Deployment Efficient Reward-Free Reinforcement Learning in Linear MDPs

Dan Qiao, Yu-Xiang Wang

Introduction and Preliminaries

◇ Motivation of deployment efficiency

In many real-world reinforcement learning (RL) tasks, it is costly to run fully adaptive algorithms that update the exploration policy frequently. Instead, collecting data in large batches using the current policy deployment is usually cheaper.

◇ Some examples

- Recommendation systems
- Healthcare
- Database optimization
- Computer networking
- New material design

◇ Goal of this work

Design an algorithm that can minimize the number of policy deployments while maintaining (nearly) the same sample complexity as its fully-adaptive counterparts.

◇ Problem setup

Linear MDP with feature map ϕ whose dimension is d and planning horizon H.

Assumption on reachability For policy π, we define $\Lambda_{\pi,h} := \mathbb{E}_\pi[\phi(s_h,a_h)\phi(s_h,a_h)^\top]$. We assume $\lambda^\star = \min_{h\in[H]}\sup_\pi \lambda_{\min}(\Lambda_{\pi,h}) > 0$.

Policy We consider non-Markovian policies.

Reward-free RL The procedure contains two steps:

1. Exploration phase: Given ϵ and δ, the learner explores an MDP for $K(\epsilon,\delta)$ episodes and collects data without rewards $\{s_h^k, a_h^k\}_{(h,k)\in[H]\times[K]}$.
2. Planning phase: The learner outputs a function $\widehat{\pi}(\cdot)$ which takes reward function as input. The function $\widehat{\pi}(\cdot)$ satisfies that for any valid reward function r, $V^{\widehat{\pi}(r)}(r) \geq V^\star(r) - \epsilon$.

The goal of reward-free RL is to design a procedure that satisfies the above guarantee with probability at least $1-\delta$ while collecting as few episodes as possible.

◇ Deployment complexity

We say that an algorithm has deployment complexity of M, if the algorithm is guaranteed to finish running within M policy deployments. In each deployment, the algorithm is allowed to deploy non-Markovian policies.

◇ Questions of interest

Can we achieve near optimal sample complexity and near optimal deployment complexity at the same time?

Algorithms and Results

◇ Algorithmic design.

Algorithm 1 Layer-by-layer Reward-Free Exploration via Experimental Design (Exploration)

```
1: Input: Accuracy $\epsilon$. Failure probability $\delta$.
2: Initialization: $\iota = \log(dH/\epsilon\delta)$. Error budget for each layer $\bar{\epsilon} = \frac{C_1\epsilon}{H^2\sqrt{d\cdot\iota}}$. Construct $\Pi^{exp}_{\epsilon/3}$ as in Section 3.2. Number of episodes for each deployment $N = \frac{C_2 d\iota}{\bar{\epsilon}^2} = \frac{C_2 d^2 H^4 \iota^3}{C_1^2\epsilon^2}$. Dataset $\mathcal{D} = \emptyset$.
3: for $h = 1, 2, \cdots, H$ do
4:     Solve the following optimization problem.
5:     $$\pi_h = \operatorname*{argmin}_{\pi\in\Delta(\Pi^{exp}_{\epsilon/3})\ \text{s.t.}\ \lambda_{\min}(\widehat{\Sigma}_\pi)\geq C_3 d^2 H\bar{\epsilon}\iota}\ \max_{\widehat{\pi}\in\Pi^{exp}_{\epsilon/3}} \widehat{\mathbb{E}}_{\widehat{\pi}}\left[\phi(s_h,a_h)^\top(N\cdot\widehat{\Sigma}_\pi)^{-1}\phi(s_h,a_h)\right], \quad (1)$$
6:     where $\widehat{\Sigma}_\pi$ is $\widehat{\mathbb{E}}_\pi[\phi(s_h,a_h)\phi(s_h,a_h)^\top] = \mathsf{EstimateER}(\pi, \phi(s,a)\phi(s,a)^\top, A=1, h, \mathcal{D}, s_1)$,
7:     $\widehat{\mathbb{E}}_{\widehat{\pi}}\left[\phi(s_h,a_h)^\top(N\cdot\widehat{\Sigma}_\pi)^{-1}\phi(s_h,a_h)\right] = \mathsf{EstimateER}(\widehat{\pi}, \phi(s,a)^\top(N\cdot\widehat{\Sigma}_\pi)^{-1}\phi(s,a), A = \frac{\bar{\epsilon}}{C_2 d^3 H\iota^2}, h, \mathcal{D}, s_1)$.   // Both expectations are estimated via Algorithm 4.
8:     for $n = 1, 2, \cdots, N$ do
9:         Run $\pi_h$ and add trajectory $\{s_i^n, a_i^n\}_{i\in[H]}$ to $\mathcal{D}$.   // Run Policy $\pi_h$ for $N$ episodes.
10:    end for
11: end for
12: Output: Dataset $\mathcal{D}$.
```

Algorithm 2 Find Near-Optimal Policy Given Reward Function (Planning)

```
1: Input: Dataset $\mathcal{D}$ from Algorithm 1. Feasible linear reward function $r = \{r_h\}_{h\in[H]}$.
2: Initialization: Construct $\Pi^{eval}_{\epsilon/3}$ as in Section 3.2.   // The set of policies to evaluate.
3: for $\pi \in \Pi^{eval}_{\epsilon/3}$ do
4:     $\widehat{V}^\pi(r) = \text{EstimateV}(\pi, r, \mathcal{D}, s_1)$.   // Estimate value functions using Algorithm 3.
5: end for
6: $\widehat{\pi} = \arg\max_{\pi\in\Pi^{eval}_{\epsilon/3}} \widehat{V}^\pi(r)$.   // Output the greedy policy w.r.t $\widehat{V}^\pi(r)$.
7: Output: Policy $\widehat{\pi}$.
```

◇ Key components.

1. Design of policy sets.

Policy sets	*Cardinality*	*Description*	*Relationship with each other*
The set of all policies	Infinity	The largest possible policy set	Contains the following two sets
Explorative policies: Π^{exp}_ϵ	$\log\lvert\Pi^{exp}_{\epsilon,h}\rvert = \widetilde{O}(d^2)$	Sufficient for exploration	Subset of all policies
Policies to evaluate: Π^{eval}_ϵ	$\log\lvert\Pi^{eval}_{\epsilon,h}\rvert = \widetilde{O}(d)$	Uniform policy evaluation over Π^{eval}_ϵ is sufficient for policy identification	Subset of Π^{exp}_ϵ

Table 2: Comparison of different policy sets.

2. Generalized G-optimal design

Theorem 1 *If there exists policy $\pi_0 \in \Delta(\Pi)$ such that $\lambda_{\min}(\mathbb{E}_{\pi_0}\phi_h\phi_h^\top) > 0$, then the following is bounded by d:* $\min_{\pi_0\in\Delta(\Pi)}\max_{\pi\in\Pi}\mathbb{E}_\pi\phi(s_h,a_h)(\mathbb{E}_{\pi_0}\phi_h\phi_h^\top)^{-1}\phi(s_h,a_h)$.

3. Estimating expectation/value function based on LSVI (without optimism): EstimateV and EstimateER.

◇ Comparison to previous works.

Algorithms for reward-free RL	*Sample complexity*	*Deployment complexity*
Algorithm 1 & 2 in Wang et al. [2020]	$\widetilde{O}(\frac{d^3H^6}{\epsilon^2})$	$\widetilde{O}(\frac{d^3H^6}{\epsilon^2})$
FRANCIS [Zanette et al., 2020b]‡	$\widetilde{O}(\frac{d^3H^5}{\epsilon^2})$	$\widetilde{O}(\frac{d^3H^5}{\epsilon^2})$
RFLIN [Wagenmaker et al., 2022b]‡	$\widetilde{O}(\frac{d^2H^5}{\epsilon^2})$	$\widetilde{O}(\frac{d^2H^5}{\epsilon^2})$
Algorithm 2 & 4 in Huang et al. [2022]‡	$\widetilde{O}(\frac{d^3H^5}{\epsilon^2\nu_{\min}^2})^*$	H
LARFE [Qiao et al., 2022]†	$\widetilde{O}(\frac{S^2AH^5}{\epsilon^2})$	$2H$
Our Algorithm 1 & 2 (Theorem 5.1)‡	$\widetilde{O}(\frac{d^2H^5}{\epsilon^2})$	H
Our Algorithm 1 & 2 (Theorem 7.1)⋆	$\widetilde{O}(\frac{S^2AH^5}{\epsilon^2})$	H
Lower bound [Wagenmaker et al., 2022b]	$\Omega(\frac{d^2H^2}{\epsilon^2})$	N.A.
Lower bound [Huang et al., 2022]	If polynomial sample	$\widetilde{\Omega}(H)$

◇ Future extensions.

1. Improve the computational efficiency.
2. Remove the assumption on reachability.