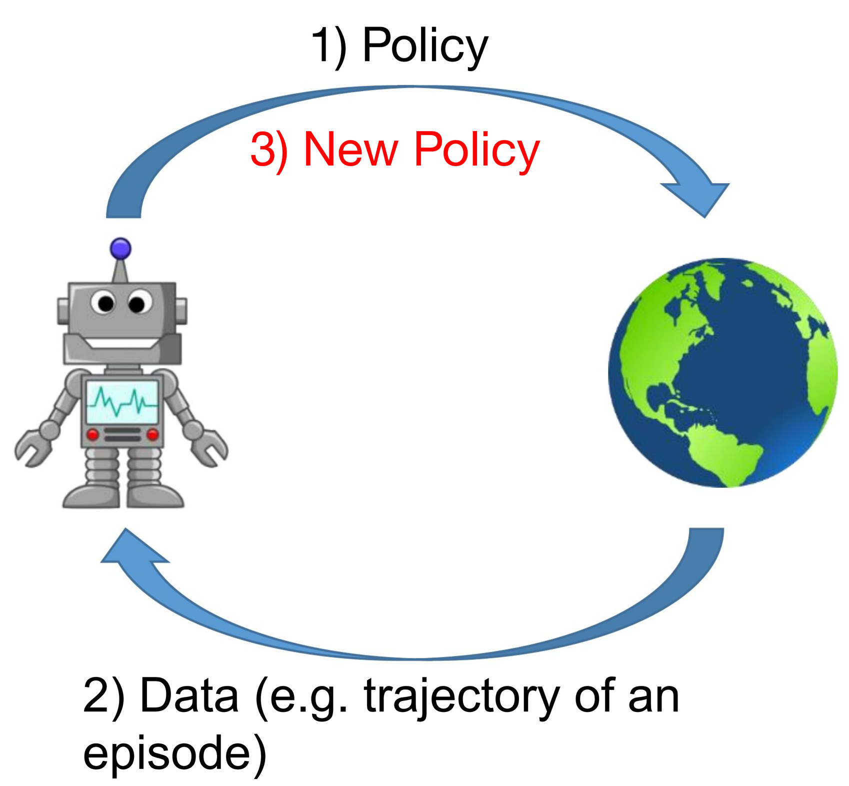
# Provably Efficient Q-Learning with Low Switching Cost

Yu Bai[1,2], Tengyang Xie[3], Nan Jiang[3], Yu-Xiang Wang[4]
[1]Stanford University [2]Salesforce Research [3]UIUC [4]UC Santa Barbara

## Motivation: RL with limited adaptivity?

- In many domains (recommendation, medical, ...), deploying a new policy is more prohibitive than gathering data with the existing policy.




Online RL is **fully adaptive.**

*Any middleground?*

Offline (batch) RL is **non-adaptive**, but much more **challenging**.

## Proposed framework: **low switching cost RL**

**Setup**: Episodic MDP with horizon H. RL algorithm plays K episodes (T= K*H steps.) Measure PAC/Regret.

**Definition**: the ***switching cost*** between two (deterministic) policies $(\pi, \pi')$ is number of different actions they would take, (summed) for all $(h, s)$:

$$n_{\rm switch}(\pi, \pi') := \#\{(h, s) \in [H][S] : \pi_h(s) \neq \pi'_h(s)\}$$

**Definition**: the ***switching cost*** of an RL algorithm that playes with policies $\pi^1, \dots, \pi^K$ is

$$N_{\rm switch} := \sum_{k=1}^{K-1} n_{\rm switch}(\pi^k, \pi^{k+1})$$

## Goal: fast exploration with low switching cost

**Prior work**: Q-Learning with UCB exploration: $\widetilde{O}(\sqrt{H^{4,3}SAT})$ regret, but $N_{\rm switch} = \Theta(HSK)$ linear in K 🤔 [Jin et al. 2018]

*Any low regret algorithm such that $N_{\rm switch}$ sublinear in K?*

## Recap: UCB2 scheduling for bandits

**Algorithm** (UCB2): Repeat until played K times:

- Select the arm 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 [Auer et al. 2002]**: UCB2 achieves same regret as UCB , and only log(K) policy switches: $N_{\rm switch} = O(A \log(K/A))$ 😀

**Idea**: Integrate UCB2 into Q-Learning!

## **Our Algorithm**: Q-Learning with UCB2 scheduling

**Key idea**: update the policy only when Q has been updated $\tau(r) = (1+\alpha)^r$ times.
**Definition**: The *triggering sequence* $\{t_n\}$ with parameter $(\alpha, r_\star)$ is

$$\{t_n\}_{n\geq 1} = \{1, 2, \dots, \tau(r_\star)\} \cup \{\tau(r_\star+1), \tau(r_\star+2), \dots\}$$

**Algorithm 2** Q-learning with UCB2-Hoeffding (UCB2H) Exploration

**input** Parameter $\eta \in (0,1)$, $r_\star \in \mathbb{Z}_{>0}$, and $c > 0$.
  **Initialize:** $\widetilde{Q}_h(x,a) \leftarrow H$, $Q_h \leftarrow \widetilde{Q}_h$, $N_h(x,a) \leftarrow 0$ for all $(x,a,h) \in \mathcal{S} \times \mathcal{A} \times [H]$. // Two sets of Q: Running estimate $\widetilde{Q}$ Policy network $Q$
  **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 $Q$
      $t = N_h(x_h, a_h) \leftarrow N_h(x_h, a_h) + 1$;
      $b_t = c\sqrt{H^3\ell/t}$ (Hoeffding-type bonus);
      $\widetilde{Q}_h(x_h, a_h) \leftarrow (1-\alpha_t)\widetilde{Q}_h(x_h, a_h) + \alpha_t[r_h(x_h, a_h) + \widetilde{V}_{h+1}(x_{h+1}) + b_t]$. // Update $\widetilde{Q}$ via Q-Learning
      $\widetilde{V}_h(x_h) \leftarrow \min\left\{H, \max_{a' \in \mathcal{A}} \widetilde{Q}_h(x_h, a')\right\}$.
      **if** $t \in \{t_n\}_{n\geq 1}$ (where $t_n$ is defined in (1)) **then** ← $\{t_n\}$ is the triggering sequence above
        (Update policy) $Q_h(x_h, \cdot) \leftarrow \widetilde{Q}_h(x_h, \cdot)$. // Set $Q$ to be $\widetilde{Q}$ occasionally according to UCB2 scheduling
      **end if**
    **end for**
  **end for**

## Theoretical Result

**Theorem 1:** Our Q-Learning with UCB2-{Hoeffding, Bernstein} exploration achieves $\widetilde{O}(\sqrt{H^{4,3}SAT})$ regret and **logarithmic switching cost**:

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

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

## Application: concurrent /parallel RL

**Setup**: M agents play an episode in parallel, and can only *communicate after each episode*.

**Idea**: if policy not scheduled to switch in M episodes
⇒ can parallelize to M non-communicating agents

**Theorem 2 (Nearly linear speedup in PAC concurrent RL):** There exists concurrent versions of our algorithm, s.t. given M agents, it can find $\varepsilon$ optimal policy in $\widetilde{O}\left(H^3SA + \frac{H^{\{5,4\}}SA}{\varepsilon^2 M}\right)$ rounds.

→ Also improves upon prior work [Guo et al. 2015] in (H, S, $\varepsilon$) dependence.

## Lower bound on low-switching algos

**Simple Observation**: you "need" to switch HS(A-1) times to at least try out all the possible actions.

**Theorem 3 (Lower bound):** Any algorithm that has switching cost $N_{\rm switch} \leq HSA/2$ has to suffer from linear (trivial) worst-case regret:

$$\sup_{M \in \mathcal{M}} \mathbb{E}_M[{\rm Regret}(K)] \geq KH/4$$

**Remark**: Our algorithm achieves $N_{\rm switch} = \widetilde{O}(H^3SA)$, so still an $H^2$ gap between the lower and upper bounds.

## Discussion & future work

- Close the gap on the switching cost.
- Alternative notions of limited adaptivity:
  - Hard constraint on switching cost.
  - RL with only O(1) rounds of adaptivity.
- Connections to fully offline/batch RL.