

Provably Efficient Q-Learning with Low Switching Cost

Yu Bai¹, Tengyang Xie, Nan Jiang^{*2}, Yu-Xiang Wang^{*3}

¹Stanford University ²UIUC ³UC Santa Barbara

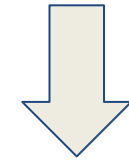
Draft poster, work in progress.

Motivation

In many RL domains, executing a new policy is expensive.

Off-policy RL: Find π_* given only off-line data from μ

Challenging... Relax



Limited Adaptivity RL: Find π_* with online data from $\{\mu_1, \dots, \mu_n\}$ for some **small n**

Local Policy Switch

Setup: episodic MDP with horizon H , play K episodes

Def: The **number of local policy switches** for an RL algorithm is

$$N_{\text{switch}} = \sum_{k=1}^{K-1} |\{(h, s) : \pi_h^k(s) \neq \pi_h^{k+1}(s)\}|$$

where π^k is the (deterministic) policy it plays at episode K .

Smaller N_{switch} \Rightarrow **closer to off-policy**

Prior work: Q-Learning with UCB-Hoeffding exploration^[1]:

$\tilde{O}(\sqrt{H^4 SAT})$ regret, but $N_{\text{switch}} = \Theta(HSK)$ **linear in K** 🤔

Any sublinear regret algo such that N_{switch} **sublinear in K**?

UCB2 for Bandits

Algorithm (UCB2):

- Select arm j that maximizes the UCB
- If this is the r -th time it's selected, play the arm exactly $\tau(r) - \tau(r-1)$ times, where

$$\tau(r) = (1 + \alpha)^r$$

Theorem^[2]: UCB2 achieves same regret as UCB & only **log(K) policy**

switches: $N_{\text{switch}} = O(A \log(K/A))$ 😊

Idea: Integrate UCB2 into Q-Learning!

Algorithm: Q-Learning with UCB2 Scheduling

Idea: update the policy according to Q only when Q has been updated $\tau(r) = (1 + \alpha)^r$ times.

Algorithm 2 Q-learning with UCB2 scheduling

input Parameter $\alpha \in (0, 1)$ and $c \geq 0$.

Initialize: $\tilde{Q}_h(x, a) \leftarrow H$, $Q_h \leftarrow \tilde{Q}_h$, $N_h(x, a) \leftarrow 0$ for all $(x, a, h) \in \mathcal{S} \times \mathcal{A} \times [H]$.

for episode $k = 1, \dots, K$ **do**

Receive x_1 .

for step $h = 1, \dots, H$ **do**

Take action $a_h \leftarrow \arg \max_{a'} Q_h(x_h, a')$, and observe x_{h+1} . // Take action according to Q

$t = N_h(x_h, a_h) \leftarrow N_h(x_h, a_h) + 1$; $b_t = c\sqrt{H^3 \ell / t}$.

$\tilde{Q}_h(x_h, a_h) \leftarrow (1 - \alpha_t)\tilde{Q}_h(x_h, a_h) + \alpha_t[r_h(x_h, a_h) + \tilde{V}_{h+1}(x_{h+1}) + b_t]$. // Update \tilde{Q} via Q-Learning

$\tilde{V}_h(x_h) \leftarrow \min \{H, \max_{a' \in \mathcal{A}} \tilde{Q}_h(x_h, a')\}$.

if $t = \tau(r)$ **for some** r **then**

(Update policy) $Q(x_h, \cdot) \leftarrow \tilde{Q}(x_h, \cdot)$.

// Set Q to be \tilde{Q} occasionally according to UCB2 scheduling

end if

end for

end for

Theoretical Result

Theorem 1: Q-Learning with UCB2 scheduling achieves regret $\tilde{O}(\sqrt{H^4 SAT})$ and policy switch bound

$$N_{\text{switch}} \leq O(H^3 SA \log(K/A))$$

which is **logarithmic in K**. 😊

Proof highlight: improved *propagation of error argument* under delayed Q updates.

“Theorem” 2 (lower bound): **Any** sublinear regret or PAC algorithm must have

$$N_{\text{switch}} \geq \Omega(HSA)$$

Mild gap: $O(H^2 \log(K/A))$, conjecture that log is also necessary & gap is at most $O(H^2)$

Discussion & Future Work

- Algorithms with tighter regret bounds (e.g. tighten the H^4)?
- Close the gap between lower and upper bound.
- Model-based algorithms with limited adaptivity? Better bounds?

References

- [1] Jin, C., Allen-Zhu, Z., Bubeck, S. and Jordan, M.I., 2018. Is q-learning provably efficient?. In *Advances in Neural Information Processing Systems* (pp. 4868-4878).
- [2] Auer, P., Cesa-Bianchi, N. and Fischer, P., 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3), pp.235-256.