Provable Self-Play Algorithms for Competitive Reinforcement Learning

Yu Bai* Chi Jin†

February 25, 2020

【强化学习 109】Provable Self-play

张楚珩
清华大学 交叉信息院博士在读

针对 self-play 的强化学习问题，基于 UCB 的思想给出 provably efficient 算法。

原文传送门

Bai, Yu, and Chi Jin. "Provable Self-Play Algorithms for Competitive Reinforcement Learning." arXiv preprint arXiv:2002.04017 (2020).

特色

在实践中，很多强化学习方法应用到了 self-play 的方法来训练强化学习智能体，取得了很好的效果。比如本专栏前面介绍过的针对 AlphaGo、StarCraft 等任务的算法都用到了这种训练方法。这里把在相互竞争环境中的 self-play 形式化设定为一个 Markov game，然后针对这个问题提出了 provably efficient 的算法。具体来说，给出了

- 基于 UCB 思想的 PPAD-complete 复杂度的 regret $\tilde{O}(\sqrt{T})$ 的算法
- 基于 explore-then-exploit 思想的 polynomial 复杂度的 regret $\tilde{O}(T^{2/3})$ 的算法
- 给出了 lower bound，并且以上算法和 lower bound 直接存在一个 gap
- 给出了两种简化情形下能够 match lower bound 的算法

总体上来说，本文是第一个 self-play RL 上的 provably efficient 算法。

Table 1: Regret and PAC guarantees of the Algorithms in this paper for zero-sum Markov games.

Settings	Algorithm	Regret	PAC	Runtime
General Markov Game	VI-ULCB (Theorem 2)	$\tilde{O}(\sqrt{H^3S^2ABT})$	$\tilde{O}(H^4S^2AB/\epsilon^2)$	PPAD-complete
	VI-explore (Theorem 5)	$\tilde{O}((H^5S^2ABT^2)^{1/3})$	$\tilde{O}(H^5S^2AB/\epsilon^2)$	Polynomial
	Mirror Descent ($H=1$) (Rakhlin and Sridharan, 2013)	$\tilde{O}(\sqrt{S(A+B)T})$	$\tilde{O}(S(A+B)/\epsilon^2)$	
Turn-Based Markov Game	VI-ULCB (Corollary 4)	$\tilde{O}(\sqrt{H^3S^2(A+B)T})$	$\tilde{O}(H^4S^2(A+B)/\epsilon^2)$	
	Mirror Descent ($H=2$) (Theorem 10)	$\tilde{O}(\sqrt{S(A+B)T})$	$\tilde{O}(S(A+B)/\epsilon^2)$	
Both	Lower Bound (Corollary 7)	$\Omega(\sqrt{H^2S(A+B)T})$	$\Omega(H^2S(A+B)/\epsilon^2)$	

过程

1、相关工作

单智能体强化学习：tabular episodic case model-based method $\tilde{O}(\sqrt{H^3SAT})$ （Azar et al 2017) model-free method $\tilde{O}(\sqrt{H^3SAT})$ (Jin et al 2018)，而相应的lower-bound 为 $\Omega(\sqrt{H^3SAT})$ 。

Markov game（MG）：之前有较多的工作基于 known transition 和 known reward，即不是强化学习的设定。Wei et al 2017, Jia et al 2019, Sidford et al 2019 在强化学习设定上解决该问题，但是需要对于 MG 或者采样的方式做一些假设，从而某种程度上回避最为关键的探索问题。

Adversarial opponents：注意这一类问题和 MG 不一样，在这一类问题中对手可以是一个和智能体作对的另一个玩家，也可以是专门干扰智能体，比如改变智能体拿到的 reward。MG 中的对手对于 reward 有一个明确的目标，最后能达到一个纳什均衡；但是这类问题则不会达到纳什均衡。

2、问题设定

Tabular episodic zero-sum Markov game

Formally, we consider tabular episodic zero-sum Markov games of the form $\mathrm{MG}(H, \mathcal{S}, \mathcal{A}, \mathcal{B}, \mathbb{P}, r)$, where

- H is the number of steps in each episode.
- $\mathcal{S} = \cup_{h\in[H+1]}\mathcal{S}_h$, and $\mathcal{S}_h$ is the set of states at step h, with $\max_{h\in[H+1]}|\mathcal{S}_h| \le S$.
- $\mathcal{A} = \cup_{h\in[H]}\mathcal{A}_h$, and $\mathcal{A}_h$ is the set of actions of the max-player at step h, with $\max_{h\in[H]}|\mathcal{A}_h| \le A$.
- $\mathcal{B} = \cup_{h\in[H]}\mathcal{B}_h$, and $\mathcal{B}_h$ is the set of actions of the min-player at step h, with $\max_{h\in[H]}|\mathcal{B}_h| \le B$.
- $\mathbb{P} = \{\mathbb{P}_h\}_{h\in[H]}$ is a collection of transition matrices, so that $\mathbb{P}_h(\cdot|s,a,b)$ gives the distribution over states if action pair (a,b) is taken for state s at step h.
- $r = \{r_h\}_{h\in[H]}$ is a collection of reward functions, and $r_h : \mathcal{S}_h \times \mathcal{A}_h \times \mathcal{B}_h \to [0,1]$ is the deterministic reward function at step h. Note that we are assuming that rewards are in $[0,1]$ for normalization. [1]

注意到在本文的设定中，每一轮两个玩家同时给出其行动，并且一个玩家目标是最大化奖励，而另一个玩家的目标是最小化这个相同的奖励，因此是一个 zero-sum 博弈。多智能体强化学习里面有很多研究有 decentralized 的设定，这里没有这种，就是两个玩家同步学习。

Value function

相应地，可以定义价值函数。从 s 出发，两个玩家分别遵循相应的策略，最后获得的奖励和。注意到，对于 max player 来说，这个 value function 数值越大越好；对于 min player 来说，这个 value

function 的数值越小越好。

$$V_h^{\mu,\nu}(s) := \mathbb{E}_{\mu,\nu}\left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}, b_{h'})\middle| s_h = s\right].$$

相应有 Q 函数

$$Q_h^{\mu,\nu}(s,a,b) := \mathbb{E}_{\mu,\nu}\left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'}, b_{h'})\middle| s_h = s, a_h = a, b_h = b\right].$$

V 和 Q 之间的关系以及 Bellman 方程

$$Q_h^{\mu,\nu}(s,a,b) = (r_h + \mathbb{P}_h V_{h+1}^{\mu,\nu})(s,a,b), \tag{1}$$
$$V_h^{\mu,\nu}(s) = \sum_{a,b} \mu_h(a|s)\nu_h(b|s) Q_h^{\mu,\nu}(s,a,b). \tag{2}$$

where we define $V_{H+1}^{\mu,\nu}(s) = 0$ for all $s \in \mathcal{S}_{H+1}$

Best response

注意到这里有两个玩家，当一个玩家策略固定的时候，对于另外一个玩家，就存在一种最优的策略。比如对于 max player 的一个固定策略 μ，那么存在一个 min player 的 best response $\nu^\dagger(\mu)$，满足

$V_h^{\mu,\nu^\dagger(\mu)}(s) = \inf_\nu V_h^{\mu,\nu}(s)$ for any (s,h). For simplicity, we denote $V_h^{\mu,\dagger} := V_h^{\mu,\nu^\dagger(\mu)}$.

考虑对手每次都给出相对于自己的 best response，那么自己也能找到一个最优的策略，使得对手是最优策略的时候，自己也能最大化自己的利益。记这种情况下，自己的策略为 $\mu^\star$：

$$V_h^{\mu^\star,\dagger}(s) = \sup_\mu V_h^{\mu,\dagger}(s), \qquad V_h^{\dagger,\nu^\star}(s) = \inf_\nu V_h^{\dagger,\nu}(s), \qquad \text{for all } (s,h).$$

根据 minmax theorem，有

$$\sup_\mu \inf_\nu V_h^{\mu,\nu}(s) = V_h^{\mu^\star,\nu^\star}(s) = \inf_\nu \sup_\mu V_h^{\mu,\nu}(s).$$

Regret

下面定义这种问题下的 regret，注意算法设计的目标就是要在给定 K 轮中 minimize regret。

Definition 1 (Regret). *For any algorithm that plays the Markov game for K episodes with (potentially adversarial) starting state s_1^k for each episode $k = 1, 2, \ldots, K$, the regret is defined as*

$$\mathrm{Regret}(K) = \sum_{k=1}^{K}\left[V_1^{\dagger,\nu^k}(s_1^k) - V_1^{\mu^k,\dagger}(s_1^k)\right],$$

where (μ^k, ν^k) denote the policies deployed by the algorithm in the k-th episode.

直接看这个定义可能不是很能够理解这个 regret 定义的含义，观察下面关系可以发现，该 regret 的含义就是每一轮每个玩家给出的策略的性能相对于该玩家可能达到的最好策略性能之间的损失和。某个玩家的策略性能需要用价值函数来衡量，而价值函数依赖于对手的策略，因此假设对手采取相对于自己策略的 best response。

$$\left[V_h^{\dagger,\hat{\nu}}(s) - \inf_{\nu} V_h^{\dagger,\nu}(s)\right] + \left[\sup_{\mu} V_h^{\mu,\dagger}(s) - V_h^{\hat{\mu},\dagger}(s)\right] = V_h^{\dagger,\hat{\nu}}(s) - V_h^{\hat{\mu},\dagger}(s). \tag{5}$$

注意到，根据 minmax theorem，中间两项相等，就消掉了。

Turn-based games

这个设定还可以概括 turn-based games。就是轮到玩家一需要操作的时候，认为另外一个玩家的动作空间大小为 1，并且把这个唯一能够操作的动作设置为一个哑变量。

3. Value iteration - upper/lower confidence bound (VI-ULCB)

本专栏前面讲过 Jin Chi 的 LinearMDP，其算法和这个类似，也是基于 UCB 的；其算法（或者 Azar et al 2017）可以被概括如下：

> **Algorithm** (UCBVI for single-player RL): Compute $\left\{Q_h^{\mathsf{up}}(s,a) : h,s,a\right\}$ based on estimated transition and optimistic (upper) estimate of reward, then play one episode with the greedy policy with respect to Q^{up}.

而这里有两个玩家，max player 希望价值函数越大越好，min player 希望价值函数越小越好；仿照这类似的思路，可以让 max player 照着 upper confidence bound 探索，让 min player 照着 lower confidence bound 探索，这样自然想到如下算法：

> **Proposal** (Naive two-player extension of UCBVI): Compute $\left\{Q_h^{\mathsf{up}}(s,a,b), Q_h^{\mathsf{low}}(s,a,b)\right\}$ based on estimated transition and {upper, lower} estimates of rewards, then play one episode where the max-player (μ) is greedy with respect to Q^{up} and the min-player (ν) is greedy with respect to Q^{low}.

但是仔细想来，这个初步的想法有一个问题，因为估计出来的 Q 函数不仅取决于一个玩家的策略，还需要取决于另外一个玩家的策略；因此，我们需要同时确定两个玩家的策略，即找到两个玩家的策略使得它们 jointly greedy。把这两个 Q 函数看做两个玩家的 payoff matrix，然后认为 jointly greedy 就是找到这个 general-sum game 中的纳什均衡：

$$(\mu_h(\cdot|s), \nu_h(\cdot|s)) = \textsc{Nash\_General\_Sum}(Q_h^{\mathsf{up}}(s,\cdot,\cdot), Q_h^{\mathsf{low}}(s,\cdot,\cdot))$$

for all (h,s), where NASH_GENERAL_SUM is a subroutine that takes two matrices $\mathbf{P},\mathbf{Q} \in \mathbb{R}^{A\times B}$, and returns the Nash equilibrium $(\phi,\psi) \in \Delta_A \times \Delta_B$ for general sum game, which satisfies

$$\phi^\top \mathbf{P}\psi = \max_{\tilde{\phi}} \tilde{\phi}^\top \mathbf{P}\psi, \quad \phi^\top \mathbf{Q}\psi = \min_{\tilde{\psi}} \phi^\top \mathbf{Q}\tilde{\psi}. \tag{6}$$

因此，该算法最终可以被概括为

> **Our Algorithm** (VI-ULCB): Compute $\left\{Q_h^{\mathsf{up}}(s,a,b), Q_h^{\mathsf{low}}(s,a,b)\right\}$ based on estimated transition and {upper, lower} estimates of rewards, along the way determining policies (μ,ν) by running the NASH_GENERAL_SUM subroutine on $(Q^{\mathsf{up}}, Q^{\mathsf{low}})$. Play one episode according to (μ,ν).

详细的算法框图如下

Algorithm 1 Value Iteration with Upper-Lower Confidence Bound (VI-ULCB)

Initialize: for any (s,a,b,h), $Q_h^{\mathsf{up}}(s,a,b) \leftarrow H$, $Q_h^{\mathsf{low}}(s,a,b) \leftarrow 0$, $N_h(s,a,b) \leftarrow 0$.
for episode $k = 1,\ldots,K$ **do**
  Receive s_1.
  for step $h = H, H-1, \ldots, 1$ **do**
    for $(s,a,b) \in \mathcal{S}_h \times \mathcal{A}_h \times \mathcal{B}_h$ **do**
      $t = N_h(s,a,b)$;
      if t=0 **then**
        $Q_h^{\mathsf{up}}(s,a,b) \leftarrow H, \quad Q_h^{\mathsf{low}}(s,a,b) \leftarrow 0.$
      else
        $Q_h^{\mathsf{up}}(s,a,b) \leftarrow \min\{\hat{r}_h(s,a,b) + [\widehat{\mathbb{P}}_h V_{h+1}^{\mathsf{up}}](s,a,b) + \beta_t, H\}$
        $Q_h^{\mathsf{low}}(s,a,b) \leftarrow \max\{\hat{r}_h(s,a,b) + [\widehat{\mathbb{P}}_h V_{h+1}^{\mathsf{low}}](s,a,b) - \beta_t, 0\}$
    for $s \in \mathcal{S}_h$ **do**
      $(\mu_h(\cdot|s), \nu_h(\cdot|s)) \leftarrow \text{NASH\_GENERAL\_SUM}(Q_h^{\mathsf{up}}(s,\cdot,\cdot), Q_h^{\mathsf{low}}(s,\cdot,\cdot))$
      $V_h^{\mathsf{up}}(s) \leftarrow \sum_{a,b} \mu_h(a|s)\nu_h(b|s) Q_h^{\mathsf{up}}(s,a,b).$
      $V_h^{\mathsf{low}}(s) \leftarrow \sum_{a,b} \mu_h(a|s)\nu_h(b|s) Q_h^{\mathsf{low}}(s,a,b).$
  for step $h = 1,\ldots,H$ **do**
    Take action $a_h \sim \mu_h(s_h)$, $b_h \sim \nu_h(s_h)$.
    Observe reward r_h and next state s_{h+1}.
    $N_h(s_h,a_h,b_h) \leftarrow N_h(s_h,a_h,b_h) + 1.$
    $N_h(s_h,a_h,b_h,s_{h+1}) \leftarrow N_h(s_h,a_h,b_h,s_{h+1}) + 1$
    $\hat{\mathbb{P}}_h(\cdot|s_h,a_h,b_h) \leftarrow N_h(s_h,a_h,b_h,\cdot)/N_h(s_h,a_h,b_h).$
    $\hat{r}_h(s_h,a_h,b_h) \leftarrow r_h.$

4. 算法分析

Regret bound

该算法可以达到如下 regret

Theorem 2 (Regret bound for VI-ULCB). *For zero-sum Markov games, Algorithm 1 (with choice of bonus $\beta_t = c\sqrt{H^2 S\iota/t}$ for large absolute constant c) achieves regret*

$$\mathrm{Regret}(K) \le \mathcal{O}\left(\sqrt{H^3 S^2 \left[\max_{h\in[H]} A_h B_h\right] T\iota}\right) \le \mathcal{O}\left(\sqrt{H^3 S^2 ABT\iota}\right)$$

with probability at least $1-p$, where $\iota = \log(SABT/p)$.

证明思路和 Linear MDP 相似，关键步骤可以概括如下

A. 在每一层上，估计的 transition 距离真实 transition 的误差随着样本数目的增多而减小（concentration），相应的误差项刚好和给出的 exploration bonus 相匹配。

Lemma 12 (Uniform Concentration). *Consider value function class*

$$\mathcal{V}_{h+1} = \{V : \mathcal{S}_{h+1} \to \mathbb{R} \mid V(s) \in [0, H] \text{ for all } s \in \mathcal{S}_{h+1}\}.$$

There exists an absolute constant c, with probability at least $1-p$, we have:

$$\left|[(\hat{\mathbb{P}}_h^k - \mathbb{P}_h)V](s,a,b)\right| \le c\sqrt{SH^2\iota/N_h^k(s,a,b)} \quad \text{for all } (s,a,b,k,h) \text{ and all } V \in \mathcal{V}_{h+1}.$$

B. 在每一层上，由于加上了 exploration bonus β_t，估计出来的 upper 和 lower 大概率包含了真实价值函数的范围。其实质就是把前面的 concentration 展开写出来，注意到前一个 concentration 并没有规定价值函数得是实际遇到的价值函数（下面红色写出的），因此这个不等式也可以应用到相应的 sup 和 inf。

$$Q_h^{\mathsf{up},k}(s,a,b) \ge \sup_\mu Q_h^{\mu,\nu^k}(s,a,b) \ge \inf_\nu Q_h^{\mu^k,\nu}(s,a,b) \ge Q_h^{\mathsf{low},k}(s,a,b)$$

$\geq Q_h^{\mu^k,\nu^k}(s,a,b)$

C. 把 regret 中的每一项放缩为 upper 和 lower；然后从最后一层到第一层逐层递归，计算出地第一层的 regret bound。

$$\mathrm{Regret}(K) = \sum_{k=1}^K \left[\sup_\mu V_1^{\mu,\nu^k}(s_1^k) - \inf_\nu V_1^{\mu^k,\nu}(s_1^k)\right] \le \sum_{k=1}^K [V_1^{\mathsf{up},k}(s_1^k) - V_1^{\mathsf{low},k}(s_1^k)]$$

PAC bound

Regret bound 可以被转化为如下 PAC bound

Corollary 3 (PAC bound for VI-ULCB). *Suppose the initial state of Markov game is fixed at s_1, then there exists a pair of (randomized) policies $(\hat{\mu},\hat{\nu})$ derived through the VI-ULCB algorithm such that with probability at least $1-p$ (over the randomness in the trajectories) we have*

$$\mathbb{E}_{\hat{\mu},\hat{\nu}}\left[V^{\dagger,\hat{\nu}}(s_1) - V^{\hat{\mu},\dagger}(s_1)\right] \le \epsilon,$$

as soon as the number of episodes $K \ge \Omega(H^4 S^2 AB\iota/\epsilon^2)$, where $\iota = \log(HSAB/(p\epsilon))$ and the expectation is over the randomization in $(\hat{\mu},\hat{\nu})$.

Runtime

该算法由于需要用到一个 Nash_General_Sum 的子程序，而这个子程序不是 polynomial 的，因此该算法不是 polynomial 的。不过该算法在 MG 的一个特殊情形，即 turn-based game 的时候是 polynomial 的。

Turn-based game

在 MG 情形下，需要求解 Nash_General_Sum，它的输入是两个各自的 payoff 矩阵，需要求解出两个玩家各自的策略，而求解其纳什均衡没有 polynomial 解法

$$\phi^\top \mathbf{P}\psi = \max_{\tilde{\phi}} \tilde{\phi}^\top \mathbf{P}\psi, \quad \phi^\top \mathbf{Q}\psi = \min_{\tilde{\psi}} \phi^\top \mathbf{Q}\tilde{\psi}. \tag{6}$$

在 turn-based 情形下，每轮只有一个玩家可以做实质性的操作，因此只需要求解一个玩家的策略，相应的求解问题则变成了一个简单的 LP 问题

$$\phi^\top \mathbf{P} = \max_{\tilde{\phi}} \tilde{\phi}^\top \mathbf{P}$$

此时，该算法是 polynomial 的。

5. VI-Explore

这里算法设计的目的是找出来一个 polynomial time 的算法。我们观察到前面算法不是 polynomial time 的根本原因是要求解一个 general sum 的问题，在该问题中两个玩家有着不一样的 payoff matrix，这导致求解比较困难。在另一类简化的问题中，两个玩家有着一样的 payoff matrix，这时候该问题退化为 zero-sum 问题，该问题的求解有 polynomial time 的有效解法。前面产生两个不同 payoff matrix 的原因是需要估计 upper 和 lower，这里考虑直接估计一个 unbiased 的 transition matrix。

equilibrium $(\phi, \psi) \in \Delta_A \times \Delta_B$ for zero-sum game, which satisfies

$$\max_{\tilde{\phi}} \tilde{\phi}^\top \mathbf{Q}\psi = \phi^\top \mathbf{Q}\psi = \min_{\tilde{\psi}} \phi^\top \mathbf{Q}\tilde{\psi}. \tag{7}$$

VI-Explore 就是先用 Jin et al 2020 的一篇文章里面的探索方法，把 transition matrix 估计得足够准确（用一个较好的探索方法+concentration），然后直接基于该估计的 transition matrix 来做 value iteration，得到相应的纳什均衡。可以证明只要 transition matrix 估计的足够准确，相应的纳什均衡也会比较准确。显然，这里的探索方法不如 UCB 方法好，因此 regret 也会相应差一些。

PAC bound 如下：

Theorem 5 (PAC bound for VI-Explore). *With probability at least* $1-p$, REWARD_FREE_EXPLORATION(ϵ) *runs for* $c(H^5S^2AB\iota/\epsilon^2 + H^7S^4AB\iota^3/\epsilon)$ *episodes with some large constant* c, *and* $\iota = \log(HSAB/(p\epsilon))$, *and outputs* $(\hat{\mathbb{P}}, \hat{r})$ *such that the Nash equilibrium* $(\hat{\mu}, \hat{\nu})$ *of* $\mathrm{MG}(\hat{\mathbb{P}}, \hat{r})$ *satisfies*

$$\left[V^{\dagger,\hat{\nu}}(s_1) - V^{\hat{\mu},\dagger}(s_1)\right] \leq \epsilon.$$

转化为 regret bound 如下：

Corollary 6 (Polynomial time algorithm via explore-then-exploit). *For zero-sum Markov games, with probability at least $1-p$, Algorithm 2 runs in $\mathrm{poly}(S,A,B,H,T)$ time, and achieves regret bound*

$$\mathcal{O}\Big((H^5S^2ABT^2\iota)^{\frac{1}{3}}+\sqrt{H^7S^4ABT\iota^3}\Big),$$

where $\iota=\log(SABT/p)$.

6. Lower bound

之前有人给出了该问题的 lower bound

Corollary 7 (Regret lower bound, corollary of Jaksch et al. (2010), Theorem 5). *The regret[3] for any algorithm on turn-based games (and thus also general zero-sum games) is lower bounded by $\Omega(\sqrt{H^2S(A+B)T})$.*

这里说，在两种简化的情况下，已知的算法是能够达到该 lower bound 的：

Theorem 9 (Weak regret for one-step simultaneous game, adapted from Rakhlin and Sridharan (2013)). *For one-step simultaneous games ($H=1$), there exists a mirror descent type algorithm that achieves weak regret bound $\mathrm{WeakRegret}(T)\le\tilde{\mathcal{O}}(\sqrt{S(A+B)T})$ with high probability.*

Theorem 10 (Weak regret for two-step turn-based game). *For one-step turn-based games ($H=2$), there exists a mirror descent type algorithm that achieves weak regret bound $\mathrm{WeakRegret}(T)\le\tilde{\mathcal{O}}(\sqrt{SAT})$ with high probability.*

注意到这里用的是 weak regret，并且前面的 lower bound 也适用于该 weak regret。weak regret 的定义如下

Definition 8 (Weak Regret). *The* weak regret *for any algorithm that deploys policies (μ^k,ν^k) in episode k is defined as*

$$\mathrm{WeakRegret}(K):=\max_{\mu}\sum_{k=1}^{K}V^{\mu,\nu^k}(x_1^k)-\min_{\nu}\sum_{k=1}^{K}V^{\mu^k,\nu}(x_1^k). \tag{8}$$

注意到它和文章中 regret 的差别在于：weak regret 是 max(sum(·))，而 regret 是 sum(max(·))。显然，regret ≥ weak regret。

发布于 2020-03-06

计算机科学 强化学习 (Reinforcement Learning) 人工智能

文章被以下专栏收录

强化学习前沿
读呀读paper

进入专栏