three types of feedback. To handle two sets of parameters in the policy and feedback model, we propose an approximated Expectation Maximization (EM) algorithm combined with Gradient Descent (GD) method that alternatively updates these parameters. Additionally, two experiments involving human participants are conducted in real-world scenarios. Our algorithm outperforms baselines under multiple metrics. Robustness analysis demonstrates that our algorithm performs well even if users do not follow our feedback model.

In chapter 4, we focus on addressing the influence caused by positions of items shown in web pages. More precisely, the influence includes two types: position bias and pseudo-exposure problem. Position bias means that a higher ranking usually leads to more clicks for a fixed item. Pseudo-exposure indicates that we do not know exactly whether items

are viewed or not. We firstly model the recommendation problem as a contextual combinatorial bandit problem. To address the above-mentioned two issues, we propose a novel algorithm, where the user-browsing model is combined with a contextual combinatorial bandit method to estimate position influence. Furthermore, we prove a sublinear expected cumulative regret bound $\tilde{O}(d\sqrt{TK})$ for our algorithm. Finally, we conduct both offline and online experiments. For the offline experiment, we propose an unbiased estimator to evaluate different methods in two real-world datasets. Moreover, we deploy these methods in Taobao for online testing. Our method is superior to other baselines under two metrics in both experiments.

In chapter 5, we consider the multi-module recommendation problem, in which a page contains multiple modules that recommend items processing different specific properties. Since different modules are displayed on one page and 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. To prompt cooperation and maximize the overall reward, we design a signal network inspired by the correlated equilibrium in game theory, which sends additional signals to all the modules. Then, we propose an entropy-regularized loss function for the signal network to coordinate agents' exploration. The experimental result verifies that our method outperforms other state-of-the-art learning algorithms.

In summary, we focus on solving recommendation problems by reinforcement learning methods. More specifically, we propose new methods for multi-arm bandit setting and multi-agent reinforcement learning setting to address challenges in various scenarios. Online and offline experiments are conducted to illustrate the performance of various methods.

6.2 Future Directions

There are many works that need to be done to address recommendation problems by reinforcement learning methods.

6.2.1 Offline Training

Different from supervised learning methods which can be trained by the dataset including features and labels, reinforcement learning methods require interactive environments, which is one of the biggest obstacles to their widespread adoption, especially in the recommendation domain. The objective of recommendation problems usually is to improve some metrics related to users' behaviours, such as click-through rate, conversion rate. Thus, the user group is the most important component of the interactive environment. However, the user group always changes, it is hence hard to model users' behaviours and build a simulated interactive environment. Although there are a few works making some attempts to build simulators [30, 81], some challenges still exist: 1) the features of items and users always change in real-world platforms, while the features are assumed fixed in simulators; 2) simulated actions of users not in the training dataset would not be accurate. Considering these drawbacks, most papers use the offline approach for training and evaluation, which is similar to traditional recommender system papers [102, 105]. They directly use RL methods to re-rank items shown in historical logs and the reward is defined as historical clicks of users. Although this approach is reasonable for some simple scenarios, one key drawback is that sequential processes are omitted. RL agents only interact with each user once during the training and evaluation, which cannot exactly demonstrate the performance of RL methods in sequential decision processes.

Offline reinforcement learning aims at obtaining a good policy by offline datasets without interactions with environments. These methods neither need simulators nor ignore sequential processes during training. We can divide offline RL into two types according to their purposes. 1) offline evaluation methods aim at evaluating RL policies by data generated by other policies and 2) offline training methods focus on how to obtain a better policy than the one used to generate data. Both of these methods need datasets including sequential interaction data (s_t, a_t, r_t), namely trajectories, rather than one-time interaction data (s, r) used in supervised learning methods.

For offline evaluation methods, the key idea is to use important sampling methods to re-weight reward [71]. More specifically, since the expected return of a policy is $\sum_a \pi(a|s)r(s,a)$, we can estimate the return of a new policy β by

$$\sum_a \pi(a|s)\frac{\beta(a|s)}{\pi(a|s)}r(s,a)$$

if we have a dataset generated by π. Although this estimator is unbiased, the variance is usually too high to obtain a good policy in many applications. Doubly robust estimators have been derived to balance variance and bias [35, 88].

For offline training methods, fitted Q-learning [77] firstly proposes to modify Q-learning method to minimize the gap between the left and right side of the Bellman equation

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}\sim\mathcal{P}(s_t,a_t),a_{t+1}\sim\pi(s_{t+1})}\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})\right].$$

However, when neural networks or other functions are used to approximate Q values, distributional shift affects training seriously. The agent we trained would select actions or visit states that rarely appear in the dataset. The true values of these actions or states are not accurately learned due to the lack of data. If the agent acts based on estimated value, the performance would be poor. To address this issue, the most direct method is to make the learned policy similar to the behaviour policy which generates the training dataset. Thus, different divergence constraints are proposed [66, 69, 82, 98]. Another approach proposes to estimate the uncertainty of Q values as a penalty term [45].

However, as far as I know, no work extends these offline methods to the recommendation domain. How to apply offline reinforcement learning in the recommendation domain is worth considering as future work. Since most companies use supervised learning methods for online services, we need to figure out if offline RL methods can learn a better policy from the data collected following these supervised methods. Another challenge is that the action space and state space are huge, it is possible that the dataset we obtain does not cover all actions and states. Or even worse, the dataset would only include a limited subset of actions or states since supervised learning methods do not explore.

6.2.2 Efficient Deployment

RL methods usually need to be trained in an online paradigm to interact with environments. If directly deployed online in an e-commerce platform, RL methods need a long period to converge and perform well. At the beginning of training, the performance is likely to be poor, which damages the income and revenue of a platform or company. This disadvantage restricts the application of RL methods. Thus, we need to consider the deployment efficiency, namely the amount of data an algorithm needs to converge [64].

For offline RL methods that are trained by collected datasets, the deployment efficiency is high. However, their performance usually is worse than online training methods [64]. For online training methods, they are required to collect millions of data online. Furthermore, if we pre-train these methods by offline dataset, their performance will firstly decrease in online training [66].

An interesting future work is to balance the performance and data efficiency in some online platforms. As far as I know, only two works [64, 66] consider the paradigm that combines offline training with online training. They first train a policy by an offline dataset and then deploy the policy in an online environment to continuously update. However, in the recommendation domain, there is no work considering the deployment efficiency of RL methods. The frequent change of state and action space of e-commerce platforms would be a challenge for deploying offline trained policies.

Appendices

Appendix A

Mathematical Proofs in Chapter 4

A.1 Proof of Lemma 4.1

Proof. Notice that $F(S_t) = 1$ except none of item is clicked, formally,

$$\mathbb{E}[F(S_t)] = 1 - \prod_{k=1}^{K} \left[1 - \gamma(a_{k,t}) w_{k,0}\right] \tag{A.1}$$

It suffices to show that $\prod_{k=1}^{K} \left[1 - \gamma(a_{k,t}) w_{k,0}\right]$ is minimized with $a^*_{1,t}, ..., a^*_{K,t}$ when the arms with the k-th highest value $\gamma(a^*_{k,t})$ aligns with the order of $w_{k,0}$, i.e., $\gamma(a^*_{1,t}) \geq \gamma(a^*_{2,t}) \geq ... \geq \gamma(a^*_{K,t})$.

$$\begin{aligned}
\log \prod_{k=1}^{K} \left(1 - \gamma(a_{k,t}) w_{k,0}\right) &= \sum_{k=1}^{K} \log \left(1 - \gamma(a_{k,t}) w_{k,0}\right) \\
&= - \sum_{k=1}^{K} \sum_{n=1}^{\infty} \frac{\left(\gamma(a_{k,t}) w_{k,0}\right)^n}{n} \\
&= - \sum_{n=1}^{\infty} \frac{1}{n} \sum_{k=1}^{K} \gamma^n(a_{k,t}) w^n_{k,0} \\
&\geq - \sum_{n=1}^{\infty} \frac{1}{n} \sum_{k=1}^{K} \gamma^n(a^*_{k,t}) w^n_{k,0} \\
&= \log \prod_{k=1}^{K} (1 - \gamma(a^*_{k,t}) w_{k,0})
\end{aligned}$$

The second equality follows by $\log(1 - x) = -\sum_{n=1}^{\infty} \frac{x^n}{n}, \forall x \in (0, 1)$ and the inequality follows by the Rearrangement inequality, which states that $\sum_i x_i y_i \geq \sum_i x_{\sigma(i)} y_{\sigma(i)}$ for any sequences of real numbers $x_1 \geq x_2 \geq ... \geq x_n$ and $y_1 \geq y_2 \geq ... \geq y_n$, and any permutations $x_{\sigma(1)}, ..., x_{\sigma(n)}$.

A.2 Proof of Lemma 4.2

Proof. Recall that $a_{k,t}$ is the selected arm by UBM-LinUCB at round t and position k based on Lemma 4.1. Let $\eta_{k,t} = \gamma(a^*_{k,t})w_{k,0} - \gamma(a_{k,t})w_{k,0} \geq 0$ and divide $\mathbb{E}[F(S_t)]$ into the sum of the probabilities that the first click appears at each position $k \in [1, K]$:

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{E}[F(S_t)] \\
&= \sum_{t=1}^{T} \sum_{k=1}^{K} \prod_{j=1}^{k-1} \left[1 - \left(\gamma(a^*_{j,t})w_{j,0} - \eta_{j,t}\right)\right] \left(\gamma(a^*_{k,t})w_{k,0} - \eta_{k,t}\right) \\
&\geq \sum_{t=1}^{T} \sum_{k=1}^{K} \prod_{j=1}^{k-1} \left(1 - \gamma(a^*_{j,t})w_{j,0}\right) \left(\gamma(a^*_{k,t})w_{k,0} - \eta_{k,t}\right) \\
&\geq \sum_{t=1}^{T} \sum_{k=1}^{K} \left[\prod_{j=1}^{k-1} \left(1 - \gamma(a^*_{j,t})w_{j,0}\right) \gamma(a^*_{k,t})w_{k,0} - \eta_{k,t}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}[F(S^*_t)] - \sum_{t=1}^{T} \sum_{k=1}^{K} \eta_{k,t}
\end{aligned} \tag{A.2}$$

Thus, we have:

$$\begin{aligned}
R(T) &\leq \sum_{t=1}^{T} \sum_{k=1}^{K} \eta_{k,t} \\
&= \sum_{t=1}^{T} \sum_{k=1}^{K} w_{k,0} \left[\gamma(a^*_{k,t}) - \gamma(a_{k,t})\right]
\end{aligned} \tag{A.3}$$

Then, we finish the proof.

A.3 Proof of Theorem 4.1

Assume that (1) $r_{k,a} = w_{k,0}\theta^* x_a$, (2) $1 \leq \sum_{k=1}^{K} w^2_{k,k-1} = \phi'_w < K$, (3) $\|x\|_2 \leq 1$ and (4) $\|\theta^*\|_2^2 \leq \beta$. The upper bound of regret is defined by Lemma 4.2:

$$R(T) \leq \mathbb{E}\left[\sum_{t=1}^{T} \sum_{k=1}^{K} w_{k,0} \left[\gamma(a^*_{k,t}) - \gamma(a_{k,t})\right]\right]$$

where $\gamma(a) = \theta^* x_a$ is the attractiveness of an item a and θ^* is the optimal parameter.

Proof. We define the event

$$\begin{aligned}
E = \{&|\langle x_{a_{k,t-1}}, \theta^* - \theta_{t-1}\rangle| \leq \alpha \sqrt{x^T_{a_{k,t-1}} A^{-1}_{t-1} x_{a_{k,t-1}}}, \\
&\forall a_{k,t} \in A_t, \forall t \leq T, \forall k \leq K\}
\end{aligned}$$

where $\langle \cdot, \cdot \rangle$ means the dot product and $\bar{E}$ is the complement of E. Then notice that $P(E) \leq 1$ and $w_{k,0}\left[\gamma(a^*_{k,t}) - \gamma(a_{k,t})\right] \leq 1$, we have

$$\begin{aligned} R(T) &\leq P(E)\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,0}\left[\gamma(a^*_{k,t}) - \gamma(a_{k,t})\right]\middle| E\right] \\ &+ P(\bar{E})\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,0}\left[\gamma(a^*_{k,t}) - \gamma(a_{k,t})\right]\middle| \bar{E}\right] \\ &\leq \mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,0}\left[\gamma(a^*_{k,t}) - \gamma(a_{k,t})\right]\middle| E\right] + TKP(\bar{E}) \end{aligned}$$

Using the definition of event E, we have

$$\gamma(a^*_{k,t}) = \langle x_{a^*_{k,t}}, \theta^* \rangle \leq \langle x_{a^*_{k,t}}, \theta_{t-1} \rangle + \alpha\sqrt{x^T_{a^*_{k,t}} A^{-1}_{t-1} x_{a^*_{k,t}}}$$

under event E. Since $a_{k,t}$ is the arm that our algorithm chooses in round t, we have

$$\begin{aligned} &\langle x_{a^*_{k,t}}, \theta_{t-1} \rangle + \alpha\sqrt{x^T_{a^*_{k,t}} A^{-1}_{t-1} x_{a^*_{k,t}}} \\ \leq &\langle x_{a_{k,t}}, \theta_{t-1} \rangle + \alpha\sqrt{x^T_{a_{k,t}} A^{-1}_{t-1} x_{a_{k,t}}} \end{aligned}$$

Thus,

$$\gamma(a_{k,t}) \leq \gamma(a^*_{k,t}) \leq \langle x_{a_{k,t}}, \theta_{t-1} \rangle + \alpha\sqrt{x^T_{a_{k,t}} A^{-1}_{t-1} x_{a_{k,t}}}$$

Using this property and $\gamma(a) = \theta^* x_a$, we have

$$\begin{aligned} \gamma(a^*_{k,t}) - \gamma(a_{k,t}) &\leq \langle x_{a_{k,t}}, \theta_{t-1} - \theta^* \rangle + \alpha\sqrt{x^T_{a_{k,t}} A^{-1}_{t-1} x_{a_{k,t}}} \\ &\leq 2\alpha\sqrt{x^T_{a_{k,t}} A^{-1}_{t-1} x_{a_{k,t}}} \end{aligned}$$

The second inequality is based on the definition of event E. Thus, we have

$$R(T) \leq 2\alpha\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,0}\sqrt{x^T_{a_{k,t}} A^{-1}_{t-1} x_{a_{k,t}}}\right] + TKP(\bar{E})$$

Using the bounds proved in the next two sections, we have

$$R(T) \leq 2\alpha\sqrt{2TKd\ln\left(1 + \frac{\phi'_w T}{\lambda d}\right)} + TK\delta$$

when $\lambda \geq \phi'_w \geq 1$ and

$$\alpha \geq \sqrt{d\ln\left(1 + \frac{\phi'_w T}{d\lambda}\right) + 2\ln\left(\frac{1}{\delta}\right)} + \sqrt{\lambda\beta}.$$

Let $\delta = \frac{1}{TK}$, we finish the proof.

A.3.1 Bound of $\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,0}\sqrt{x_{a_{k,t}}^T A_{t-1}^{-1} x_{a_{k,t}}}$

Let $z_{k,t} = \sqrt{x_{a_{k,t}}^T A_{t-1}^{-1} x_{a_{k,t}}}$.

Lemma A.1 *For any $\lambda \geq \phi'_w$, $\forall k'$ s.t. $0 \leq k' \leq k-1$, then for any time T,*

$$\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k'} z_{k,t} \leq \sqrt{2TKd \ln\left(1 + \frac{\phi'_w T}{\lambda d}\right)}$$

Proof. From Cauchy-Schwarz inequality, we give an upper bound

$$\begin{aligned}\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,0} z_{k,t} &\leq \sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k'} z_{k,t} \\ &\leq \sqrt{TK}\sqrt{\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k'}^2 z_{k,t}^2}\end{aligned}$$

, $\forall k'$ *s.t.* $0 \leq k' \leq k-1$. The first inequation uses the property of $w_{k,k'}$ in the main body of Chapter 4 to leverage the following lemma (cannot be used for $w_{k,0}$ due to the definition of A_t):

Lemma A.2 *(Lemma 4.4 [51]) If $\lambda \geq \sum_{k=1}^{K} w_{k,k'}^2$, $\forall k'$ s.t. $0 \leq k' \leq k-1$, then for any time T,*

$$\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k'}^2 z_{k,t}^2 \leq 2d \ln\left(1 + \frac{1}{\lambda d}\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k'}\right)$$

Using Lemma A.2 and the property that

$$\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k'} \leq \sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k-1}$$

, the upper bound is

$$\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k'} z_{k,t} \leq \sqrt{2TKd \ln\left(1 + \frac{\phi'_w T}{\lambda d}\right)} \tag{A.4}$$

A.3.2 Bound of $P(\bar{E})$

Lemma A.3 *For any $\delta \in (0,1), \lambda \geq \phi'_w \geq 1$, $\beta \geq \|\theta^*\|_2^2$, if*

$$\alpha \geq \left[\sqrt{2\ln\left(1 + \frac{\phi'_w T}{2}\right) + 2\ln\left(\frac{1}{\delta}\right)} + \sqrt{\lambda\beta} \right],$$

we have $P(\bar{E}) \geq \delta$.

Proof. We first define

$$\bar{\eta}_{k,t} = w_{k,k'} \left[\gamma(a_{k,t}) - \gamma_t(a_{k,t}) \right]$$

to use Theorem 1 of [1], where $\gamma_t(a_{k,t}) = \theta_t x_{k,t}$ and $\gamma(a_{k,t}) = \theta^* x_{k,t}$. $\bar{\eta}_{k,t}$ is a Martingale Difference Sequence and bounded by $[-1,1]$. Thus, it is a conditionally sub-Gaussian with constant $R = 1$. We also define that

$$\begin{aligned} S_t &= \sum_{t=1}^{T} \sum_{k=1}^{K} w_{k,k'} x_{a_{k,t}} \bar{\eta}_{k,t} \\ &= b_t - \sum_{t=1}^{T} \sum_{k=1}^{K} w_{k,k'}^2 x_{a_{k,t}} \gamma_t(a_{k,t}) \\ &= b_t - \left[\sum_{t=1}^{T} \sum_{k=1}^{K} w_{k,k'}^2 x_{a_{k,t}} x_{a_{k,t}}^T \right] \theta^* \end{aligned}$$

where the second equation uses the definition of b_t and $\mathbb{E}[r_{a_{k,t}}] = w_{k,k'} \gamma(a_{k,t})$. Then, according to the Theorem 1 of [1], for any $\delta \in (0,1)$, with probability at least $1 - \delta$,

$$\|S_t\|_{A_t^{-1}} \leq \sqrt{2\ln\left(\frac{\det(A_t)^{\frac{1}{2}} \det(A_0)^{-\frac{1}{2}}}{\delta} \right)} \tag{A.5}$$

where $\|S_t\|_{A_t^{-1}} = \sqrt{S_t^T A_t^{-1} S_t}$. Moreover, using the trace-determinant inequality and the assumption 1) $\|x_a\| \leq 1$, 2) $w_{k,k'} \leq w_{k,k-1}$, we have

$$\det(A_t)^{\frac{1}{d}} \leq \frac{trace(A_t)}{d} = \lambda + \frac{1}{d} \sum_{t=1}^{T} \sum_{k=1}^{K} w_{k,k'}^2 \|x_{a_{k,t}}\|_2^2 \leq \lambda + \frac{\phi'_w T}{d}$$

Since $A_0 = \lambda I$, we have

$$\|S_t\|_{A_t^{-1}} \leq \sqrt{d\ln\left(1 + \frac{\phi'_w T}{d\lambda}\right) + 2\ln\left(\frac{1}{\delta}\right)} \tag{A.6}$$

Since $\lambda \geq 1$, notice that

$$\begin{aligned} A_t\theta_t = b_t &= S_t + \left[\sum_{t=1}^{T}\sum_{k=1}^{K} w_{k,k'}^2 x_{a_{k,t}} x_{a_{k,t}}^T\right]\theta^* \\ &= S_t + (A_t - \lambda I)\theta^* \end{aligned}$$

Thus,

$$\theta_t - \theta^* \leq A_t^{-1}(S_t - \lambda\theta^*)$$

Then, for any $a \in A, k \leq K$, based on Cauchy-Schwarz inequality and triangle inequality, we have

$$\begin{aligned} |\langle x_{a_{k,t}}, \theta_t - \theta^*\rangle| &\leq |x_a^T A_t^{-1}(S_t - \lambda\theta^*)| \\ &\leq \|x_a^T\|_{A_t^{-1}} \|(S_t - \lambda\theta^*)\|_{A_t^{-1}} \\ &\leq \|x_a^T\|_{A_t^{-1}} \left[\|S_t\|_{A_t^{-1}} + \lambda\|\theta^*\|_{A_t^{-1}}\right] \\ &\leq \|x_a^T\|_{A_t^{-1}} \left[\|S_t\|_{A_t^{-1}} + \sqrt{\lambda K}\right] \end{aligned}$$

where we use the property

$$\|\theta^*\|_{A_t^{-1}}^2 \leq \frac{1}{Eig_{\min}(A_t)}\|\theta^*\|_2^2 \leq \frac{1}{\lambda}\|\theta^*\|_2^2 \leq \frac{\beta}{\lambda}$$

in the last inequality and $Eig(A_t)$ is the eigenvalue of A_t. Then using the Eq. (A.6), with probability $1 - \delta$,

$$\begin{aligned} &|\langle x_{a_{k,t}}, \theta_{t-1} - \theta^*\rangle| \\ \leq &\|x_a^T\|_{A_t^{-1}} \left[\sqrt{d\ln\left(1 + \frac{\phi'_w T}{d\lambda}\right) + 2\ln\left(\frac{1}{\delta}\right)} + \sqrt{\lambda\beta}\right] \end{aligned}$$

Recall the definition of E, when

$$\alpha \geq \sqrt{d\ln\left(1 + \frac{\phi'_w T}{d\lambda}\right) + 2\ln\left(\frac{1}{\delta}\right)} + \sqrt{\lambda\beta},$$

then $P(E) \leq 1 - \delta$ and thus $P(\bar{E}) \geq \delta$.

A.4 UBM estimator

Let $\langle W, \Phi(a, \cdot, \cdot|X)\rangle = \sum_{k=1}^{K}\sum_{k'=0}^{k} w_{k,k'}\Phi(a, k, k'|X)$, where $w_{k,k'}$ is the position bias for the position k when the item at rank k' is clicked and $k' = 0$ if there is no click before position k. Then, we have:

$$
\begin{aligned}
V_{UBM}(\Phi) &= \mathbb{E}_X\left[\mathbb{E}_{\substack{S\sim\Phi(\cdot|X)\\ r\sim D(\cdot|X)}}\left[\sum_{k=1}^{K}\sum_{k'=0}^{k} r(a_k, k, k'|X)\right]\right] \\
&= \mathbb{E}_X\left[\sum_{S}\Phi(S|X)\left[\sum_{k=1}^{K}\sum_{k'=0}^{k}\sum_{a\in\mathcal{A}} r(a, k, k'|X)\mathbb{1}\{a_k = a, c_{last} = k'\}\right]\right] \\
&= \mathbb{E}_X\left[\sum_{k=1}^{K}\sum_{k'=0}^{k}\sum_{a\in\mathcal{A}} r(a, k, k'|X)\sum_{S}\Phi(S|X)\mathbb{1}\{a_k = a, c_{last} = k'\}\right] \\
&= \mathbb{E}_X\left[\sum_{k=1}^{K}\sum_{k'=0}^{k}\sum_{a\in\mathcal{A}} r(a, k, k'|X)\Phi(a, k, k'|X)\right] \\
&= \mathbb{E}_X\left[\sum_{a\in\mathcal{A}}\gamma(a|X)\sum_{k=1}^{K}\sum_{k'=0}^{k} w_{k,k'}\Phi(a, k, k'|X)\right] \\
&= \mathbb{E}_X\left[\sum_{a\in\mathcal{A}}\gamma(a|X)\sum_{k=1}^{K}\sum_{k'=0}^{k} w_{k,k'}\Phi(a, k, k'|X)\right] \\
&= \mathbb{E}_X\left[\sum_{a\in\mathcal{A}}\gamma(a|X)\langle\tilde{W}, \pi(a, \cdot, \cdot|X)\rangle\frac{\langle\tilde{W}, \Phi(a, \cdot, \cdot|X)\rangle}{\langle W, \pi(a, \cdot, \cdot|X)\rangle}\right] \\
&= \mathbb{E}_X\left[\mathbb{E}_{\substack{S\sim\pi(\cdot|X)\\ r\sim D(\cdot|X)}}\left[\sum_{k=1}^{K}\sum_{k'=0}^{k} r(a_k, k, k'|X)\frac{\langle\tilde{W}, \Phi(a_k, \cdot, \cdot|X)\rangle}{\langle\tilde{W}, \pi(a_k, \cdot, \cdot|X)\rangle}\right]\right]
\end{aligned}
$$

where c_{last} is the position of the last click before k. Thus, with the assumption that users' behaviors follow the UBM model, the estimator is unbiased.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

[2] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[4] Robert J Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.

[5] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 449–458, 2017.

[6] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[7] A Lord Birnbaum. Some latent trait models and their use in inferring an examinee's ability. *Statistical theories of mental test scores*, 1968.

[8] Craig Boutilier. A pomdp formulation of preference elicitation problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 239–246, 2002.

[9] Thomas Cederborg, Ishaan Grover, Charles L Isbell, and Andrea Lockerd Thomaz. Policy shaping with human teachers. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3366–3372, 2015.

[10] Olivier Chapelle and Ya Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the International Conference on World Wide Web*, pages 1–10, 2009.

[11] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the International Conference on Machine Learning*, pages 151–159, 2013.

[12] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *Proceedings of the International Conference on Learning Representations*, 2018.

[13] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. Reinforcement learning based recommender system using biclustering technique. *arXiv preprint arXiv:1801.05532*, 2018.

[14] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. Click models for web search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1–115, 2015.

[15] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 87–94, 2008.

[16] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B*, pages 1–38, 1977.

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

[18] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[19] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint cs.AI/1705.08926*, 2017.

[20] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2974–2982, 2018.

[21] Simon French. *Decision theory: An Introduction to the Mathematics of Rationality*. Halsted Press, 1986.

[22] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2625–2633, 2013.

[23] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[24] Xu He, Bo An, Yanghua Li, Haikai Chen, Qingyu Guo, Xin Li, and Zhirong Wang. Contextual user browsing bandits for large-scale online mobile recommendation. In *Proceedings of the ACM Conference on Recommender Systems*, pages 63–72, 2020.

[25] Xu He, Bo An, Yanghua Li, Haikai Chen, Rundong Wang, Xinrun Wang, Runsheng Yu, Xin Li, and Zhirong Wang. Learning to collaborate in multi-module recommendation via multi-agent reinforcement learning without communication. In *Proceedings of the ACM Conference on Recommender Systems*, pages 210–219, 2020.

[26] Xu He, Haipeng Chen, and Bo An. Learning behaviors with uncertain human feedback. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 131–140, 2020.

[27] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3215–3222, 2018.

[28] Mark K Ho, Michael L Littman, Fiery Cushman, and Joseph L Austerweil. Teaching with rewards and punishments: Reinforcement or communication? In *CogSci*, 2015.

[29] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. *arXiv preprint arXiv:1803.00710*, 2018.

[30] Jin Huang, Harrie Oosterhuis, Maarten de Rijke, and Herke van Hoof. Keeping dataset biases out of the simulation: A debiased simulator for reinforcement learning based recommender systems. In *Proceedings of the ACM Conference on Recommender Systems*, pages 190–199, 2020.

[31] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Navrekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767*, 2019.

[32] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. *arXiv preprint arXiv:1810.02912*, 2018.

[33] Martin Jankowiak and Fritz Obermeyer. Pathwise derivatives beyond the reparameterization trick. *arXiv preprint arXiv:1806.01851*, 2018.

[34] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pages 7254–7264, 2018.

[35] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 652–661, 2016.

[36] Thorsten Joachims, Laura A Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–161, 2005.

[37] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the International Conference on Machine Learning*, pages 440–447, 2008.

[38] Sumeet Katariya, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. DCM bandits: Learning to rank with multiple clicks. In *Proceedings of the International Conference on Machine Learning*, pages 1215–1224, 2016.

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[40] David C Knill and Alexandre Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, 27(12):712–719, 2004.

[41] W Bradley Knox, Brian D Glass, Bradley C Love, W Todd Maddox, and Peter Stone. How humans teach agents. *International Journal of Social Robotics*, 4(4):409–421, 2012.

[42] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the International Conference on Knowledge capture*, pages 9–16, 2009.

[43] W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 5–12, 2010.

[44] Junpei Komiyama, Junya Honda, and Akiko Takeda. Position-based multiple-play bandit problem with unknown position bias. In *Advances in Neural Information Processing Systems*, pages 4998–5008, 2017.

[45] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[46] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. Cascading bandits: Learning to rank in the cascade model. In *Proceedings of the International Conference on Machine Learning*, pages 767–776, 2015.

[47] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvári. Combinatorial cascading bandits. In *Advances in Neural Information Processing Systems*, pages 1450–1458, 2015.

[48] Paul Lagrée, Claire Vernade, and Olivier Cappe. Multiple-play bandits in the position-based model. In *Advances in Neural Information Processing Systems*, pages 1597–1605, 2016.

[49] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the International Conference on World Wide Web*, pages 661–670, 2010.

[50] Shuai Li, Yasin Abbasi-Yadkori, Branislav Kveton, S Muthukrishnan, Vishwa Vinay, and Zheng Wen. Offline evaluation of ranking policies with click models. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1685–1694, 2018.

[51] Shuai Li, Baoxiang Wang, Shengyu Zhang, and Wei Chen. Contextual combinatorial cascading bandits. In *Proceedings of the International Conference on Machine Learning*, pages 1245–1253, 2016.

[52] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[53] Feifei Lin, Xu He, and Bo An. Context-aware multi-agent coordination with loose couplings and repeated interaction. In *Proceedings of the International Conference on Distributed Artificial Intelligence*, pages 103–125, 2020.

[54] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning*, pages 157–163. Elsevier, 1994.

[55] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and trends in information retrieval*, 3(3):225–331, 2009.

[56] Weiwen Liu, Shuai Li, and Shengyu Zhang. Contextual dependent click bandit algorithm for web recommendation. In *International Computing and Combinatorics Conference*, pages 39–50, 2018.

[57] Yong Liu, Yinan Zhang, Qiong Wu, Chunyan Miao, Lizhen Cui, Binqiang Zhao, Yin Zhao, and Lu Guan. Diversity-promoting deep reinforcement learning for interactive recommendation. *arXiv preprint arXiv:1903.07826*, 2019.

[58] Robert Loftin, Bei Peng, James MacGlashan, Michael L Littman, Matthew E Taylor, Jeff Huang, and David L Roberts. Learning behaviors via human-delivered discrete feedback: Modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-agent Systems*, 30(1):30–59, 2016.

[59] Robert Tyler Loftin, James MacGlashan, Bei Peng, Matthew E Taylor, Michael L Littman, Jeff Huang, and David L Roberts. A strategy-aware technique for learning behaviors from discrete human feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 937–943, 2014.

[60] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[61] James MacGlashan, Mark K. Ho, Robert Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E. Taylor, and Michael L. Littman. Interactive learning from policy-dependent human feedback. In *Proceedings of the International Conferences on Machine Learning*, pages 2285–2294, 2017.

[62] James MacGlashan, Michael Littman, Robert Loftin, Bei Peng, David Roberts, and Matthew Taylor. Training an agent to ground commands with reward and punishment. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[63] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[64] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

[65] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[66] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[67] Harrie Oosterhuis and Maarten de Rijke. Ranking for relevance and display preferences in complex presentation layouts. In *Proceedings of the International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 845–854, 2018.

[68] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. Value-aware recommendation based on reinforced profit maximization in e-commerce systems. *arXiv preprint arXiv:1902.00851*, 2019.

[69] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[70] Rodrigo Pérez-Dattari, Carlos Celemin, Javier Ruiz-del Solar, and Jens Kober. Interactive learning with corrective feedback for policies based on deep neural networks. *arXiv preprint arXiv:1810.00466*, 2018.

[71] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[72] Lijing Qin, Shouyuan Chen, and Xiaoyan Zhu. Contextual combinatorial bandit and its application on diversified online recommendation. In *Proceedings of the SIAM International Conference on Data Mining*, pages 461–469, 2014.

[73] Wei Qiu, Xinrun Wang, Runsheng Yu, Xu He, Rundong Wang, Bo An, Svetlana Obraztsova, and Zinovi Rabinovich. Rmix: Learning risk-sensitive policies for cooperative reinforcement learning agents. *arXiv preprint arXiv:2102.08159*, 2021.

[74] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the International Conference on Machine Learning*, pages 784–791, 2008.

[75] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

[76] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[77] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the European Conference on Machine Learning*, pages 317–328, 2005.

[78] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[79] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the International conference on machine learning*, pages 1889–1897, 2015.

[80] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[81] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4902–4909, 2019.

[82] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[83] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

[84] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[85] Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold, Yori Zwols, Daniel Visentin, and Ben Coppin. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. *arXiv preprint arXiv:1512.01124*, 2015.

[86] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[87] Ryuichi Takanobu, Tao Zhuang, Minlie Huang, Jun Feng, Haihong Tang, and Bo Zheng. Aggregating e-commerce search results from heterogeneous sources via hierarchical reinforcement learning. In *Proceedings of the World Wide Web Conference*, pages 1771–1781, 2019.

[88] Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 2139–2148, 2016.

[89] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2094–2100, 2016.

[90] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, pages 1096–1103, 2008.

[91] Thomas J Walsh, István Szita, Carlos Diuk, and Michael L Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 591–598, 2009.

[92] Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. Learning efficient multi-agent communication: An information bottleneck approach. In *Proceedings of the International Conference on Machine Learning*, pages 9908–9918, 2020.

[93] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. A theoretical analysis of NDCG type ranking measures. In *Proceedings of the Conference on Learning Theory*, pages 25–54, 2013.

[94] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016.

[95] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. *arXiv preprint arXiv:1709.10163*, 2017.

[96] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[97] Zheng Wen, Branislav Kveton, and Azin Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In *Proceedings of the International Conference on Machine Learning*, pages 1113–1122, 2015.

[98] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[99] Runsheng Yu, Yu Gong, Xu He, Bo An, Yu Zhu, Qingwen Liu, and Wenwu Ou. Personalized adaptive meta learning for cold-start user preference prediction. *arXiv preprint arXiv:2012.11842*, 2020.

[100] Mengchen Zhao, Zhao Li, Bo An, Haifeng Lu, Yifan Yang, and Chen Chu. Impression allocation for combating fraud in e-commerce via deep reinforcement learning with action norm penalty. In *Proceedings of the International Joint Conferences on Artificial Intelligence Organization*, pages 3940–3946, 2018.

[101] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. Deep reinforcement learning for search, recommendation, and online advertising: a survey. *ACM SIGWEB Newsletter*, Spring:1–15, 2019.

[102] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. *arXiv preprint arXiv:1805.02343*, 2018.

[103] Xiangyu Zhao, Long Xia, Yihong Zhao, Dawei Yin, and Jiliang Tang. Model-based reinforcement learning for whole-chain recommendations. *arXiv preprint arXiv:1902.03987*, 2019.

[104] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. *arXiv preprint arXiv:1802.06501*, 2018.

[105] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the World Wide Web Conference*, pages 167–176, 2018.

[106] Masrour Zoghi, Tomas Tunys, Mohammad Ghavamzadeh, Branislav Kveton, Csaba Szepesvari, and Zheng Wen. Online learning to rank in stochastic click models. In *Proceedings of the International Conference on Machine Learning*, pages 4199–4208, 2017.

[107] Shi Zong, Hao Ni, Kenny Sung, Nan Rosemary Ke, Zheng Wen, and Branislav Kveton. Cascading bandits for large-scale recommendation problems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 835–844, 2016.

[108] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. Reinforcement learning to optimize long-term user engagement in recommender systems. *arXiv preprint arXiv:1902.05570*, 2019.