Safe Coupled Deep Q-Learning for Recommendation Systems

Runsheng Yu, Yu Gong, Rundong Wang, Bo An, Qingwen Liu, Wenwu Ou
runshengyu@gmail.com

Abstract

Reinforcement Learning (RL) is one of the prevailing approaches to optimize long-term user engagement in Recommendation Systems (RS). However, the well-known exploration strategies of RL (e.g., the ϵ-greedy strategy) encourage agents to interact and explore the environment freely, which may recommend unpleasant items to the users frequently, violating their preferences and making them lose confidence in the RS platform. To avoid such irrelevant and unpleasant recommendations, we propose a novel safe RL approach to maximize accumulated long-term reward under the safety guarantee. Our contributions are three-fold. Firstly, we introduce a novel training scheme with two value functions to maximize the accumulated long-term reward under the safety constraint. Secondly, we theoretically show that our methods are able to converge and maintain safety with a high probability during the training process. Thirdly, we implement two practical methods, including a Simhash-based method as well as a relaxation method for large-scale environments. Experiments on immediate recommendation, sequential recommendations, as well as safe gridworld reveal that our methods outperform the state-of-the-arts dramatically.

1 Introduction

Recommendation Systems (RS) assist people to seek products or news they prefer efficiently (Ricci, Rokach, and Shapira 2011). With a focus on optimizing the long-term feedback (e.g., click-through rate and conversion rate (Elahi, Ricci, and Rubens 2016)), the online training scheme based Reinforcement Learning (RL) has demonstrated impressive performance in various long-term user engagement environments (Zhao et al. 2019; Zou et al. 2019; Zhao et al. 2018a; Shani, Heckerman, and Brafman 2005; Chen et al. 2019c,a).

However, due to the nature of RL, i.e., trial and error, standard RL methods interact and explore freely within environments to learn an (near) optimal policy. However, this scheme incurs a key challenge in the RS scenarios. That is, under the online training scheme, free exploration will recommend irrelevant items to users, making them lose confidence in the RS platform and causing damage to both the RS platform and the users. For example, assuming the RS agent is interacting with a vegetarian, if it always recommends meat to that user, he will be annoyed and leave. Therefore, it is indispensable to introduce a strategy to avoid unsafe actions.

A straight-forward approach is to mask all the unsafe actions. Like our previous example, for a vegetarian user, he may not be angry if the agent avoids recommending meat items. However, for an unknown user, it is not an easy task to know which items should be masked in advance if the agent knows nothing about the user. We also notice that two important properties in RS (Ricci, Rokach, and Shapira 2011): 1) users can tolerate bad items if they are not recommended too many times and 2) some users' behaviors can be inferred by the behavior of similar users. Therefore, we aim to design a learning-based method to guarantee safety by finding which actions should be masked through interacting with different users.

Safe RL is a promising method to prevent unsafe exploration. These approaches can be categorized into two types: model-free methods and model-based (safe state) methods. For the former, the environments are modeled as Constrained Markov Decision Processes (CMDP) with the constraint that the expected accumulative cost cannot exceed some predefined constant and CMDP is solved by RL based optimization methods (e.g., the Lagrangian multipliers policy/value iteration) (Yu et al. 2019; Achiam et al. 2017; Chow et al. 2017). For the latter, the constraint is based on states (the agent must be in a safe state), and uncertainty modeling methods (e.g., the Gaussian process or the Lyapunov function) are leveraged to estimate the transition function with uncertainty for the next states (Polymenakos, Abate, and Roberts 2019; Berkenkamp et al. 2017).

However, neither of these methods are appropriate for the RS scenarios. For *model-based methods*, they focus on state safety, e.g., whether the quadrotor hurts when committing actions to a certain state (Berkenkamp et al. 2017). Differently, in real-world RS, an agent is allowed to commit few unsafe actions so long as the value of the accumulated cost is not lower than a given threshold and thus the safety of the trajectory rather than safety of a state is defined as the safety criteria. Besides, these methods have large time complexity (Berkenkamp, Schoellig, and Krause 2016), being computationally infeasible for large-scale environments like sequential RS. For *model-free methods*, although these

Preprint.

methods model the environment with CMDP, suiting the RS scenarios well, most of them still suffer from unsafety during the training process, i.e., safety may only be approximately guaranteed after a sufficient long learning period (Cheng et al. 2019). As a result, users may leave in advance before they find the safe strategies. Therefore, designing a training scheme that both consider safety for the trajectory and avoid unsafe actions as many times as possible during the entire training process is necessary.

In this paper, to optimize the long-term user engagement with the safety constraint, we propose the Safe Coupled Deep Q Network (SC-DQN). Our main contributions are as follows. Firstly, we explore a novel safe RL approach containing two value functions to maximize the accumulated long-term reward under the safety constraint. Secondly, we theoretically show the convergence and safety properties of our methods with mild assumptions in both tabular case and specified function space. Thirdly, for large-scale environments, we implement two practical deep learning based RL approaches to accelerate training and reducing the overestimated bias, including a Simhash-based method and a relaxation method. Experiments on immediate recommendation, sequential interactive recommendations (or sequential recommendations for short), and safe gridworlds reveal that our methods outperform the state-of-the-arts dramatically.

2 Related Works

Our work is related to safe RL as well as RL based RS. Safe RL aims to avoid choosing actions that may be dangerous. Previous studies can be mainly categorised into two tracks: model-based ones and model-free ones. For the former, the main idea is to learn the dynamics of the environments to avoid actions which cause unsafe states (Berkenkamp et al. 2017; Turchetta, Berkenkamp, and Krause 2016; Polymenakos, Abate, and Roberts 2019; Akifumi Wachi 2020). However, these approaches are computationally expensive to learn when the environment is complicated (e.g., the state space is large). Moreover, the definition of safety in model-based RL may not suit RS well as we mentioned above. For the model-free methods, one of the primarily used model-free approaches is Lagrangian multiplier based methods, which convert the optimization problem (e.g., linear programming) into the Lagrangian multipliers form (Yu et al. 2019; Achiam et al. 2017; Chow et al. 2017). However, most of these methods do not guarantee safety during the learning procedure. Another approach is the Lyapunov-based methods, which construct the Lyapunov functions to guarantee global safety of a behavior policy by solving a Linear Program (LP) at every step. However, this method is computationally expensive since it needs to solve LP at each step (Chow et al. 2018). Different from the methods mentioned above, our method is approximately safe with high probability during the whole training process.

The main ideas of RL-based RS regard RS as (partially observable) MDP and apply RL and its variants to find the optimal strategies with interactions (Zou et al. 2019; Zhao et al. 2018a; Shani, Heckerman, and Brafman 2005; Chen et al. 2019c,a). However, as we mentioned above, due to ignoring the safety, these methods might recommend irrelevant item to harm an RS system in the long run. An alternative approach to address the challenge is the off-policy training (training from logged data) (Zhao et al. 2018b; Chen et al. 2019b; Xin et al. 2020), focusing on training the agents only with log data. Unfortunately, these off-policy training schemes suffer from high variance problems seriously (Munos et al. 2016). An approach similar to our method is Reward Constrained Recommendation (RCR), intending to model the RS through CMDP (Zhang et al. 2019). However, they focus on the constraint of the natural language feedback. Moreover, like other RL methods, RCR is also not a safe method during the training stage. To our best knowledge, we are the *first* to introduce safe RL into the RS, considering finding the optimal strategy under the safety constraint during the training process.

3 Problem Formulation

This section gives an example to discuss why safety is crucial in RS, and formally describes the Constrained Markov Decision Process (CMDP) for the RS with the accumulated cost function.

3.1 An Example for Safe Recommendation

Taking the sequential recommendation as an example, as shown in Fig. 1, the RS agent first receives a user's profile and browsing history, and selects an item from the item list (one out of three in this example). The user will give feedback (click, view or leave) to the agent according to the recommended item. This process will repeat until the user leaves the platform or the agent has recommended all the items, which can be formulated as a Markov Decision Process (MDP). Also, notice that if the agent recommends irrelevant items too many times (like the peaked cap in Fig. 1), the user may lose interest and leave the platform. Therefore, it motivates us to add safety constraint into the model to avoid recommending irrelevant items (unsafe actions) too many times as CMDP.

3.2 Constrained Markov Decision Process for RS

Formally, a feed stream (the interaction between RS and user) can be formulated as an MDP, defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma \rangle$, where state $s \in \mathcal{S}$ is the user information (including but not limited to the user profile and user browsing history), action $a \in \mathcal{A}$ is the item recommended by the RS in the t-th interaction , $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, R_{\max}]$ is the reward function (e.g., reward is positive when user clicks an item. $R_{\max}$ is the highest reward value), $P : S \times A \times S \to [0, 1]$ is the transition function which indicates the probability of meeting the next state s' after taking action a_t in state s_t and γ is the discount rate. The objective of $\mathcal{M}$ is to find a policy π that maximizes the accumulated reward $J^\pi = \max \mathbb{E}^\pi \left\{ \sum_{t=0}^{T} \gamma^t r_t | s_0 = s \right\}$. We introduce the safety constraint into MDP as the CMDP, i.e., we augment the tuple $\mathcal{M}$ with cost set $\mathcal{C}$ and further define the expected cost in terms of policy π under state s: $\underline{J}^\pi = \mathbb{E}^\pi \left\{ \sum_{t=0}^{T} \bar{\gamma}^t c_t | s \right\}$, where $c_t : \mathcal{S} \times \mathcal{A} \to [C_{\min}, C_{\max}]$ is the cost function[1]

[1]The cost is always negative in this paper, i.e., $C_{\max} \leq 0$. The more risk action the agent takes, the lower the cost value is.

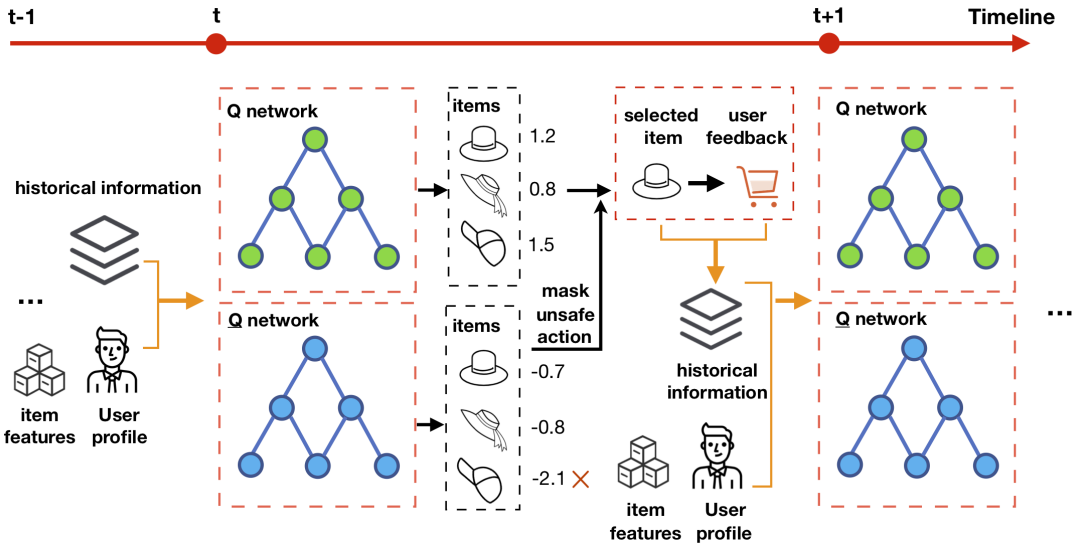


Figure 1: The framework of SC-DQN in sequential recommendation. In each time step t, the input of the Q and $\underline{Q}$ networks include the user profile (user features), the historical information (browsing history), as well as the item features. The output contains the values and the safe values of each item. The recommended item is selected according to π and the user gives feedback to the agent.

(e.g., cost is negative when user views without purchasing or the user leaves the platform). at step t and $\bar{\gamma} \in (0, 1)$ is the discount rate for the cost[2]. Similar to (Chow et al. 2018; Yu et al. 2019), the safe policy set can be formulated as $\Omega_{\text{safe}} = \{\pi | \underline{J}^{\pi}(s_0) \geq \epsilon\}$, where s_0 is the initial state, and the unsafe set is defined as $\Omega_{\text{unsafe}} = \Omega / \Omega_{\text{safe}}$, where Ω is the policy space.

4 Safe Coupled Q-Learning

This section discusses the idea of Safe Coupled Q-Learning. We first introduce the safe strategy and the safe coupled value iteration to obtain the near-optimal safe strategy. Then, to further reduce unsafe exploration, we design the safe coupled deep Q-Learning to make decision based on similar states. Finally, we introduce the bias elimination technique to prevent unsafe strategies during the training process.

4.1 Safe Strategy

We start with the main idea of the standard Q-learning (Sutton and Barto 2018): the policy is designed to choose the action that induces the highest Q-value: $\arg\max_a Q(s^i, a)$ in state s^i. Obviously, the standard Q-learning may choose an unsafe action. Intuitively, if there exists a stepwise oracle function mapping each action a to the expected accumulated cost under π: $\underline{J}^{\pi}$, the agent can easily mask these actions to guarantee safety. Based on this idea, we define $\underline{Q}^{\pi}(s, a)$ as an approximation of $\underline{J}^{\pi}$[3] and since $\mathbb{E}^{\pi}\left\{\sum_{t=0}^{T} \bar{\gamma}^t c_t\right\} \approx \mathbb{E}^{\pi}\left\{\tilde{C} + \underline{Q}\right\}$, when $\underline{Q}$ is accurate enough, $\tilde{C} + \underline{Q} \geq \epsilon$ can be used as the approximated safety criteria, where $\tilde{C} = \sum_{t=0}^{\hat{t}} \gamma^t c_t$ is the accumulated cost at time step $\hat{t}$. Then, we set the policy as $\tilde{\pi}(a|s^i) := \arg\max_{a \in \tilde{A}} Q(s^i, a)$, where $\tilde{A}(s^i) = \{a | \tilde{C} + \underline{Q}(s^i, a) \geq \epsilon\}$. Assuming $Q > 0$ (this can be easily done by reward shaping), we can further simplify the policy as $\tilde{\pi}(a|s^i) = \arg\max_a [Q(s^i, a) \mathbb{1}(\tilde{C} + \underline{Q}(s^i, a) \geq \epsilon)]$, where $\mathbb{1}(\cdot)$ is an indicator function to select the actions that avoid unsafety. Specially, when $\forall a, \mathbb{1}(\tilde{C} + \underline{Q}(s^i, a) \geq \epsilon) = 0$, the policy will randomly choose an action since all the actions are unsafe.

4.2 Safe Coupled Value Iteration

Previous section proposes a possible safe strategy method under the assumption that Q and $\underline{Q}$ are (near-) optimal. In this subsection, we focus on how to get the near-optimal Q and $\underline{Q}$. Following the Value Iteration (VI) technique, we implement the Bellman equation approaches (Sutton and Barto 2018) to both Q and $\underline{Q}$. That is, for Q, the Bellman operator $\mathcal{T}$ is:

$$\mathcal{T}Q(s, a) := r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V^{\tilde{\pi}}(s') \right], \quad (1)$$

where $V(s') = \max_{a'} Q(s', a') [\mathbb{1}(\tilde{C} + \underline{Q}(s', a) \geq \epsilon)]$. Similarly, the Bellman operator $\underline{\mathcal{T}}$ for $\underline{Q}$ can be formulated as:

$$\underline{\mathcal{T}}\underline{Q}(s, a) := c(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\sum\nolimits_a P^{\underline{Q}}_{a's'} \underline{Q}(s', a') \right], \quad (2)$$

where $P^{\underline{Q}}_{a's'} := P(a' \mid s', \underline{Q})$ is the probability of taking each action a' at state s' under $\underline{Q}$ (e.g., it can be defined by the Boltzmann distribution $P^{\underline{Q}}_{a's'} := \frac{e^{\underline{Q}(s', a')}}{\sum_b e^{\underline{Q}(s', b)}}$). Eq. (2) helps to guarantee convergence because $\tilde{\pi}$ depends on both Q and $\underline{Q}$ (details can be found in Appendix B). Based on the

[2] It is possible to set $\bar{\gamma} = 1$ under the assumption of proper and stationary policies (Assumptions 7.2.1 in (Bertsekas 2015)) and with a little abuse of the notations, we hereinafter use γ to replace $\bar{\gamma}$ if the statement is clear.

[3] We omit the superscript π and variables s, a for $Q(s, a)$ and $\underline{Q}(s, a)$ if the statement is clear.

two Bellman equations, we can update Q and $\underline{Q}$ iteratively, i.e., $Q^{k+1}(s,a) = \mathcal{T}Q^k(s,a)$ and $\underline{Q}^{k+1}(s,a) = \underline{\mathcal{T}}\underline{Q}^k(s,a)$, where k is the number of iterations. Since this approach relies on the VI technique, we name it as the Safe Coupled Value Iteration (SCVI).

4.3 Safe Coupled Deep Q Network

However, SCVI suffers various challenges in real-world applications. Similar to all the tabular RL methods, when the state space is large, storing the complete Q-tabular is memory-consuming. Moreover, SCVI can still be unsafe because update the tabular requires committing actions among almost all the trajectories, including unsafe trajectories. To address the challenge, we abstract the states together to reduce the space memory and mitigate the meeting time of unsafe states, since similar states which are mapped together share similar safety property.

A straightforward way to carry out this abstraction is to learn to abstract the environment and make decisions end to end (the Neural Network (NN) based methods) (Gelada et al. 2019), named as the Safe Coupled Deep Q Network (SC-DQN). Different from the widely adopted NN structure which uses the raw matrix of the environment as input (Mnih et al. 2015), the input of NN in RS needs the embedding framework since it includes various discrete and continuous features. Following (He et al. 2017), we map the features into the continuous (embedding) states.

The loss functions of Q and $\underline{Q}$ are designed following the similar Temporal Difference (TD) loss technique in deep Q-learning (Mnih et al. 2015):

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{s,a,s'\sim D}\left[Q(s,a;\theta) - r(s,a) - \gamma Q\left(s', \tilde{\pi}(s');\theta\right)\right]^2,\\ \underline{\mathcal{L}} &= \mathbb{E}_{s,a,s'\sim D}\left[\underline{Q}(s,a;\underline{\theta}) - c(s,a) - \gamma \underline{Q}\left(s', \tilde{\pi}(s');\underline{\theta}\right)\right]^2.\end{aligned} \tag{3}$$

where θ and $\underline{\theta}$ are trainable parameters, D is the replay buffer, $\mathcal{L}$ and $\underline{\mathcal{L}}$ are the loss functions for $Q(s,a;\theta)$ and $\underline{Q}(s,a;\underline{\theta})$ respectively[4]. Now, Q^h and $\underline{Q}^h$ can be trained jointly.

4.4 Eliminating the bias

Despite the fact that SC-DQN can alleviate the occurrence of unsafe strategies, it is still hard to guarantee $\underline{Q}^h$ is close to $\underline{Q}^*$ during the whole training process. That is, assuming $\underline{Q}^h$ and $\underline{Q}^*$ are far from each other, the $\tilde{\pi}$ induced by $\underline{Q}^h$ is also biased, and thus the trajectories generated by $\tilde{\pi}$ cannot guarantee safety any longer. This bias problem seriously affects the safety guarantee in SC-DQN. To address this challenge, a bias estimation and elimination method is required.

We first formally define the bias in SC-DQN in a certain state s^i and action a^j as $\hat{\epsilon} = (\underline{Q}^h(s^i,a^j) - \underline{Q}^*(s^i,a^j))^2$. Intuitively, the bias can be interpreted as the squared Euclidean distance between the value of $\underline{Q}^*$ and $\underline{Q}^h$. Assuming in each iteration, $\underline{Q}^h$ is well-training within the dataset in D[5], we have an alternative form of value iteration:

$$\begin{aligned}&\underline{\mathcal{L}}_D\left(\underline{Q}^{h'}, \underline{Q}^h\right) = \tfrac{1}{|D|}\textstyle\sum_{(s,a,r,s')\in D}\Big[\underline{Q}^{h'}(s,a) - \\ &c(s,a) - \gamma \max_{a'} \underline{Q}^h\left(s', a'\right)\Big]^2,\\ &\underline{Q}^h_{k+1} = \underline{\mathcal{T}}\underline{Q}^h_k := \arg\min_{\underline{Q}^{h'}} \underline{\mathcal{L}}_D\left(\underline{Q}^{h'}, \underline{Q}^h_k\right),\end{aligned}$$

where k is the number of iterations. The equation above indicates that each update can always be optimal w.r.t. the samples in dataset D (i.e., the updating is sufficient). Then, using the similar technique from the concentration inequalities (Sridharan 2002), we have the upper bound for $\hat{\epsilon}$:

Theorem 1. *For state s^i, the distance between the estimated abstracted Q-value $\underline{Q}^h(s^i,a)$ and the optimal $\underline{Q}^*(s^i,a)$ are bound by*

$$\begin{aligned}&\left(\underline{Q}^h(s^i,a) - \underline{Q}^*(s^i,a)\right)^2 \le \frac{2|C_0|}{1-\gamma}\sqrt{\frac{1}{2(n\wedge 1)}\log\frac{2\gamma|A|}{\delta}}\\ &+ \frac{(\sqrt{2}C_{\max})^2}{(1-\gamma)^2} + \left(\frac{C_{\max}}{1-\gamma} - Q^h_k(s,a)\right)^2,\end{aligned}$$

with probability at least $1-\delta$, where $C_0 = C_{\max} - C_{\min}$ and $n \wedge 1 = \max\{n, 1\}$.

Due to space limitation, the details of the proofs can be found in Appendix A. For simplification, we define $\sigma^2 := \frac{2|C_0|}{1-\gamma}\sqrt{\frac{1}{2(n\wedge 1)}\log\frac{2\gamma|A|}{\delta}} + \frac{(\sqrt{2}C_{\max})^2}{(1-\gamma)^2} + \left(\frac{C_{\max}}{1-\gamma} - Q^h_k(s,a)\right)^2$ and we have $\underline{Q}^*(s^i,a) \ge \underline{Q}^h(s^i,a) - \sigma$ with probability at least $1-\delta$. Then, we can guarantee safety by taking those actions satisfying the safety constraint: $\underline{Q}^h(s^i,a) - \sigma \ge \epsilon$, and the safe policy $\tilde{\pi}$ can be similarly modified as $\pi(a|s) := \arg\max_a[Q(s^i,a)(\mathbb{1}(\tilde{C} + \underline{Q}(s^i,a) - \sigma \ge \epsilon))]$. The pseudo-code can be found in Alg. 1. To balance the exploration and exploitation trade-off in practice, we multiply a positive constant ϕ (i.e., $\frac{\phi|C_0|}{1-\gamma}\sqrt{\frac{1}{2(n\wedge 1)}\log\frac{2\gamma|A|}{\delta}}$) to reduce the count-based module.

Notice that s^i can also be an abstracted state (Lemma 3 in Appendix A) which can speed up the training process when the original state space is large.

Algorithm 1: Safe Coupled Deep Q-Learning

Result: the Well-trained parameters θ and $\underline{\theta}$

Initialize Q and $\underline{Q}$ with parameters θ & $\underline{\theta}$ and the experience buffer D.;
for *episode* τ **do**
  for *each step t* **do**
    get s_t, take action $a_t = \pi(a|s_t)$, and reward r_t and s_{t+1}. ;
    store tuple $\langle s_t, a_t, r_t, c_t, s_{t+1}\rangle$ in D.;
  end
  sample a batch of $\langle s, a, c, r, s^{'}\rangle$ from D.;
  train θ and $\underline{\theta}$ by minimizing $\mathcal{L}$ and $\underline{\mathcal{L}}$. ;
end

[4]For simplification, we will use Q^h and $\underline{Q}^h$ as the symbols of $Q(s,a;\theta)$ and $\underline{Q}(s,a;\underline{\theta})$ if no special mentioned.

[5]This can be done by larger batch size and longer training time in each training step.

5 Theoretical Analysis

This section shows the convergence analysis as well as the safety analysis. We first prove that our methods are always safe with a high probability:

Theorem 2. *If $\Omega_{safe} \neq \emptyset$, the statement $\pi \cap \Omega_{unsafe} = \emptyset$ is established with probability at least $1 - \delta$.*

Theorem 2 reveals that if the environment itself has at least one safety policy ($\Omega_{safe} \neq \emptyset$)[6], then the policy π, with a high probability, is always safe ($\pi \cap \Omega_{unsafe} = \emptyset$). Specially, when the agent is well-trained ($\pi = \pi^*$), $\pi \cap \Omega_{unsafe} = \emptyset$ is almost surely ($\delta = 0$).

Now, we focus on the convergence analysis when Q and $\underline{Q}$ are in function space. To facilitate our analysis, we define an operator to update the Q function:

$$\begin{aligned} &\mathcal{L}_D\left(Q^{h'}, Q^h\right) = \tfrac{1}{|D|}\textstyle\sum_{(s,a,r,s')\in D} |Q^{h'}(s,a) - c(s,a) \\ &-\gamma \max_{a'} Q^h(s',a')|, \\ &Q^h_{k+1} = \tau Q^h_k := \arg\min_{Q^{h'}} \mathcal{L}_D\left(Q^{h'}, Q^h_k\right). \end{aligned} \tag{4}$$

Eq. (3) is a method to find the optimal Q value minimizing the TD error w.r.t. the samples in the dataset (replay memory) D, while Eq. (4) indicates the optimal function to find the Q. Since the former is an approximation to the latter if enough gradient descent is taken at each interaction, we use the latter to analyze our method.

Based on the simplification, we give the convergence analysis for our method in a specified function space:

Theorem 3. *For a stationary policy, assuming that the neural networks based Q satisfies two conditions: (1) the feature extraction layers are fixed. (2) The decision layer is a linear mapping. When Q is initialized by zero, i.e., $Q^h_1 = 0$, we always have:*

$$\begin{aligned} \left\| Q^h_k(s,a) - Q^*(s,a)\right\|_\infty \leq \gamma^{k-1}\frac{R_{\max}}{1-\gamma} + 2^{(4N+7)}e^{(N+2)}\times \\ (N+2)\exp(LaW(\frac{4b^2 2^{(-8N-8)}e^{(-2N-4)}\log\delta}{nd(N^2+4N+5)})/2), \end{aligned}$$

with probability at least $(1-\delta)^2$, where N is the dimension of embedding.

$LaW(\cdot)$ is the Lambert W function. Notice that when$\lim_{n\to\infty,k\to\infty} \left\| Q^h_k(s,a) - Q^*(s,a)\right\|_\infty = 0$. This result reveals that Q coverages to Q^*.

Notice that our analysis is based on a specific neural network structure, which can be implemented easily to other structures.

Moreover, our methods can be applied to the immediate recommendation (stochastic Multi-Armed Bandit, MAB) environment with slight modification. The complete theoretical analysis and the pseudo-code can be found in Appendix B and C respectively.

[6]This also tick out those environments that are impossible to be safe.

6 Practical Implementations

In practice, it is still hard to implement the safe policy π since it includes the visiting times of each state: $1 \wedge n$, which changes slowly for most states in large-scale environment. This result induces the overestimation of σ and further causes all safe actions to be banned totally. We name it as the overestimated bias problem, and two practical approaches are introduced to address the challenges.

6.1 Simhash-based State-action Counting

Inspired by (Tang et al. 2017), we use the Simhash method to map similar states together to reduce the state space. Simhash is a method for quickly estimating how similar two states are, and the hash function is defined as $\phi(s) := \mathrm{sgn}(Gg(s)) \in \{-1,1\}^k$, where G is a $k \times D$ matrix with i.i.d. entries drawn from a standard Gaussian distribution $\mathcal{N}(0,1)$, $\mathrm{sgn}(\cdot)$ is the step function and $g : S \times A \to \mathbb{R}^D$ is a prepossessing function. Notice that in Simhash, a well-known equation is $\mathbf{P}[\phi(s_1) = \phi(s_2)] = \mathrm{sim}(s_1, s_2)$, where sim is similarity function (Charikar 2002). Thus, applying this conclusion we can immediately get with probability at least $(1-\delta)\min_{s_1,s_2}\mathrm{sim}(s_1,s_2)$ that safety is guaranteed. We name this methods as Simhash SCDQN (S2CDQN).

6.2 Relaxation of the Safety Bound

However, the count-based method is both time and memory consuming since it requires to map the states into hash values, store the hash table, and count the visiting times of the abstracted states. To make our method more sample efficient and memory saving, a relaxation approach is leveraged to further reduce the training time.

We approximate σ^2 as: $\sigma^2 \approx \frac{(\sqrt{2}C_{\max})^2}{(1-\gamma)^2} + (\frac{C_{\max}}{1-\gamma} - Q^h_k(s,a))^2$. Now, σ is irrelevant to the term $1 \wedge n$. This relaxation helps diminish both the memory space and time since it no longer requires to store and search the visited states. This method is named as Relaxed SC-DQN (RSC-DQN).

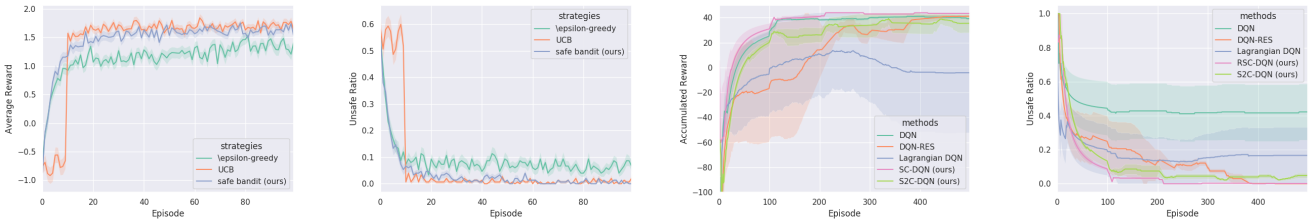
Both S2C-DQN and RSC-DQN have their own advantages. the former's strategy is more risk-sensitive than the latter, while the latter is more time and memory saving than the former.

7 Experiments

We conduct four experiments to evaluate the performance of our proposed methods, including immediate recommendation, sequential recommendation I and II, and safe gridworld.

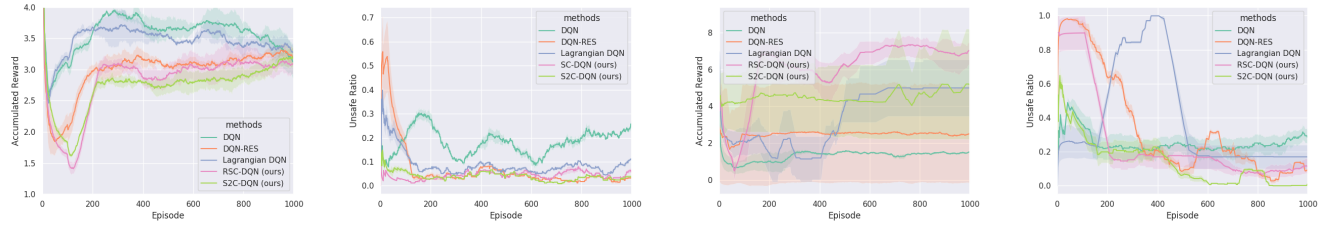
7.1 Immediate Recommendation

*Environment setup.*We firstly assess the performance of our methods within a special case of CMDP: the immediate recommendation, one of the widely adopted models for single-step recommendation (Li et al. 2010). The baselines include two standard policies: the ϵ-greedy strategy as well as the UCB strategy (Slivkins 2019). We build the environment based on the *synthetic data*: the total number of items (arms) is 10 and each item obeys the Gaussian distribution with the same variance 0.7 and different means ($-3 + 0.5 * N$,



(a) Reward in IR (b) Unsafe Ratio in IR. (c) Accumulated Reward in SG. (d) Unsafe Ratio in SG.

Figure 2: The reward and the unsafe ratio in Immediate Recommendation (IR) and Safe Gridworld (SG).



(a) Accumulated Reward in RS I. (b) Unsafe Ratio in RS I. (c) Accumulated Reward in RS II. (d) Unsafe Ratio in RS II.

Figure 3: The accumulated reward and unsafe number in the Sequential Recommendations I and II.

	ϵ-greedy	UCB	Ours
UA $\downarrow$	1317$\pm$0.080	954$\pm$0.059	**554**$\pm$0.035

Table 1: The total number of unsafe actions in immediate recommendation. UA means unsafe action. The best results are highlighted in **bold**. The mean and standard deviation are reported by 100 independent trials.

$N \in \{0, 1, \ldots, 9\}$). The training time is 150 with 100 trials. The cost is $c = \mathbb{1}(r \leq \epsilon) * r$, where $\epsilon = 0$. We use two metrics to evaluate the performance of the methods mentioned above: the accumulated reward and the unsafe ratio ($\frac{\sum_{i=1}^{n} \mathbb{1}(\sum_j c_j < \epsilon)}{n}$). All the details of metrics and hyper-parameters can be found in Appendix E.

Results. As shown in Fig.2(a) and 2(b), our methods outperform the baselines with less unsafe ratio and higher convergence rate. Moreover, Tab. 1 reveals that our methods can prevent the unsafe action dramatically than other baselines in the MAB environment, with nearly half number of the unsafe actions (554) than that of the second-best method (954).

7.2 Sequential Recommendation I

Environment setup. We also conduct experiments on sequential recommendation, which recommends an item (e.g., a movie) and receives a reward (the score of that movie) from the user. We choose the MovieLens-1M as the dataset and simulate the user behavior using the neural networks[7]. The user chooses an integer from $[1, 5]$ as the score of that movie and the reward is set as 1 when score > 3. To encourage diversity in recommendation, we further penalize the repeated recommended items. The reward is $r_t = 1 * \mathbb{1}([\text{score} > 3] \wedge [a \notin \tilde{A}_t])$ and the cost is $c_t = -1 * \mathbb{1}(\text{score} \leq 3) - 0.5 * \mathbb{1}(a \in \tilde{A}_t)$, where $\tilde{A} = \langle a'_t \rangle_{t'=0}^{t-1}$ is the augmented set of actions from step 0 to $t-1$. The threshold ϵ is set to -1. We randomly choose $1,000$ users for the online training process. Details of the hyper-parameters, the formal definition of metrics, and the simulators are revealed in Appendix E.

Baselines. We compare S2C-DQN and RSC-DQN with several state-of-the-arts baselines. (1) *DQN*. This is the standard DQN without any cost[8]. (2) *DQN-RES*. DQN-RES is similar to DQN except that we add the cost implicitly into the reward by the REward Shaping (RES) technique, where shaping reward is defined as $r_t^s = r_t + c_t$. (3) *LDQN* (Chow et al. 2015; Zhang et al. 2019). The Lagrangian-based DQN (LDQN) is known as one of the scalable safe RL methods in large-scale CMDP and we use it as the standard safe RL baseline[9].

Results. From Fig. 3(b), except for DQN, all the methods can have low average unsafe ratio (about 0.1), while our methods have low unsafe ratio at the first 50 episodes (less than 0.1), comparing with DQN-RES and LDQN, which have about $0.4 - 0.5$ unsafe ratio. From Fig. 3(a), our methods, together with DQN-RES do not have a higher average reward than other SOTAs, revealing the fact that there is a trade-off between safe and high scores. Our methods guarantee safety and thus its convergence rate might not be as high as DQN, which has the highest rate but fails to be safe. Also, Tab. 2 indicates that our methods achieve the best and second-best precision value but are not good enough in CR metric. This

[7] We simulate the online learning process to avoid the pricey cost, and potential risk in the real-world online A/B test.

[8] To stabilize and accelerate training, we use double DQN with priorier replay buffer for all the methods in this paper.

[9] For completeness, we introduce the LDQN in appendix E.

may be caused by the fact that our methods prefer to choose conservative actions to guarantee safety.

7.3 Sequential Recommendation II

Environment setup. We further certify our method in another real-world sequential recommendation environment. We firstly collect 10,000 users with different log data from one of the largest ecommerce platforms in the world, including item features and users' feedback (skipping, leaving, and viewing). Similar to Section 7.2, we build the online simulator for users by learning the user behavior from log data. Due to the imbalanced distribution in users' feedback, $\frac{1}{64}$ is chosen as the resampling ratio. For the environment, we model the states as an augmented browsing history, including the previous user feedback, the items' features, and the current step. The reward is defined as: $r_t = 1 * \mathbb{1}([\text{score} = click]) - 0.1 * \mathbb{1}(a \in \tilde{A}_t)$, while the cost is $c_t = -5 * \mathbb{1}([\text{score} = leave]) - 0.1 * \mathbb{1}(a \in \tilde{A}_t)$. We set the $\epsilon = -3$. The baselines and hyper-parameter settings are the same as those in Section 7.2. We randomly choose $1,000$ users for the online training.

Results. Firstly, from Fig. 3(c) and 3(d), we can find that our methods outperform SOTA in both accumulated reward and unsafe ratio. Notice that though S2C-DQN has relatively low UN, it does not achieve high reward score. On the contrary, RSC-DQN has high reward score but its UF is not low enough. This forms a trade-off between safety and reward. Secondly, Tab. 2 reveals that our methods are also well-performing w.r.t click ratio and browsing history, indicating that our methods can help the RS system to satisfy the users' requirements in the long run.

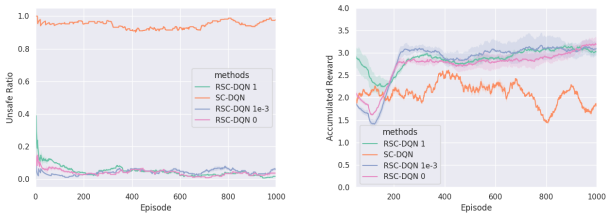
7.4 Safe Gridworld

Environment setup. Finally, to illustrate that our method can be flexibly extended to other safe environments, we conduct an experiment on the AI safety gridworld (Leike et al. 2017), which includes various environments to testify the performance of RL methods. Here, we choose the distributional shift I as the environment, where the agent needs to find the goal without touching lava, as shown in Appendix E. The baselines are those methods similar to the sequential RS. The rewards are -1 and 50 for taking a step, and reaching the goal respectively, while the cost is c is -50 when falling into the lava. The threshold ϵ is manually set to -20. The parameter settings and network structures are similar to the sequential recommendation environment except that we do not use the embedding layers since the safety grid-world environment does not include too many features.

Results. Fig. 2(d) and 2(c) indicate that our method (especially RSC-DQN) can achieve both completed safety (the unsafe ratio is zero) and find near-optimal strategy fastest among other SOTAs. One interesting phenomenon is that comparing with DQN, both our methods have similar convergence rate w.r.t total rewards but have significantly lower unsafe ratio, while for DQN-RES, our methods have similar convergence rate w.r.t unsafe ratio but converge much faster w.r.t total rewards. This indicates that our methods have advantage in finding optimal results and keeping safety among other SOTAs.

Environment	SR I		SR II	
Metrics	HR↑	CR ↑	CR↑	BH↑
DQN	**0.357**	0.919	0.179	7.269
DQN-RES	0.299	0.959	0.338	6.036
Lagrangian DQN	0.347	0.955	0.538	5.787
RSC-DQN (ours)	0.281	**0.976**	**0.699**	7.779
S2C-DQN (ours)	0.273	0.974	0.577	**7.921**

Table 2: The performance comparison of different methods. The best results are highlighted in **bold**. ↑ means higher is better. The mean and standard deviation are reported by 3 independent trials.



(a) Unsafe Ratio. (b) Accumulated Reward.

Figure 4: The ablation studies of unsafe ratio and accumulated reward in sequential RS I.

7.5 Extra experiments.

We also conduct extra experiments to evaluate our methods in various perspectives, including (1) different metrics for each user; (2) ablations; and (3) training time. For (1), it reveals that our method can converge faster than previous methods in most cases in terms of different metrics. For (2), as shown in Figs. 4(a) and 4(b), an ablation study indicates that ϕ does play an important role in agent's performance. Too large ϕ masks some accessible safe actions, too small ϕ leads to unsafety. Moreover, regarding vanilla SC-DQN, it performs the worst among all the methods. This is caused by the fact that in large-state environment, σ is always overestimated and most of the safe actions are banned, which is consistent with our conjecture. For (3), the training time coincides with our analysis that RSC-DQN is more time-consuming than S2C-DQN (2.4h vs 3.9h in GPU hours). More details can be found in Appendix E.

8 Conclusion

This paper proposes a novel safe RL method aiming to handle the challenge of irrelevant recommendation derived from free exploration. Our main contributions are three-fold. Firstly, we are the first to introduce safe RL for RS by designing two value functions to choose the actions that maximize the total rewards with safety constraints. Secondly, we show that our methods are able to converge and stay safe with a high probability during training. Thirdly, with a focus on the large-scale environments, we propose two practical methods, including a Simhash-based method as well as a relaxiation-based method to reduce the overestimated bias. Experiments on immediate recommendation, sequential recommendations,

as well as safe gridworld reveal that our methods outperform the state-of-the-arts dramatically.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *ICML*, 22–31.

Akifumi Wachi, Y. S. 2020. Safe reinforcement learning in constrained Markov decision processes. In *ICML*, volume 34, preprint.

Berkenkamp, F.; Schoellig, A. P.; and Krause, A. 2016. Safe controller optimization for quadrotors with Gaussian processes. In *ICRA*, 491–496.

Berkenkamp, F.; Turchetta, M.; Schoellig, A.; and Krause, A. 2017. Safe model-based reinforcement learning with stability guarantees. In *NeurIPS*, 908–918.

Bertsekas, D. P. 2015. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific.

Charikar, M. S. 2002. Similarity estimation techniques from rounding algorithms. In *TC*, 380–388.

Chen, H.; Dai, X.; Cai, H.; Zhang, W.; Wang, X.; Tang, R.; Zhang, Y.; and Yu, Y. 2019a. Large-scale interactive recommendation with tree-structured policy gradient. In *AAAI*, 3312–3320.

Chen, J.; and Jiang, N. 2019. Information-theoretic considerations in batch reinforcement learning. In *ICML*, 1042–1051.

Chen, M.; Beutel, A.; Covington, P.; Jain, S.; Belletti, F.; and Chi, E. H. 2019b. Top-k off-policy correction for a REINFORCE recommender system. In *WSDM*, 456–464.

Chen, X.; Li, S.; Li, H.; Jiang, S.; Qi, Y.; and Song, L. 2019c. Generative adversarial user model for reinforcement learning based recommendation system. In *ICML*, 1052–1061.

Cheng, R.; Orosz, G.; Murray, R. M.; and Burdick, J. W. 2019. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *AAAI*, 3387–3395.

Chow, Y.; Ghavamzadeh, M.; Janson, L.; and Pavone, M. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *JMLR* 18(1): 6070–6120.

Chow, Y.; Nachum, O.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2018. A Lyapunov-based approach to safe reinforcement learning. In *NeurIPS*, 8092–8101.

Chow, Y.; Tamar, A.; Mannor, S.; and Pavone, M. 2015. Risk-sensitive and robust decision-making: A cvar optimization approach. In *NeurIPS*, 1522–1530.

Devroye, L.; Györfi, L.; and Lugosi, G. 1996. *A Probabilistic Theory of Pattern Recognition*, volume 31. Springer-Verlag.

Elahi, M.; Ricci, F.; and Rubens, N. 2016. A survey of active learning in collaborative filtering recommender systems. *CSR* 20: 29–50.

Gelada, C.; Kumar, S.; Buckman, J.; Nachum, O.; and Bellemare, M. G. 2019. DeepMDP: Learning continuous latent space models for representation learning. In *ICML*, 2170–2179.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW*, 173–182.

Jiang, N.; Krishnamurthy, A.; Agarwal, A.; Langford, J.; and Schapire, R. E. 2017. Contextual decision processes with low Bellman rank are PAC-learnable. In *ICML*, 1704–1713.

Leike, J.; Martic, M.; Krakovna, V.; Ortega, P. A.; Everitt, T.; Lefrancq, A.; Orseau, L.; and Legg, S. 2017. AI Safety Gridworlds. *arXiv preprint arXiv:1711.09883* .

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 661–670.

Metelli, A. M.; Likmeta, A.; and Restelli, M. 2019. Propagating uncertainty in reinforcement learning via Wasserstein barycenters. In *NeurIPS*, 4335–4347.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Mohri, M.; Rostamizadeh, A.; and Talwalkar, A. 2012. *Foundations of Machine Learning*, volume 1. MIT Press.

Munos, R.; Stepleton, T.; Harutyunyan, A.; and Bellemare, M. 2016. Safe and efficient off-policy reinforcement learning. In *NeurIPS*, 1054–1062.

Munos, R.; and Szepesvári, C. 2008. Finite-time bounds for fitted value iteration. *JMLR* 9(May): 815–857.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted Boltzmann machines. In *ICML*.

Polymenakos, K.; Abate, A.; and Roberts, S. 2019. Safe policy search using Gaussian process models. In *AAMAS*, 1565–1573.

Ricci, F.; Rokach, L.; and Shapira, B. 2011. Introduction to recommender systems handbook. In *Recommender Systems Handbook*. Springer.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952* .

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An MDP-based recommender system. *JMLR* 6: 1265–1295.

Slivkins, A. 2019. Introduction to Multi-armed Bandits. *arXiv preprint arXiv:1904.07272* .

Sridharan, K. 2002. *A gentle introduction to concentration inequalities*. Citeseer.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

Tang, H.; Houthooft, R.; Foote, D.; Stooke, A.; Chen, O. X.; Duan, Y.; Schulman, J.; DeTurck, F.; and Abbeel, P. 2017. # exploration: A study of count-based exploration for deep reinforcement learning. In *NeurIPS*, 2753–2762.

Turchetta, M.; Berkenkamp, F.; and Krause, A. 2016. Safe exploration in finite Markov decision processes with Gaussian processes. In *NeurIPS*, 4312–4320.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double Q-learning. In *AAAI*.

Xin, X.; Karatzoglou, A.; Arapakis, I.; and Jose, J. M. 2020. Self-supervised reinforcement learning for recommender systems. *SIGIR* .

Yu, M.; Yang, Z.; Kolar, M.; and Wang, Z. 2019. Convergent policy optimization for safe reinforcement learning. In *NeurIPS*, 3127–3139.

Zhang, R.; Yu, T.; Shen, Y.; Jin, H.; and Chen, C. 2019. Text-based interactive recommendation via constraint-augmented reinforcement learning. In *NeurIPS*, 15214–15224.

Zhao, X.; Xia, L.; Tang, J.; and Yin, D. 2019. Deep reinforcement learning for search, recommendation, and online advertising: a survey. *SIGWEB* Spring 2019: 4.

Zhao, X.; Xia, L.; Zhang, L.; Ding, Z.; Yin, D.; and Tang, J. 2018a. Deep reinforcement learning for page-wise recommendations. In *Recsys*, 95–103.

Zhao, X.; Zhang, L.; Ding, Z.; Xia, L.; Tang, J.; and Yin, D. 2018b. Recommendations with negative feedback via pairwise deep reinforcement learning. In *KDD*, 1040–1048.

Zou, L.; Xia, L.; Ding, Z.; Song, J.; Liu, W.; and Yin, D. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *KDD*, 2810–2818.

A Omitted Proofs

This section shows the missing proofs in the main text, including 1) the upper bound for $(\underline{Q}(s^i,a)-\underline{Q}^*(s^i,a))^2$ in a certain state, 2) the safety guarantee of our method and 3) the convergence analysis of our method.

To facilitate our analysis, we define $\|\cdot\|_\infty$ as the infinity norm. We also use the inner product to represent the expectation: i.e., $PV^\pi(s') := \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^\pi(s')\right]$ and omit the superscript π if there is no confusion.

A.1 Upper bound for action selection

We first prove the upper bound for $(\underline{Q}(s^i,a)-\underline{Q}^*(s^i,a))^2$ in a certain state s^i. As we discuss in the main text, we use an alternative form of the temporal difference:

$$\begin{aligned}&\mathcal{L}_D\left(\underline{Q}^{h'},\underline{Q}^h\right)=\tfrac{1}{|D|}\textstyle\sum_{(s,a,r,s')\in D}\left[\underline{Q}^{h'}(s,a)-c(s,a)-\gamma\underline{Q}^h\left(s',a'\right)\right]^2,\\&\underline{Q}^h_{k+1}=\tau\underline{Q}^h_{k+1}:=\arg\min_{\underline{Q}^h_k}\mathcal{L}_D\left(\underline{Q}^{h'},\underline{Q}^h_k\right).\end{aligned}\tag{5}$$

These formulations are adapted from (Munos and Szepesvári 2008). τ is an operator to find $\underline{Q}$ that minimize $\mathcal{L}_D\left(\underline{Q}^{h'},\underline{Q}^h\right)$.

Lemma 1. *For $\forall k$, when the policy is stationary, the inequality is always established:*

$$\left\|\mathcal{T}\underline{Q}^h_k(s,a)-\underline{Q}^{h*}(s,a)\right\|^2_\infty\le\gamma^2\left\|\underline{Q}^h_k(s,a)-\underline{Q}^{h*}(s,a)\right\|^2_\infty.$$

Proof. This can be done by using the definition of $\underline{Q}^h$.

$$\begin{aligned}&\left\|\mathcal{T}\underline{Q}^h_k(s,a)-\underline{Q}^{h*}(s,a)\right\|^2_\infty=\left\|\mathcal{T}\underline{Q}^h_k(s,a)-\mathcal{T}\underline{Q}^{h*}(s,a)\right\|^2_\infty\\&=\left\|r+\gamma\mathbb{E}\underline{Q}^h_k(s,a)-r-\gamma\mathbb{E}\underline{Q}^*(s,a)\right\|^2_\infty\\&\le\gamma^2\left\|\underline{Q}^h_k(s,a)-\underline{Q}^*(s,a)\right\|^2_\infty,\end{aligned}$$

which completes the proof. □

Lemma 2. *We have $|\mathcal{L}_{D^{s^i}}\left(\underline{Q}_k(s^i,a),\underline{Q}_{k-1}(s^i,a)\right)-\mathcal{L}\left(\underline{Q}_k(s^i,a),\underline{Q}_{k-1}(s^i,a)\right)|\le\chi=\frac{|C_0|}{1-\gamma}\sqrt{\frac{1}{2(n\wedge1)}\log\frac{2|A|}{\delta}}$ with probability at least $1-\delta$, where n is the visiting number for state s^i and $n\wedge1:=\max\{n,1\}$.*

Proof. Notice that the $\underline{Q}$ function is bound by $[C_{\min},C_{\max}]$. We first define a sequence of random variables $\langle X_l\rangle_1^N$:

$$X_l=\left[\underline{Q}^h_k(s^i_l,a^j_l)-c(s_l,a_l)-\gamma\underline{Q}^h_{k-1}(s^{i'}_l,a^{j'}_l)\right]^2.$$

Then we can check that $X_l-\mathbb{E}\left[\underline{Q}^h_k(s^i,a^j)-c(s,a)-\gamma\underline{Q}^h_{k-1}(s^{i'},a^j)\right]^2$ is a martingale difference sequence bounded by $[(\frac{C_{\min}}{1-\gamma})^2,(\frac{C_{\max}}{1-\gamma})^2]$ and $\mathcal{L}_{D^{s^i}}\left(\underline{Q}_k(s^i,a),\underline{Q}_{k-1}(s^i,a)\right)=\frac{1}{n}\sum_{l=1}^n X_l$, i.e., the loss function is the average of the random variables X_l. D^{s^i} is the tuple $\{s^i,a,c,s^{'}\}$, containing state s^i. Then, by applying the Azuma's inequality and the union bound w.r.p actions, we can get

$$\forall a,\mathbb{P}\left\{\left|\frac{1}{n}\sum_{l=1}^n X_l-\mathbb{E}(X_l)\right|\ge\frac{t}{n}\right\}\le2\gamma|A|\exp\left(-\frac{2t^2(n\wedge1)}{|C|^2/(1-\gamma)^2}\right).$$

Therefore, with probability at least $1-\delta$, we have $\left|\frac{1}{n}\sum_{l=1}^n X_l-\mathbb{E}(X_l)\right|\le\chi$, where

$$\chi=\frac{|C_{\min}-C_{\max}|}{1-\gamma}\sqrt{\frac{1}{2(n\wedge1)}\log\frac{2\gamma|A|}{\delta}}.\tag{6}$$

□

Lemma 3. *For a fixed abstraction mapping function $h(\cdot)$, we have $|\mathcal{L}_{D^{s^i}}\left(\underline{Q}_k(s^i,a),\underline{Q}_{k-1}(s^i,a)\right)-\mathbb{E}\mathcal{L}\left(\underline{Q}_k(s^i,a),\underline{Q}_{k-1}(s^i,a)\right)|\le\chi=\frac{|C_0|}{1-\gamma}\sqrt{\frac{1}{2(n\wedge1)}\log\frac{2|A|}{\delta}}$ with probability at least $1-\delta$, where n is the count number for state embedding $h(s^i)$ and $n\wedge1:=\max\{n,1\}$.*

Proof. This proof is similar to Lemma 2, except that the n is for the visiting time for $h(s)$ rather than that of s. □

Remark. As we mentioned above, it is very hard to count the visiting times of different states when the state size is large. Moreover, it is also sample inefficient and unsafe if we separate all the states, since some states might have similar properties. Therefore, this lemma above reveals that we may count the visiting times of a more compact state representation so as to decrease the unsafe risk while speeds up the training process.

Corollary 1. *If $\underline{Q}_0^h(s^i, a^j) = 0$ (the Q is initialized by zero), for state s^i and action a^j, the upper bound is always established:*

$$\left(\underline{Q}_k^h(s^i, a^j) - \underline{Q}^{h,*}(s^i, a^j)\right)^2 \leq \frac{2|C_{\min} - C_{\max}|}{1-\gamma}\sqrt{\frac{1}{2(n \wedge 1)}\log\frac{2\gamma|A|}{\delta}} + \frac{(C_{\max})^2\gamma^{2k}}{(1-\gamma)^2}. \tag{7}$$

Proof.

$$\begin{aligned}
\left(\underline{Q}_k^h(s^i, a^j) - \underline{Q}^{h,*}(s^i, a^j)\right)^2 &= \left(\underline{Q}_k^h(s^i, a^j) - \underline{\mathcal{T}Q}_{k-1}(s^i, a^j) + \underline{\mathcal{T}Q}_{k-1}(s^i, a^j) - \underline{Q}^*(s^i, a^j)\right)^2, \\
&\leq \left\|\underline{Q}_k(s^i, a) - \underline{\mathcal{T}Q}_{k-1}(s^i, a)\right\|_\infty^2 + \left\|\underline{\mathcal{T}Q}_{k-1}(s, a) - \underline{\mathcal{T}Q}^*(s, a)\right\|_\infty^2, \\
&\leq \left\|\underline{Q}_k(s^i, a) - \underline{\mathcal{T}Q}_{k-1}(s^i, a)\right\|_\infty^2 + \gamma^{2k}\left\|\underline{Q}_0(s, a) - \underline{Q}^*(s, a)\right\|_\infty^2, \\
&\leq \max_a \left[\mathcal{L}_{D^{s^i}}\left(\underline{Q}_k(s^i, a), \underline{Q}_{k-1}(s^i, a)\right) - \mathcal{L}_{D^{s^i}}\left(\underline{\mathcal{T}Q}_{k-1}(s^i, a), \underline{Q}_{k-1}(s^i, a)\right)\right] + \frac{(C_{\max})^2\gamma^{2k}}{(1-\gamma)^2}, \\
&\leq 2\chi + \frac{(C_{\max})^2\gamma^{2k}}{(1-\gamma)^2} = \frac{2|C_{\min} - C_{\max}|}{1-\gamma}\sqrt{\frac{1}{2(n \wedge 1)}\log\frac{2\gamma|A|}{\delta}} + \frac{(C_{\max})^2\gamma^{2k}}{(1-\gamma)^2}.
\end{aligned}$$

□

The second inequality uses Lemma 1 while the third inequality is adapted from Theorem 11 in (Chen and Jiang 2019). The last inequality uses Lemma 2.

Lemma 4. *$(Q^h(s^i, a) - Q^*(s^i, a))^2$ is bounded by $(\frac{C_{\max}}{1-\gamma})^2 + (Q^*(s^i, a) - Q_k^h(s^i, a))^2$.*

Proof. The proof can be got immediately by noticing that $\forall s, a : |Q^*(s, a)| \leq |\frac{C_{\max}}{1-\gamma}|$:

$$\begin{aligned}
(Q^h(s, a) - Q^*(s, a))^2 &\leq (Q^{*h}(s, a) - Q_k^h(s, a))^2 + (Q^*(s, a) - Q_k^h(s, a))^2 \\
&\leq \|Q^{*h}(s, a) - Q_k^h(s, a)\|_\infty^2 + (Q^*(s, a) - Q_k^h(s, a))^2 \\
&\leq (\frac{C_{\max}}{1-\gamma})^2 + (Q^*(s, a) - Q_k^h(s, a))^2.
\end{aligned}$$

□

Theorem 1. *For a certain state $s^i \in S$, with probability at least $1 - \delta$, assume the policy is stationary, the upper bound for $\left(\underline{Q}^h(s^i, a) - \underline{Q}^*(s^i, a)\right)^2$ is:*

$$\left(\underline{Q}(s^i, a) - \underline{Q}^*(s^i, a)\right)^2 \leq \frac{2|C_{\min} - C_{\max}|}{1-\gamma}\sqrt{\frac{1}{2(n \wedge 1)}\log\frac{2\gamma|A|}{\delta}} + \frac{(\sqrt{2}C_{\max})^2}{(1-\gamma)^2} + (Q^*(s, a) - Q_k^h(s, a))^2. \tag{8}$$

Proof. we decompose the term $\left(\underline{Q}^h(s^i, a) - \underline{Q}^*(s^i, a)\right)^2$ by the triangle inequality and using Corollary 1 and Lemma 4:

$$\begin{aligned}
&\left(\underline{Q}^h(s^i, a) - \underline{Q}^*(s^i, a)\right)^2 \\
&\leq \left(\underline{Q}^h(s^i, a) - \underline{Q}^{h,*}(s^i, a)\right)^2 + \left(\underline{Q}^{h,*}(s^i, a) - \underline{Q}^*(s^i, a)\right)^2 \\
&\leq \frac{2|C_{\min} - C_{\max}|}{1-\gamma}\sqrt{\frac{1}{2(n \wedge 1)}\log\frac{2\gamma|A|}{\delta}} + \frac{(\sqrt{2}C_{\max})^2}{(1-\gamma)^2} + (Q^*(s, a) - Q_k^h(s, a))^2 \\
&\leq \frac{2|C_{\min} - C_{\max}|}{1-\gamma}\sqrt{\frac{1}{2(n \wedge 1)}\log\frac{2\gamma|A|}{\delta}} + \frac{(\sqrt{2}C_{\max})^2}{(1-\gamma)^2} + (C_{\max} - Q_k^h(s, a))^2
\end{aligned}$$

□

As we assume each gradient descent in deep learning setting can approximated the operator $\underline{Q}^h_{k+1} := \arg\min_{\underline{Q}^h} \mathcal{L}_D\left(\underline{Q}_{k+1}, \underline{Q}^h\right)$ in the main text. Therefore, Theorem 1 can be used to estimate the bias for $\underline{Q}^h(s^i, a)$.

A.2 Safety Property

This sections we discuss our method can always guarantee safety.

We first show that the optimal policy π^* is safe:

Lemma 5. *Assuming that there always exists a policy is safety in a constrained MDP, i.e. $\Omega_{safe} \neq \emptyset$, for the $\underline{Q}^*$ and its policy π^*, the policy π^* will never choose actions that leads to unsafety: $\pi^* \cap \Omega_{unsafe} = \emptyset$.*

Proof. Before we begin our analysis, we first define unsafety in state s^i at step $\hat{t}$:

$$\forall a, \hat{C} + \gamma^{\hat{t}} \mathbb{E}^{\pi^*}\left\{\sum_{t=\hat{t}}^{T} \gamma^t c_t | s_{\hat{t}}\right\} \leq \epsilon, \tag{9}$$

where $\hat{C}$ is the accumulated sum of cost until time $\hat{t}-1$. Now we start to proof that the inequality mentioned above is not true during the whole steps.

The following proof is based on induction and contradiction. We first explore the case when the agent is 'lucky': i.e., the agent choose action which always leads to safe state. We prove that in a safety state, under our policy, we can always reach another safety state within a probability.

i). When $\hat{t} = 1$, if there exists a at least one state s_j that satisfies the safe criteria, as we analyze above, absolutely, the policy will choose actions to these states.

We then show that it is impossible all the states are unsafe by contradiction. Assuming that all the state is unsafe, then for all the policies cannot reach a safety trajectory: $\Omega_{safe} = \emptyset$ which contradicts the assumption that there always exists a safe policy. Therefore, when $\hat{t} = 1$ there always have a safe state. Furthermore, the policy will choose actions to these states.

ii). Assume when $s_{\hat{t}-1}$ ($\hat{t} > 2$), the safety is guaranteed. Then for $s_{\hat{t}}$, the proof is similar to case i). If there exists a at least one state s_j that satisfies the safe criteria, as we analyze above, the policy will choose actions to these states.

If the statement: $\forall s_{\hat{t}}, \hat{C} + \mathbb{E}^{\pi^*}\left\{\sum_{t=\hat{t}}^{T} \gamma^t c_t | s_{\hat{t}}\right\} \leq \epsilon$ is true, then we have:

$$\tilde{C} + \mathbb{E}^{\pi^*}\left\{\sum_{t=\tilde{t}}^{T} \gamma^t c_t | s_{\hat{t}}\right\} = \hat{C} + \gamma^{\hat{t}-1}\mathbb{E}^{\pi^*}\left\{c_{\hat{t}-1}\right\} - \gamma^{\hat{t}-1} c_{\hat{t}-1} + \mathbb{E}^{\pi^*}\left\{\sum_{t=\hat{t}}^{T} \gamma^t c_t | s_{\hat{t}}\right\} \geq \epsilon$$

where $\tilde{C} = \hat{C} + \gamma^{\hat{t}-1}\mathbb{E}^{\pi^*}\left\{c_{\hat{t}-1}\right\} - \gamma^{\hat{t}-1} c_{\hat{t}-1}$. The second inequality is because $\mathbb{E}^{\pi^*}\left\{c_{\hat{t}-1}\right\} \geq \gamma^{\hat{t}-1} c_{\hat{t}-1}$. Then when the transition function leads to the state that is unsafe. Definitely, choosing any of the policy will all lead to unsafety. But through backtracking, we will always find one state satisfying the safe criteria (The first one is the initial state). Therefore, π^* always satisfy the safety criteria.

The final thing is to prove $\mathbb{E}^{\pi^*}\left\{c_{\hat{t}-1}\right\} \geq \gamma^{\hat{t}-1} c_{\hat{t}-1}$. This can be done by contradiction. If $\mathbb{E}^{\pi^*}\left\{c_{\hat{t}-1}\right\} < \gamma^{\hat{t}-1} c_{\hat{t}-1}$ and $\hat{C} + \mathbb{E}^{\pi^*}\left\{\sum_{t=\hat{t}}^{T} \gamma^t c_t | s_{\hat{t}}\right\} \leq \epsilon$, then we can check that $s_{\hat{t}-1}$ is not safety any longer which contradicts the statement.

Thus the statement of the Lemma is true. □

Now, we show the safety for π.

Theorem 2. *If $\Omega_{safe} \neq \emptyset$, the statement $\pi \cap \Omega_{unsafe} = \emptyset$ is established with probability at least $1 - \delta$.*

Proof. Before we begin our proof, we first show that

$$\underline{Q}(s_0, a) = c(s_0) + \gamma \mathbb{E}_{s'} \underline{Q}(s', \pi(s')) \geq c(s_0) + \gamma \mathbb{E}_{s'}(\underline{Q}^*(s', \pi(s'))) = c(s_0, a) + \mathbb{E}^{\pi^*}\left\{\sum_{t=1}^{T} \gamma^t c_t | s'\right\}$$

This first inequality is achieved due to definition of π as well as the fact that the $\underline{Q}(s, a) \leq \underline{Q}^*(s, a) + \sigma(s, a)$.

We can expand this result to all the step. For any step $\hat{t} \leq T$, notice that $\mathbb{E}^{\pi^*}\{\mathbb{E}^{\pi^*}\left\{\sum_{t=\hat{t}}^{T} \gamma^t c_t | s^t\right\} | s^0\} = \mathbb{E}^{\pi^*}\left\{\sum_{t=\hat{t}}^{T} \gamma^t c_t | s^t\right\}$ (due to the tower property), similarly, we have $\underline{Q}(s_{\hat{t}}, a) \geq c(s_{\hat{t}}, a) + \mathbb{E}^{\pi^*}\left\{\sum_{t=\hat{t}}^{T} \gamma^t c_t | s^{\hat{t}+1}\right\}$.

Now, based on Lemma 5, the policy is always safe: i.e., $\pi^* \cap \Omega_{unsafe} = \emptyset$.

Then since $\underline{Q}(s, a) \geq c(s, a) + \mathbb{E}^{\pi^*}\left\{\sum_{t=1}^{T} \gamma^t c_t | s'\right\}$, we have $\pi \cap \Omega_{unsafe} = \emptyset$ with probability at least $1 - \delta$. □

A.3 Convergence Analysis

This section we present the convergence analysis of our method, including the tabular case as well as the specified neural network case.

We first show that the convergence analysis of a special case of our method: the safe constrained coupled value iteration in tabular case:

Lemma 6. *(contraction) Both Q and $\underline{Q}$ satisfy:*

$$\left\|\left(\underline{\mathcal{T}}\underline{Q}\right)(s,a) - \underline{\mathcal{T}}\underline{Q}^*(s,a)\right\|_\infty \leq \gamma \left\|\underline{Q}(s,a) - \underline{Q}^*(s,a)\right\|_\infty \tag{10}$$

and

$$\left\|\left(\mathcal{T}Q\right)(s,a) - \left(\mathcal{T}Q^*\right)(s,a)\right\|_\infty \leq \gamma \left|Q(s,a) - Q^*(s,a)\right|_\infty \tag{11}$$

Proof.

$$\begin{aligned}
&\left\|\left(\mathcal{T}Q\right)(s,a) - \left(\mathcal{T}Q^*\right)(s,a)\right\|_\infty \\
&= \left|\left(r(s,a) + \gamma P V^{\pi}(s')\right) - \left(r(s,a) + \gamma P V^{\pi,*}(s')\right)\right|_\infty \\
&\overset{(a)}{\leq} \gamma P \max_{s'} \left|V(s') - V^*(s')\right| \\
&\overset{(b)}{=} \gamma P \max_{s} \left|V(s) - V^*(s)\right| \\
&\overset{(c)}{\leq} \gamma P \max_{s,a} \left|Q(s,a) - Q^*(s,a)\right| \\
&\leq \gamma \left\|Q(s,a) - Q^*(s,a)\right\|_\infty
\end{aligned}$$

(a) is due to the Jesen's inequality and (b) is the property of the stationary policy. (c) comes from the fact that $\max_a \left|Q(s,a) - Q^*(s,a)\right| \geq \left|\max_a\{Q\left(s,a\right)\mathbb{1}\{\underline{Q}\left(s,a\right) - \epsilon\} - Q^*(s,a)\mathbb{1}\{\underline{Q}^*\left(s,a\right)\}\right| \geq \left|\max_a\{Q\left(s,a\right)\mathbb{1}\{\underline{Q}\left(s,a\right) - \epsilon\} - \max_{a'} Q^*(s,a')\mathbb{1}\{\underline{Q}^*\left(s,a'\right)\}\right|$. For $\underline{Q}$, the proof is similar:

$$\begin{aligned}
&\left\|\left(\underline{\mathcal{T}}\underline{Q}\right)(s,a) - \left(\underline{\mathcal{T}}\underline{Q}^*\right)(s,a)\right\|_\infty \\
&\overset{(a)}{\leq} \gamma P \max_{s,a} \left|\underline{Q}(s,a) - \underline{Q}^*(s,a)\right| \\
&\leq \gamma \left\|\underline{Q}(s,a) - \underline{Q}^*(s,a)\right\|_\infty
\end{aligned}$$

(a) is from the fact that $\max_a \left|\underline{Q}(s,a) - \underline{Q}^*(s,a)\right| \geq \left|\sum_a P^{\underline{Q}}_{a,s}\underline{Q}(s,a) - P^{\underline{Q}^*}_{a,s}\underline{Q}^*(s,a)\right|$. □

Proposition 1. *Under the proposed value iteration scheme (Eq.* (1) & (2)*), both the $\underline{Q}$ and Q function can converge to the optimal Q and $\underline{Q}$ function respectively. If we decompose with uncertain decomposition, i.e., $\underline{Q} = \mu_t - \tilde{\sigma}_t$, with $\tilde{\sigma}_t = \gamma P\tilde{\sigma}_t, \quad \mu_{t+1} = c + \gamma P\mu_t$;we have $\lim_{t\to\infty}\tilde{\sigma}_t = 0$, $\lim_{t\to\infty}\mu_t = \underline{Q}^*$.*

Proof. Based on lemma 6, we have:

$$\left\|\left(\underline{\mathcal{T}}\underline{Q}^k\right)(s,a) - \left(\underline{\mathcal{T}}\underline{Q}^*\right)(s,a)\right\|_\infty \leq \gamma \left\|\left(\underline{Q}^k\right)(s,a) - \left(\underline{Q}^*\right)(s,a)\right\|_\infty$$

and

$$\left\|\left(\mathcal{T}Q^k\right)(s,a) - \left(\mathcal{T}Q^*\right)(s,a)\right\|_\infty \leq \gamma \left\|\left(Q^k\right)(s,a) - \left(Q^*\right)(s,a)\right\|_\infty \tag{12}$$

Applying the Banach fixed point theorem, we can conclude that both Q and $\underline{Q}$ are able to converge to optimal solution. Specifically, for $\underline{Q}$, with uncertain decomposition we have

$$\sigma_{t+1} = \gamma P\sigma_t, \quad \mu_{t+1} = c + \gamma P\mu_t; \tag{13}$$

Directly solving the equation above, we have $\sigma^* = \vec{0}, \mu^* = (I - \gamma P)^{-1}c = \underline{Q}^*$, where $\vec{0}$ is the zero vector. □

Here, we need to stress that $\sigma(s,a)$ and $\tilde{\sigma}(s,a)$ are not the same but we can choose $\tilde{\sigma}(s,a)$ to match $\sigma(s,a)$ just like (Metelli, Likmeta, and Restelli 2019).

Now we begin to analysis the neural network case. Unfortunately, due to the fact that the hypothesis of our neural network is infinite, leading the upper bound to be infinite when applying the Hoffding's inequality. Therefore, we need to ferret out another inequality which has a tighter bound for the hypothesis:

Lemma 7. *(Mohri, Rostamizadeh, and Talwalkar 2012) The pseudo-dimension of hyper-planes in R^n is given by*

$$\operatorname{Pdim}\left(\left\{\mathbf{x}\mapsto \mathbf{w}\cdot\mathbf{x}+b : \mathbf{w}\in\mathbb{R}^N, b\in\mathbb{R}\right\}\right) = N+1 \tag{14}$$

This Lemma is directly from Theorem 10.4 of (Mohri, Rostamizadeh, and Talwalkar 2012), where N is the embedding dimension[10].

Lemma 8. *(Devroye, Györfi, and Lugosi 1996; Jiang et al. 2017) Suppose $Pdim(H) \leq d$, then for any $\alpha > 0$, we have:*

$$\Pr\left\{\sup_{h\in\mathcal{H}}\left|\frac{1}{n}\sum_{i=1}^{n}h\left(x_i\right)-\mathbb{E}\left[h\left(x_i\right)\right]\right|>\alpha\right\}\leq 8e(d+1)\left(\frac{16e}{\alpha}\right)^{d}\exp\left(-\frac{n\alpha^2}{128b^2}\right) \tag{15}$$

where $\mathcal{H}\subset\mathcal{X}\to[0,b]$ be a hypothesis class, $(x_1,\ldots,x_n)$ be i.i.d. samples drawn from some distribution supported on $\mathcal{X}$. This lemma is from Corollary 2 of (Jiang et al. 2017) .

Corollary 2. *Based on Lemma 7 and 8 we always have*

$$\sup_{h\in\mathcal{H}}\left|\frac{1}{n}\sum_{i=1}^{n}h\left(x_i\right)-\mathbb{E}\left[h\left(x_i\right)\right]\right|\leq 2^{(4d+3)}e^{(d+1)}(d+1)\exp(LambertW(\frac{4b^2 2^{(-8d)}e^{(-2d-2)}\log\delta}{nd(d^2+2d+1)})/2) = 2^{(4N+7)}e^{(N+2)}(N+2) \tag{16}$$

with probability $1-\delta$, where $LambertW(\cdot)$ is the Lambert W function and d is the pesudo-dimension.

Proof. Solving Eq. (15)[11], we have:

$$\begin{aligned}&\sup_{h\in\mathcal{H}}\left|\frac{1}{n}\sum_{i=1}^{n}h\left(x_i\right)-\mathbb{E}\left[h\left(x_i\right)\right]\right|\\&\leq 2^{(4d+3)}e^{(d+1)}(d+1)\exp(LambertW(\frac{4b^2 2^{(-8d)}e^{(-2d-2)}\log\delta}{nd(d^2+2d+1)})/2).\end{aligned}$$

The last inequality is due to the fact that $0\geq LambertW(\frac{4b^2 2^{(-8d)}e^{(-2d-2)}\log\delta}{nd(d^2+2d+1)})/2)$ while the last equation is from Lemma 7. □

Before we get into deeper into the analysis of NN-based Q-network. To facility our analysis, we define an ideal operator update the Q function which is similar to Eq. (5):

$$\begin{aligned}&\mathcal{L}_D\left(Q^{h'},Q^h\right)=\tfrac{1}{|D|}\textstyle\sum_{(s,a,r,s')\in D}\left|Q^{h'}(s,a)-c(s,a)-\gamma\max_{a'}Q^{h'}\left(s',a'\right)\right|,\\&Q^h_{k+1}=\hat{\tau}Q^h_k:=\arg\min_{Q^h}\mathcal{L}_D\left(Q^{h'},Q^h_k\right).\end{aligned}$$

Lemma 9. *Assume that the neural networks based Q satisfy those conditions: 1). the feature extraction layers are fixed and 2)the decision layer is a linear mapping. When the Q is initialized by zero, i.e., $Q^h_1=0$. Then, based on Corollary 2, we have*

$$\left\|\tilde{Q}^h_k(s,a)-Q^{h*}_k(s,a)\right\|_\infty\leq 2^{(4N+7)}e^{(N+2)}(N+2)\exp(LaW(\frac{4b^2 2^{(-8N-8)}e^{(-2N-4)}\log\delta}{nd((N+1)^2+2N+3)})/2)+\gamma\frac{R_{\max}}{1-\gamma}$$

with probability at least $1-\delta$.

Proof. Deducting $\left\|\tilde{Q}^h_k-Q^{h*}\right\|_\infty$ we obtain:

$$\begin{aligned}\left\|\tilde{Q}^h_k-Q^{h*}\right\|_\infty&=\left\|\tilde{Q}^h_k+\mathcal{T}\tilde{Q}^h_{k-1}-\mathcal{T}\tilde{Q}^h_{k-1}-Q^{h*}\right\|_\infty\\&\leq\left\|\tilde{Q}^h_k-\mathcal{T}\tilde{Q}^h_{k-1}\right\|_\infty+\left\|\mathcal{T}\tilde{Q}^h_{k-1}-\mathcal{T}Q^{h*}\right\|_\infty\\&\leq\left\|\tilde{Q}^h_k-\mathcal{T}\tilde{Q}^h_{k-1}\right\|_\infty+\gamma\left\|\tilde{Q}^h_{k-1}-Q^{h*}\right\|_\infty\\&\leq\left\|\tilde{Q}^h_k-\mathcal{T}\tilde{Q}^h_{k-1}\right\|_\infty+\gamma\left\|\tilde{Q}^h_{k-1}-Q^{h*}\right\|_\infty\\&\leq\sum_{i=2}^{k}\gamma^{k-i+1}\left\|\tilde{Q}^h_i-\mathcal{T}\tilde{Q}^h_{i-1}\right\|_\infty+\gamma^{k-1}\left\|\tilde{Q}^h_1-Q^{h*}\right\|_\infty\end{aligned} \tag{17}$$

[10]More details about the pseudo-dimension can also be found in (Mohri, Rostamizadeh, and Talwalkar 2012).

[11]We leverage the computer-aided tool (Sympy https://www.sympy.org/en/index.html) to solve this inequality due to the high computation.

Using the similar approach in Corollary 1, we have $\left\|\tilde{Q}_i^h - \mathcal{T}\tilde{Q}_{i-1}^h\right\|_\infty = \max\left[\mathcal{L}_D\left(\underline{Q}_k, \underline{Q}_{k-1}\right) - \mathcal{L}_D\left(\mathcal{T}^h\underline{Q}_{k-1}, \underline{Q}_{k-1}\right)\right]$, then if we set

$$|c(s^i, a^j) + \gamma Q(s', \pi(s')) - Q(s^i, \pi(a^j))| - |c(s^i, a^j) + \gamma Q(s', \pi(s')) - \hat{\mathcal{T}}Q(s^i, \pi(a^j))| := Z_i$$

as a random variable bounded by $[0, \frac{2R_{\max}}{1-\gamma}]$, applying the Corollary 2 we have:

$$\forall i, \left\|\tilde{Q}_i^h - \mathcal{T}\tilde{Q}_{i-1}^h\right\|_\infty \leq 2^{(4N+7)}e^{(N+2)}(N+2)\exp(LaW(\frac{4b^2 2^{(-8N-8)}e^{(-2N-4)}\log\delta}{nd((N+1)^2+2N+3)})/2) \tag{18}$$

Adding them together, we finally get:

$$\left\|\tilde{Q}_k^h - Q^{h*}\right\|_\infty \leq \frac{1}{1-\gamma}2^{(4N+7)}e^{(N+2)}(N+2)\exp(LaW(\frac{4b^2 2^{(-8N-8)}e^{(-2N-4)}\log\delta}{nd((N+1)^2+2N+3)})/2) + \gamma^{k-1}\frac{R_{\max}}{1-\gamma}$$

with probability at least $1-\delta$. □

Lemma 10. *Based on Lemma 6, we have*

$$\left\|\mathcal{T}Q^{h*} - \mathcal{T}Q^*\right\|_\infty = 0 \tag{19}$$

Proof.

$$\begin{aligned}
&\left\|\mathcal{T}Q^{h*} - \mathcal{T}Q^*\right\|_\infty = \|r(s,a) + \gamma(PV^*(h(s')) - r(h(s),a) - \gamma P^*V(s')\|_\infty \\
&\leq \|\gamma(PV^*(h(s')) - \gamma PV^*(s')\|_\infty \\
&\leq \gamma\left\|V(h(s')) - V(s')\right\|_\infty \overset{(a)}{\leq} \gamma\left|Q^{h*} - Q^*\right\|_\infty
\end{aligned}$$

(a) is due to the fact that $\max_a\left|Q^{h*}(s,a) - Q^*(s,a)\right| \geq \left|\max_a\{Q^{h*}(s,a)\,\mathbb{1}\{\underline{Q}^{h*}(s,a)\} - Q^*(s,a)\mathbb{1}\{\underline{Q}^*(s,a)\}\right| \geq \left|\max_a\{Q^{h*}(s,a)\,\mathbb{1}\{\underline{Q}^{h*}(s,a)\} - \max_{a'}Q^*(s,a')\mathbb{1}\{\underline{Q}^*(s,a')\}\right|$ which is similar to Lemma 6.

Also, we have:

$$\left\|\mathcal{T}Q^{h*} - \mathcal{T}Q^*\right\|_\infty = \left\|Q^{h*} - Q^*\right\|_\infty$$

because of the optimal operator.

Thus combining the analysis above, we finally have:

$$\left\|\mathcal{T}Q^{h*} - \mathcal{T}Q^*\right\|_\infty = \gamma\left\|V(h(s')) - V(s')\right\|_\infty \overset{(a)}{\leq} \gamma\left\|Q^{h*} - Q^*\right\|_\infty \to \left\|\mathcal{T}Q^{h*} - \mathcal{T}Q^*\right\|_\infty = 0$$

□

Theorem 3. *For a stationary policy, assuming that the neural networks based Q satisfies those two conditions: (1) the feature extraction layers are fixed. (2) The decision layer is a linear mapping. When Q is initialized by zero, i.e., $Q_1^h = 0$, we always have:*

$$\left\|\,Q_k^h(s,a) - Q^*(s,a)\right\|_\infty \leq 2^{(4N+7)}e^{(N+2)}(N+2)\exp(LaW(\frac{4b^2 2^{(-8N-8)}e^{(-2N-4)}\log\delta}{nd((N+1)^2+2N+3)})/2) + \gamma^{k-1}\frac{R_{\max}}{1-\gamma},$$

and

$$\left\|\,\underline{Q}_k^h(s,a) - \underline{Q}^*(s,a)\right\|_\infty \leq 2^{(4N+7)}e^{(N+2)}(N+2)\exp(LaW(\frac{4b^2 2^{(-8N-8)}e^{(-2N-4)}\log\delta}{nd((N+1)^2+2N+3)})/2) + \gamma^{k-1}\frac{|C_{\min}|}{1-\gamma},$$

with probability at least $(1-\delta)^2$.

Proof. Apply Lemmas 6 and 9, we can find that:

$$\begin{aligned}
&\left\|\tilde{Q}_k^h(s,a) - Q^*(s,a)\right\|_\infty \\
&\leq \left\|\,Q_k^{h*}(s,a) - Q^*(s,a)\right\|_\infty + \left\|\tilde{Q}_k^h(s,a) - Q^{h*}(s,a)\right\|_\infty \\
&\leq 2^{(4N+7)}e^{(N+2)}(N+2)\exp(LaW(\frac{4b^2 2^{(-8N-8)}e^{(-2N-4)}\log\delta}{nd((N+1)^2+2N+3)})/2) + \gamma^{k-1}\frac{R_{\max}}{1-\gamma},
\end{aligned}$$

For, $\left\|\tilde{Q}_k^h(s,a) - Q^*(s,a)\right\|_\infty$, we can use the similar technique. □

When $n \to \infty$ and $k \to \infty$, we can check that $\left\|\,Q_k^h(s,a) - Q^*(s,a)\right\|_\infty = 0$.

Algorithm 2: Safe Constrained Multi-armed Bandit

Result: the well-trained Q and $\underline{Q}$.

Initialize Q and $\underline{Q}$ with zero matrice.;

for *each step t* **do**

  Pull an action $a_t = \pi_t(a)$ according to Eq. (21), and receive the reward r and cost c.;

end

Update Q_{t+1} and $\underline{Q}_{t+1}(a)$ according to Eqs. (22).;

B Relationship to Multi-armed Bandits with Constraints

When the length of the time step is *one* and the environment is stateless, the MDP degenerates to the multi-armed bandit (MAB) settings with constraints (Slivkins 2019). In the following, we will show that with a slight modification of our method, we can apply our approach to MAB with constraints.

Lemma 11. *When the time step is only* one *and the environment is stateless, if $C_{\max} = 0$, we have*

$$|\underline{c}(a) - c^*(a)| \leq |C_{\min} - C_{\max}|\sqrt{\frac{\log n}{2(n \wedge 1)}}\,, |r(a) - r^*| \leq |R_{\max}|\sqrt{\frac{\log n}{2(n \wedge 1)}} \tag{20}$$

with probability at least $(1 - \frac{2}{n^2})^2$, where $\underline{c}(a)$ and $r(a)$ are the estimated cost and reward for arm a and c^* and r^* are the true cost and reward for that arm.

Proof. The proof can be easily done by applying the same approaches from Lemma 1.5 in (Slivkins 2019) to c and r. □

For real-world application, we can control the exploration rate by introducing two manually set constants ψ_1 & ψ_2 to replace $|C_{\min} - C_{\max}|$ and $|R_{\max}|$ respectively as what UCB method has done (Slivkins 2019). Details can be found in C.1.

C Algorithm Details

C.1 Safe Constrained Multi-armed Bandit

Adapted from Chapter 2 in (Sutton and Barto 2018), we modify the original UCB method into our new algorithms. We firstly define the policy $\bar{\pi}$ at step t as:

$$\bar{\pi}_t = \underset{a \in A'}{\arg\max} \left[Q_t(a) + \psi_1 \sqrt{\frac{\ln t}{n}} \right], A' = \left\{ a | \left[\underline{Q}_t(a) - \psi_2 \sqrt{\frac{\ln t}{n}} \right] \geq \epsilon \right\}. \tag{21}$$

where ψ_1 & ψ_2 are the positive hyper-parameters. We additionally define the updating rules for Q and $\underline{Q}$:

$$\begin{aligned} Q_{t+1}(a) &= Q_t(a) + \frac{1}{t}\left[r - Q_t(a)\right], \\ \underline{Q}_{t+1}(a) &= \underline{Q}_t(a) + \frac{1}{t}\left[c - \underline{Q}_t(a)\right]. \end{aligned} \tag{22}$$

The pseudo-code can be found in Alg. 2.

C.2 Safe Double DQN with priorier replay buffer

We briefly introduce how to combine the Double DQN (DDQN) method with the prioritized replay buffer in our framework.

For the safe DDQN, adapted from the original DDQN (Van Hasselt, Guez, and Silver 2016) we can define the TD-loss

$$L_d = \left(r_{t+1} + \gamma Q_{\bar{\theta}}\left(s_{t+1}, \underset{a'}{\text{argmax}} Q_\theta\left(s_{t+1}, a'\right)\right) - Q_\theta\left(s_t, a_t\right)\right)^2 + \left(c_{t+1} + \gamma \underline{Q}_{\bar{\theta}}\left(s_{t+1}, \underset{a'}{\text{argmax}} \underline{Q}_\theta\left(s_{t+1}, a'\right)\right) - \underline{Q}_\theta\left(s_t, a_t\right)\right)^2$$

Where $Q_{\bar{\theta}}$ and $\underline{Q}_{\bar{\theta}}$ are the target networks for Q and $\underline{Q}$ respectively. The first term is exact the double-Q TD-loss for Q while second term is the TD-loss for $\underline{Q}$.

We introduce the prioritized replay buffer to further stabilize the prioritized replay buffer (Schaul et al. 2015):

$$p_t \propto |L_d|^{\omega/2},$$

where ω is a hyper-parameter to determine the shape of the distribution.

Algorithm 3: Safe Double DQN with priorier replay buffer

Result: the well-trained parameter θ , $\bar{\theta}$, $\hat{\theta}$, and $\underline{\hat{\theta}}$.

Initialize Q and $\underline{Q}$ with parameter θ , $\bar{\theta}$, their target networks with target networks $\hat{\theta}$, $\underline{\hat{\theta}}$ and the experience buffer D.;
for *episode* τ **do**
  for *each step t* **do**
    get s_t, take action $a_t = \pi(a|s_t)$, and reward r_t and s_{t+1}. Store tuple $\langle s_t, a_t, r_t, c_t, s_{t+1}\rangle$ in replay buffer D.;
  end
  sample a batch of $\langle s, a, r, c, s^{'}\rangle$ from D with priority p_t.;
  train θ , $\bar{\theta}$, $\hat{\theta}$, and $\underline{\hat{\theta}}$ by minimizing $\mathcal{L}_d$.;
end

D Lagrangian DQN

We briefly introduce the Lagrangian DQN. Generally speaking, Lagrangian DQN is to converting the original CMDP into a minmax problem:

$$\min_{\pi}\max_{\lambda\geq 0} Q_{\pi}\left(x_0\right) + \lambda\left(\underline{Q}_{\pi}\left(x_0\right) - \epsilon\right),$$

where λ is the Lagrange multiplier of the CMDP cost. We can optimize the Q, $\underline{Q}$ and λ synchronously with three time scale gradient descent.

E Extra Experiments

This section show more details of the experiments missing in the main text.

E.1 Details of implementation.

Hyper-parameters. Regarding the hyper-parameters, for MAB, we set the degree of exploration to 3 for UCB strategy, $\epsilon = 0.1$ for the ϵ-greedy strategy, and $\psi_1 = \psi_2 = 3$ for our method.For other environments, we use the same hyper-parameters among all the methods, as shown in Tab. 4. We conduct all the experiments in Python 3 and Pytorch[12] within a single GeForce GTX 1080 Ti. The running time for RSC-DQN and S2C-DQN in SR II are shown in Tab. 3.

Metrics. We also implement other metrics to evaluate our methods in different aspects:

1) Accumulated reward. Accumulated reward is to show how agent perform in the environment. For MAB environment, we use the accumulated reward is exact the instant reward since MAB only has one step.

2) Unsafe Ratio (UR). UR is to evaluate how agent is against the cost: UR $= \sum_{i=1}^{n} \mathbb{1}\left(\sum_j c_j < \epsilon\right)/n$, where i is the i-th user, total n users and j is the index of time step.

3) Precision (P): P is to measure how much the recommended items are in the ground-true item set: $P = \sum_{i=1}^{n} \frac{|A_i \cap A_i^*|}{n}$.

4) Click Ratio (CR): CR measures whether the item is clicked by the user: HR $= \sum_{i=1}^{n}\sum_{j}^{m} \mathbb{1}\left(a_{ij}^* \in A_{ii}\right)/(n*m)$, i, j means the i-th user with j-th recommended items.

5) Browsing Length (BL): BL is to estimate how long a user interacts with the RS.

6) Cost Action (CA). cost action is to judge the number of actions that imply costs ($\mathbb{1}\left(\sum_j c_{ij} < 0\right)/n$): UA $= \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbb{1}\left(\sum_j c_{ij} < 0\right)/n$.

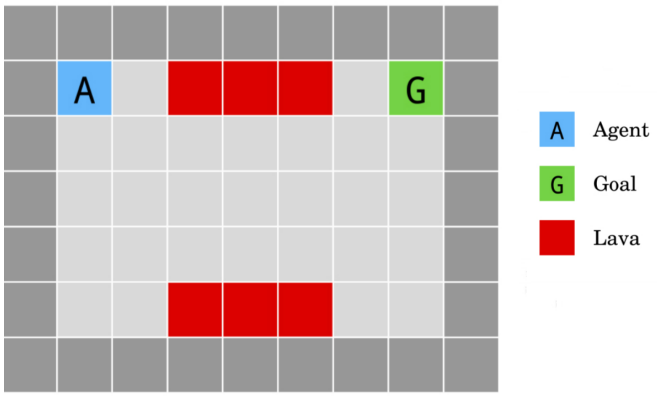


Figure 5: The Distributional Shift I environment. The figure is reproduced from (Leike et al. 2017).

7) Unsafe Number (UN). UN aims to evaluate the number that the *trajectory* is unsafe: UR $= \sum_{i=1}^{n} \mathbb{1}\left(\sum_j c_j < \epsilon\right)$.

Simulator. As we mentioned in the main text, both the Sequential Recommendation I and Sequential Recommendation II are based on NN. The input of the simulator in RS I are the recommended features as well as the feature of users (as shown in

[12] https://pytorch.org/

Table 3: The training time (GPU hours) for different methods.

	RSC-DQN	S2C-DQN	DQN	SC-DQN
Training time (hours)	2.41	3.85	2.41	3.97

Table 4: The hyper-parameters for Sequential Recommendation I, Sequential Recommendation II, and Safe GridWorld.

Parameters	Sequential Recommendation I	Sequential Recommendation II	Safe GridWorld
Batch size	1024	1024	1024
Learning Rate	0.0001	0.0001	0.0001
Buffer Size	40000	40000	40000
Discount rate γ	0.99	0.99	0.99
Qdiscount rate	0.99	0.99	0.99
α prioritised replay	0.6	0.6	0.6
β prioritised replay	0.1	0.1	0.1
units for each hidden layer	30,15	30,15	30,15
gradient clipping norm	0.7	0.7	0.7
update every n steps	n=1	n=1	n=1
ϕ	1e-3	1e-3	1e-3

Tab. 5 and the output is the score of the items. For RS II, the input is the user profile, the sequence information (some statistical information, including how many times user click and view) and recommended item, while the output is the user feedback (click, view or leave). Moreover, for the network structure, each FC layer consists of a linear transformation with RELU as activation function (Nair and Hinton 2010). Notice that the user behavior simulators in both RS I and RS II do not perform well when the network structure is simple. Therefore, for each features, we use different embedding layers to extract each of the feature. Then we concatenate all the embeddings together and further use two FC layers to get the output. As we mentioned in the main text, we use the resampling tricks in RS II to avoid sparseness. Regarding the loss function, we use mean squared error in RS I and cross entropy error in RS II. The MSE on test set is 1.3 in RS I while the AUC is 0.60 and the accuracy is 0.81 in RS II. This result illustrates the NN based simulator can simulate the true users' behaviors well.

Ablations. For ablations, we choose different ϕ to evaluate the effects of count-based exploration, including 1) RSC-DQN 1e-0: RSC-DQN with $\phi = 1$ 2) RSC-DQN 1e-3: RSC-DQN with $\phi = 10^{-3}$ (our default setting); and 3) RSC-DQN 0, RSC-DQN 0 is equal to the RSC-DQN.

Simhash. For the Sim-hash, we use the code from[13].

E.2 Analysis.

This is an extension of Sec. 7.5. From Fig. 8, we can find that our methods are well-performing in almost all the metrics. Also, the reason that average HR is not high in Tab. 2 is because our methods converage slower than other SOTAs, inducing lower HR in the early stage. Fig. 7 reveals that our methods also perform well regarding the shaping reward. But the rewarding shaping needs delicate reward design, our method can achieve similar scores without this time-consuming design process. In Fig. 6 and Tab. 7, we can find that expect for DQN, all the methods have low cost actions, indicating that these safe methods are all able to prevent cost actions. Among these methods, ours have lower cost actions than others, which means our methods are more safe than others.

Fig. 9 and Tab. 8 illustrate that the count-based module does play an important role in terms of performance. However, too large ϕ does harm to the agent. We can find that too large ϕ causes large unsafe ratio because too large ϕ bans all possible actions (includes the safe actions) and when all the actions are banned, the agent will randomly choose an action from the action set. This is the main reason that too large ϕ ironically intensifies the unsafety. Moreover, the phenomenon that too large actions induce higher reward is due to the fact that some actions are dangerous but leads to high reward. Moreover, we find that $\phi = 0$ can reduce the training time while reach similar performance. This motivates us to relax this term.

From Tab. 3, the training time for RSC-DQN is far less than S2C-DQN. This phenomenon is caused by the fact that counting the occurrence of each states is time-consuming. Combining Tab. 2, since RSC-DQN and S2C-DQN have similar performance, we conclude that RSC-DQN may be more sample efficiency.

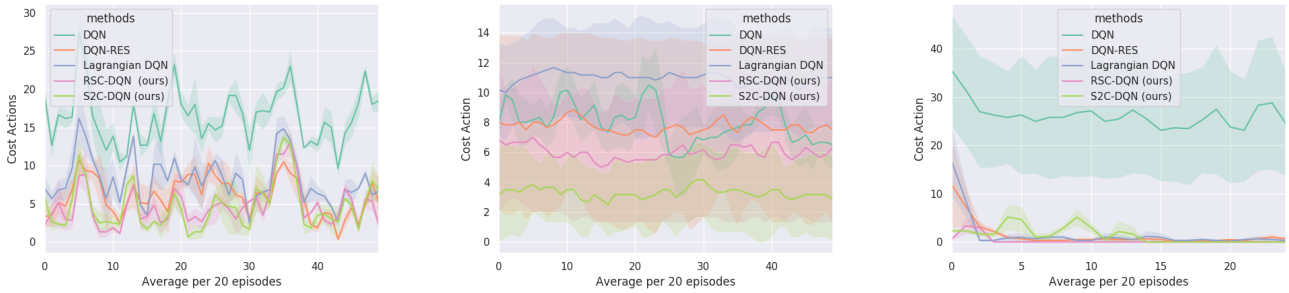
[13]https://github.com/openai/EPG

Table 5: Statistics of MovieLens-1M and Production dataset.

	MovieLens-1M	**Production dataset**
Number of the users	5940	6591
Item numbers	3883	45067
Features for items	genre, direct, actor	ctr, cvr, price, logctr, logcvr, logprice, ctrcvr, cvrprice, ctrcvrprice.
Features for users	state, country, age	gender, age, occupation, zipcode, state, country

Table 6: The average unsafe ratio in different environments. SR means sequential recommendation. The best results are highlighted in **bold**. The mean and standard deviation are reported by 3 independent trials.

Metric	**Total Unsafe Number** $\downarrow$		
Methods	**SR I**	**SR II**	**Gridworlds**
DQN	180.44$\pm$0.03	658.33$\pm$1.26	226.91$\pm$0.03
DQN-RES	57.94$\pm$0.01	371.94$\pm$0.19	220.75$\pm$0.07
Lagrangian DQN	80.26$\pm$0.01	560.76$\pm$0.14	306.37$\pm$0.07
RSC-DQN (ours)	**37.27**$\pm$0.00	320.65$\pm$0.14	186.04$\pm$0.06
S2C-DQN (ours)	42.71$\pm$0.02	**185.26**$\pm$0.08	**171.69**$\pm$0.07



(a) Cost Actions in RS I. (b) Cost Action in RS II. (c) Cost Action in Safe GridWorld.

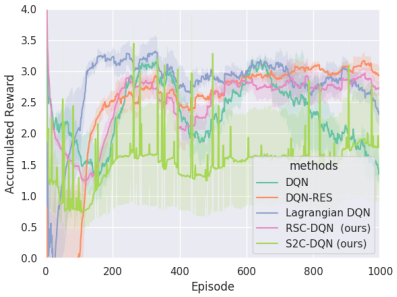
Figure 6: The cost action of various environments.

Table 7: The number of cost actions for different methods. The best results are highlighted in **bold**. The mean and standard deviation are reported by 3 independent trials.

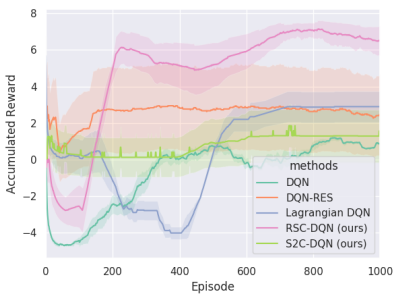
Metric	**Average Cost Actions** $\downarrow$		
Environment	**SR I**	**SR II**	**Gridworlds**
DQN	820.33	399.67	160.83
DQN-RES	300.00	386.50	58.50
Lagrangian DQN	408.83	548.16	30.50
RSC-DQN (ours)	235.33	299.16	**25.17**
S2C-DQN (ours)	**235.16**	**164.50**	28.67

Table 8: Performances of different methods in SR I. The best results are highlighted in **bold**. The mean is reported by 3 independent trials.

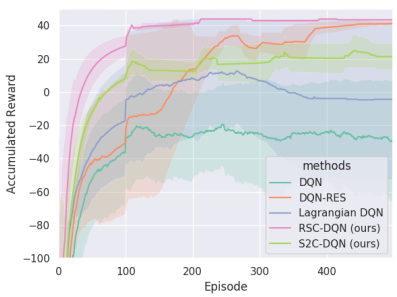
Environment	Sequential Recommendation I			
Methods	**RSC-DQN 1e-0**	**SC-DQN**	**S2C-DQN 1e-3**	**S2C-DQN 0**
Avg Unsafe Number $\downarrow$	86.83	947.33	**37.27**	42.71
Avg Precision $\uparrow$	0.966	0.266	**0.976**	0.974
Avg Click Ratio $\uparrow$	**0.295**	0.208	0.281	0.273



(a) Shaping Reward in RS I.

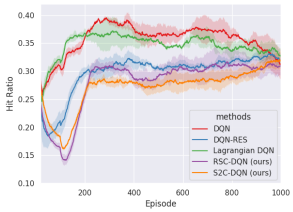


(b) Shaping Reward in RS II.

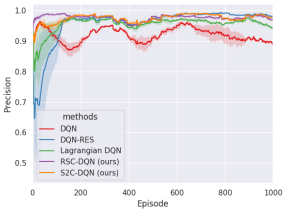


(c) Shaping Reward in Safe GridWorld.

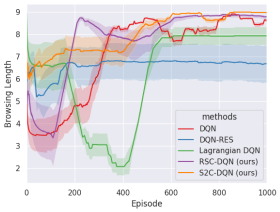
Figure 7: The shaping reward of various environments.



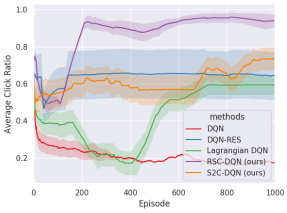
(a) Click Ratio in RS I.



(b) Precision in RS I.

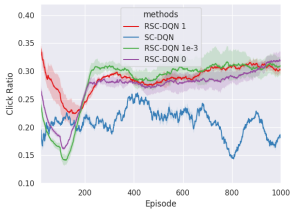


(c) Browsing History in RS II.

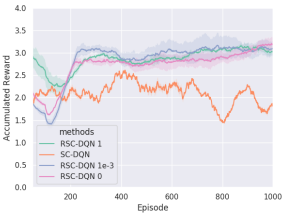


(d) Click Ratio in RS II.

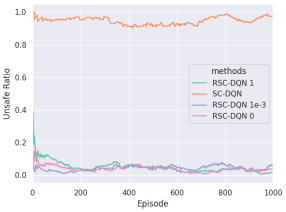
Figure 8: The metrics of different methods in each step.



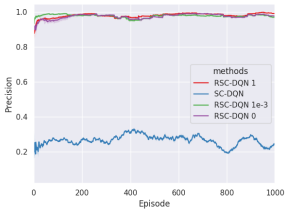
(a) Click Ratio in RS I.



(b) Reward in RS I.



(c) Unsafe Ratio in RS I.



(d) Precision in RS I.

Figure 9: The ablation studies in RS I.