

# Reward-Free Exploration for Reinforcement Learning

Chi Jin

Princeton University  
chij@princeton.edu

Akshay Krishnamurthy

Microsoft Research, New York  
akshay@cs.umass.edu

Max Simchowitz

University of California, Berkeley  
msimchow@berkeley.edu

Tiancheng Yu

Massachusetts Institute of Technology  
yutc@mit.edu

February 10, 2020

## 【强化学习 110】Reward-Free Exploration



张楚琦

清华大学 交叉信息院博士在读

39 人赞同了该文章

这篇文章提出了一个比较新的设定，先在状态空间探索，然后求解 MDP 问题。文章给出了在这种设定下的 provably efficient 算法和相应的 lower bound。

### 原文传送门

Jin, Chi, et al. "Reward-Free Exploration for Reinforcement Learning." arXiv preprint arXiv:2002.02794 (2020).

### 特色

从 model-based 的角度来看，强化学习问题中最困难的不在于估计 reward，而在于估计 transition function。并且，如果想要以较高的精度估计到 transition matrix 中的每一个格子的数值（e.g., in terms of total variation）是几乎不可能的。从 model-free 的角度来看，这个问题就是要找一个好的具有探索性的策略，而这里的探索性指的是在状态空间（或者状态-行动空间）上的探索。

这篇文章提出了解决强化学习问题的一个范式：先进行探索（exploration phase），再进行规划求解（planning phase）。在探索的阶段不接受奖励信息，只是在状态空间上纯探索，以此得到一个探索性的策略，并且执行该策略得到一个数据集  $D$ 。在规划阶段，对于任意一个给定的奖励函数  $r$ ，利用数据集  $D$  估计出来的 transition function，应用标准的强化学习方法求解到好的策略。本文结合一些之前的强化学习算法的理论保证，给出了一个算法使得该算法在这个设定上 probably efficient。

### 过程

#### 1、背景和相关文献

首先，这种强化学习的范式在某些方面具有显著的优势：通过探索得到这样一个数据集  $D$  之后，我们就可以在不对 MDP 进行任何的采样的情况下，针对任意的奖励函数都找到接近最优的策略。这在某些情况下特别有用，比如，我们有一个期望的智能体的 behavior，但是需要设计一个奖励函

数使得智能体达到这样的 behavior。（还可以参考 meta learning 等设定）

其次，在技术上，该问题的实质是强化学习里面最为困难的一个问题：对于状态空间的探索问题。而且在该范式下，直接分离出来该问题，在 exploration phase 中解决。

在相关的文献上：RMax [Brafman and Tenenbholz, 2002] 可以被改造成 reward-free 的形式，但是样本效率比较低；其他针对特定 reward 的算法得到的 PAC 算法，肯定不一定对于任意的 reward 最优；有一些适用 function approximation 的探索方法 [Du et al 2019, Misra et al 2019]，这两篇我还不太熟；还有一篇比较类似的，[Hazan et al 2019] (ICML 19')，也是一个 reward-free 的常见（不太记得专栏有没有写了），证明了他们的算法能够达到一个 max-entropy 的策略（即，在状态空间上是 max-entropy），但是没有说明探索得到的“成果”如何转化为相对于任意奖励函数的最优策略。当时读这一篇的时候就觉得肯定得填上这一环，但是主要的技术原因在于 max-entropy 并不保证“每个”状态都被较好地访问到；而这篇文章观察到，有些实在怎么着都访问不太到的状态其实也不是很影响最后的 value function，因此把不去过多地访问它们也没事。

## 2、Reward-free setting

文章提出的这个范式，在第一个探索阶段只做 reward-free 的探索，这个交互和标准的 RL 交互的区别就在于环境不返回奖励。相比于标准 RL，其他方面都一样，比如都具有一个固定的初始状态分布，并且要从该分布出发根据 transition dynamics 来访问各个状态。

---

### Protocol 1 Reward-Free Exploration

---

```
for  $k = 1$  to  $K$  do
  learner decides a policy  $\pi_k$ 
  environment samples the initial state  $s_0 \sim \mathbb{P}_1$ .
  for  $h = 1$  to  $H$  do
    learner selects action  $a_h \sim \pi_h(\cdot|s_h)$ 
    environment transitions to  $s_{h+1} \sim \mathbb{P}_h(\cdot|s_h, a_h)$ 
    learner observes the next state  $s_{h+1}$ 
```

---

知乎 @张楚珩

## 3、Overview

**Algorithm overview.** Our algorithm proceeds with following high level steps:

1. learn a set of policies  $\Psi$  which allow us to visit all “significant” states with reasonable probability.
2. collect a sufficient amount of data by executing policies in  $\Psi$ .
3. compute the empirical transition matrix  $\hat{\mathbb{P}}$  using the collected data.
4. for each reward function  $r$ , find a near-optimal policy by invoking a planning algorithm with transitions  $\hat{\mathbb{P}}$  and reward  $r$ .

知乎 @张楚珩

其中，第 1、2 步就是我们前面讲到的 exploration phase；而 3、4 步就是 planning phase。

最后文章证明了该算法只需要在 exploration phase 做这么多个轨迹的探索，就能在 planning phase 对于任意的奖励函数都找到 epsilon-optimal policy。

**Theorem 3.1.** *There exists an absolute constant  $c > 0$  and a reward-free exploration algorithm such that, for any  $p \in (0, 1)$ , with probability at least  $1 - p$ , the algorithm outputs  $\epsilon$ -optimal policies for an arbitrary number of adaptively chosen reward functions. The number of episodes collected in the exploration phase is bounded by*

$$c \cdot \left[ \frac{H^5 S^2 A \iota}{\epsilon^2} + \frac{S^4 A H^7 \iota^3}{\epsilon} \right], \quad (3)$$

where  $\iota := \log(SAH/(p\epsilon))$ .

知乎 @张楚珩

#### 4、Exploration phase

这个阶段的通过智能体和环境的 reward-free 交互，找到一个具有探索性的策略，然后输出一个使用该策略得到的采样数据集。算法如下图所示。其中 3-7 行是在学习得到一个探索性的策略集合，具体的方式是对于每一个状态，都构造一个只在该状态上有奖励的奖励函数，然后使用已有的 RL 算法来得到能够把智能体带到该状态的策略集合。（注意到带到第  $h$  层的状态  $s$  之后，再从该状态出发，选择什么 action 都无所谓了，因此第 6 行把 Euler 得到的策略稍微又改了一下）接下来第 8-11 行则执行该策略族，得到一个数据集  $\mathcal{D}$ 。

---

##### Algorithm 2 Reward-free RL-Explore

---

```

1: Input: iteration number  $N_0, N$ .
2: set policy class  $\Psi \leftarrow \emptyset$ , and dataset  $\mathcal{D} \leftarrow \emptyset$ .
3: for all  $(s, h) \in \mathcal{S} \times [H]$  do
4:    $r_{h'}(s', a') \leftarrow \mathbb{I}[s' = s \text{ and } h' = h]$  for all  $(s', a', h') \in \mathcal{S} \times \mathcal{A} \times [H]$ .
5:    $\Phi^{(s, h)} \leftarrow \text{EULER}(r, N_0)$ .
6:    $\pi_h(\cdot | s) \leftarrow \text{Uniform}(\mathcal{A})$  for all  $\pi \in \Phi^{(s, h)}$ .
7:    $\Psi \leftarrow \Psi \cup \Phi^{(s, h)}$ .
8: for  $n = 1 \dots N$  do
9:   sample policy  $\pi \sim \text{Uniform}(\Psi)$ .
10:  play  $\mathcal{M}$  using policy  $\pi$ , and observe the trajectory  $z_n = (s_1, a_1, \dots, s_H, a_H, s_{H+1})$ .
11:   $\mathcal{D} \leftarrow \mathcal{D} \cup \{z_n\}$ 
12: Return: dataset  $\mathcal{D}$ .
```

---

知乎 @张楚珩

我们前面提到，对于有些实在访问不到的状态（无论什么策略都访问不到），其实可以直接把它们放弃掉。因为，即使这些状态上的奖励函数很高，但是我们怎么控制也达不到这些状态，因此最后的最优策略也不可能特别地去访问这些状态。我们记这部分状态为 *insignificant*，具体定义如下：

**Definition 3.2.** A state  $s$  in step  $h$  is  $\delta$ -**significant** if there exists a policy  $\pi$ , so that the probability to reach  $s$  following policy  $\pi$  is greater than  $\delta$ . In symbol:

$$\max_{\pi} P_h^{\pi}(s) \geq \delta$$

文章证明思路的关键就在于，通过该 exploration 得到的数据集对于任意 significant 的状态，都能够以比较大的概率访问到。

**Theorem 3.3.** *There exists absolute constant  $c > 0$  such that for any  $\epsilon > 0$  and  $p \in (0, 1)$ , if we set  $N_0 \geq cS^2AH^4\iota_0^3/\delta$  where  $\iota_0 := \log(SAH/(p\delta))$ , then with probability at least  $1 - p$ , that Algorithm 2 will returns a dataset  $\mathcal{D}$  consisting of  $N$  trajectories  $\{z_n\}_{n=1}^N$ , which are i.i.d sampled from a distribution  $\mu$  satisfying:*

$$\forall \delta\text{-significant } (s, h), \quad \max_{a, \pi} \frac{P_h^\pi(s, a)}{\mu_h(s, a)} \leq 2SAH. \quad \text{知乎 @张楚珩(4)}$$

文章之所以要用 Euler 是因为该算法得到的策略集的性能和最优价值函数的大小有关，即，如果这个最优价值函数本身就不特别好（比如，即使最优策略也访问不到那些 reward 比较大但是 insignificant 的状态），那么得到的策略和最优价值函数的差距也不会特别大。这一点对于本文的证明比较关键。

**Lemma 3.4.** *There exists absolute constant  $c > 0$  such that for any  $N_0 > 0$  and  $p \in (0, 1)$ , with probability at least  $1 - p$ , if we run EULER algorithm for  $N_0$  episodes, it will output a policy set  $\Phi$  with  $|\Phi| = N_0$  that satisfies:*

$$\mathbb{E}_{s_1 \sim \mathbb{P}_1} \left[ V_1^*(s_1) - \frac{1}{N_0} \sum_{\pi \in \Phi} V_1^\pi(s_1) \right] \leq c \cdot \left\{ \sqrt{\frac{SAH\iota_0 \cdot \mathbb{E}_{s_1 \sim \mathbb{P}_1} V_1^*(s_1)}{N_0}} + \frac{S^2AH^4\iota_0^3}{N_0} \right\}$$

where  $\iota_0 = \log(SAHN_0/p)$ .

知乎 @张楚珩

## 5、Planning phase

规划阶段的做法就比较简单了，就是从数据集里面估计 transition matrix 然后使用该 transition matrix 来求解。

---

### Algorithm 3 Reward-free RL-Plan

---

- 1: **Input:** a dataset of transition  $\mathcal{D}$ , reward function  $r$ , accuracy  $\epsilon$ .
  - 2: **for** all  $(s, a, s', h) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times [H]$  **do**
  - 3:    $N_h(s, a, s') \leftarrow \sum_{(s_h, a_h, s_{h+1}) \in \mathcal{D}} \mathbb{I}[s_h = s, a_h = a, s_{h+1} = s']$ .
  - 4:    $N_h(s, a) \leftarrow \sum_{s'} N_h(s, a, s')$ .
  - 5:    $\mathbb{P}_h(s'|s, a) = N_h(s, a, s')/N_h(s, a)$ .
  - 6:  $\hat{\pi} \leftarrow \text{APPROXIMATE-MDP-SOLVER}(\hat{\mathbb{P}}, r, \epsilon)$ .
  - 7: **Return:** policy  $\hat{\pi}$ .
- 

知乎 @张楚珩

证明上主要的技术就是把两方面的误差叠加起来：估计的 transition matrix 和真实 transition matrix 的误差，在估计的 transition matrix 上求解出来的策略和这上面最优策略之间的误差。

$$\begin{aligned}
& \mathbb{E}_{s_1 \sim \mathbb{P}_1} \{V_1^{\pi^*}(s_1; r) - V_1^{\hat{\pi}}(s_1; r)\} \\
& \leq \underbrace{\mathbb{E}_{s_1 \sim \mathbb{P}_1} \{V_1^{\pi^*}(s_1; r) - \hat{V}_1^{\pi^*}(s_1; r)\}}_{\text{Evaluation error 1}} + \underbrace{\mathbb{E}_{s_1 \sim \mathbb{P}_1} \{\hat{V}_1^{\pi^*}(s_1; r) - \hat{V}_1^{\hat{\pi}}(s_1; r)\}}_{\leq 0 \text{ by definition}} \\
& + \underbrace{\mathbb{E}_{s_1 \sim \mathbb{P}_1} \{\hat{V}_1^{\hat{\pi}}(s_1; r) - \hat{V}_1^{\pi}(s_1; r)\}}_{\text{Optimization error}} + \underbrace{\mathbb{E}_{s_1 \sim \mathbb{P}_1} \{\hat{V}_1^{\pi}(s_1; r) - V_1^{\hat{\pi}}(s_1; r)\}}_{\text{Evaluation error 2}}
\end{aligned}$$

知乎 @张楚珩

其中， $\pi_1$  表示初始状态分布，带 hat 的价值函数表示在估计的 MDP 上得到的价值函数，带 hat 的策略表示通过 APPROXIMATE-MDP-SOLVER 得到的策略。最后，可以得到如下定理

**Theorem 3.5.** *There exists absolute constant  $c > 0$ , for any  $\epsilon > 0$ ,  $p \in (0, 1)$ , assume dataset  $\mathcal{D}$  has  $N$  i.i.d. samples from distribution  $\mu$  which satisfies Eq.(4) with  $\delta = \epsilon / (2SH^2)$ , and  $N \geq cH^5S^2A/\epsilon^2$ , then*

*with probability at least  $1 - p$ , for any reward function  $r$  simultaneously, the output policy  $\hat{\pi}$  of Algorithm 3 is  $3\epsilon$ -suboptimal. That is:*

$$\mathbb{E}_{s_1 \sim \mathbb{P}_1} [V_1^*(s_1; r) - V_1^{\hat{\pi}}(s_1; r)] \leq 3\epsilon$$

文章再使用 NPG [Agarwal et al 2019]（专栏前面有讲过）关于 sample complexity 的结论，就可以得到最后的定理。

---

**Algorithm 4** Natural Policy Gradient (NPG)

---

- 1: **Input:** transition matrix  $\mathbb{P}$ , reward function  $r$ , stepsize  $\eta$ , iteration number  $T$ .
  - 2: initialize  $\pi_h^{(0)}(\cdot|s) \leftarrow \text{Uniform}(\mathcal{A})$  for all  $(s, h)$
  - 3: **for**  $t = 0, \dots, T-1$  **do**
  - 4:   evaluate  $Q_h^{\pi^{(t)}}(s, a)$  using Bellman equation Eq.(1) for all  $(s, a, h)$ .
  - 5:   update  $\pi_h^{(t+1)}(a|s) \propto \pi_h^{(t)}(a|s) \cdot \exp(\eta Q_h^{\pi^{(t)}}(s, a))$  for all  $(s, a, h)$ .
  - 6: **Return:** policy  $\pi^{(T)}$ .
- 

**Proposition 3.7.** *for any learning rate  $\eta$  and iteration number  $T$ , the output policy  $\pi^{(T)}$  of Algorithm 4 satisfies the following:*

$$\mathbb{E}_{s_1 \sim \mathbb{P}_1} [V_1^*(s_1) - V_1^{\pi^{(T)}}(s_1)] \leq \frac{H \log A}{\eta T} + \eta H^2$$

知乎 @张楚珩

## 6. Lower bound

这一部分没太看明白，但是暂时不关心，就只贴一下结论。大致上来说，lower bound 通过构造法来证明，本文的构造主要考虑一个情况，即各个状态都能差不多概率被访问到（都 significant），但是奖励是可以随机地只出现在任意一个 state-action 上面，因此在探索阶段需要把这些状态都照顾到，因此至少需要一些样本才可能达到最后的结果。



**Theorem 4.1.** Let  $C > 0$  be a universal constant. Then for  $A \geq 2$ ,  $S \geq C \log_2 A$ ,  $H \geq C \log_2 S$ , and any  $\epsilon \leq \min\{1/4, H/48\}$ , any reward-free exploration algorithm Alg which satisfies the guarantee of Theorem 3.7 with  $p = 1/2$  and accuracy parameter  $\epsilon$  must collect  $\Omega(S^2 A H^2 / \epsilon^2)$  trajectories in expectation. This is true even if Alg can return randomized or history-dependent (non-Markov) policies, and holds even if the rewards and transitions are identical across stages  $h$ .

## 7、ZeroMax

本文中的算法可以看做是根据 Euler 算法得到的；文章附录中还给出了一种根据 RMax 算法得到的适用于 reward-free exploration 的 ZeroMax 算法。该算法做法上也很直观，建立一个集合  $K$  表示见到过次数较多的状态。在每次迭代中，先构建一个 MDP 的估计，接着在该估计的 MDP 上求解策略，最后运行该策略进行采样。该估计的 MDP 分配奖励给访问次数较少的状态，从而激励策略能够尽可能访问到所有能访问到的状态。

$$\mathcal{K} := \{(s, h) : \forall a \in \mathcal{A}, N_h(s, a) \geq m\}$$

Now ZEROMAX explores as follows. In each episode  $i \in [N]$ , the agent has a known set  $\mathcal{K}_i$  and

1. builds an empirical MDP  $\hat{\mathcal{M}}_{i, \mathcal{K}_i}$  with parameters

$$\mathbb{P}_h(\cdot | s, a) = \begin{cases} \hat{\mathbb{P}}_{h,i}(\cdot | s, a) & \text{if } (s, h) \in \mathcal{K}_i \\ \mathbb{1}_{\{s' = s\}} & \text{otherwise} \end{cases} \quad r_h(s, a) = \begin{cases} 0 & \text{if } (s, h) \in \mathcal{K}_i \\ 1 & \text{otherwise} \end{cases} \quad (7)$$

where  $\hat{\mathbb{P}}_{h,i}$  is the empirical estimation of  $\mathbb{P}_h$  in the  $i$ -th episode.

2. computes  $\pi_i = \pi_{\hat{\mathcal{M}}_{i, \mathcal{K}_i}}^*$  on  $\hat{\mathcal{M}}_{i, \mathcal{K}_i}$  by value iteration.
3. samples a trajectory from the environment following  $\pi_i$ .
4. constructs  $\mathcal{K}_{i+1}$  for the next episode

知乎 @张楚珩

由于该算法的目标是希望能够 uniformly 地访问到各个状态，但是这其实浪费了很多精力在 insignificant 的状态上，因此最后的效率不如文章正文中提到的算法效率高。

## 8、MaxEnt exploration

针对 [Hazan et al 2019] (ICML 19') 提出的 MaxEnt 算法，这篇文章在附录中也把该算法最后推导到了本文 exploration + planning 的设定上，得到了相应的 sample complexity。最后结果表明，因为和 ZeroMax 类似的原因，该算法效率不高。

---

Remark: The key is to construct a exploration policy such that it can visit every significant state with a visitation probability that not differs too much from maximum possible visitation probability. Next, we can apply the theorems for NPG to obtain a sample complexity bound. Notice that there is a

distribution mismatch coefficient in the bound, which is exactly what we ensured in the exploration phase.

编辑于 2020-03-08

强化学习 (Reinforcement Learning)

深度学习 (Deep Learning)

机器学习

▲ 赞同 39 ▼

4 条评论

分享

喜欢

★ 收藏

...

文章被以下专栏收录



强化学习前沿  
读呀读paper

进入专栏