Federated Q-Learning: Linear Regret Speedup and Logarithmic Communication Cost

Zhong Zheng

April 20, 2024

Reinforcement Learning and Episodic MDP

Reinforcement Learning and Episodic MDP

- Reinforcement Learning (RL) is a subfield of machine learning focused on sequential decision-making. The primary objective of RL is to obtain an optimal policy through sequential interactions.

Reinforcement Learning and Episodic MDP

- Reinforcement Learning (RL) is a subfield of machine learning focused on sequential decision-making. The primary objective of RL is to obtain an optimal policy through sequential interactions.
- In this work, we focus on a tabular episodic MDP $\mathcal{M} := (\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ with time inhomogeneous transition kernels. Here S, A, H represent the number of states, actions, and steps in an episode, respectively. $\mathbb{P} := \{\mathbb{P}_h\}_{h=1}^{H}$ is the time-inhomogeneous transition kernel. $r := \{r_h\}_{h=1}^{H}$ is the collection of reward functions.

Reinforcement Learning and Episodic MDP

- Reinforcement Learning (RL) is a subfield of machine learning focused on sequential decision-making. The primary objective of RL is to obtain an optimal policy through sequential interactions.
- In this work, we focus on a tabular episodic MDP $\mathcal{M} := (\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$ with time inhomogeneous transition kernels. Here S, A, H represent the number of states, actions, and steps in an episode, respectively. $\mathbb{P} := \{\mathbb{P}_h\}_{h=1}^H$ is the time-inhomogeneous transition kernel. $r := \{r_h\}_{h=1}^H$ is the collection of reward functions.
- A policy π is a collection of H functions $\left\{\pi_h : \mathcal{S} \to \Delta^{\mathcal{A}}\right\}_{h\in[H]}$, where $\Delta^{\mathcal{A}}$ is the set of probability distributions over $\mathcal{A}$.

Value Functions and Optimal Policies

Value Functions and Optimal Policies

- We use $V_h^\pi : \mathcal{S} \to \mathbb{R}$ and $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to denote the state value function and the action value function at step h under policy π.

$$V_h^\pi(x) := \sum_{h'=h}^{H} \mathbb{E}_{(x_{h'}, a_{h'}) \sim (\mathbb{P}, \pi)} \left[r_{h'}(x_{h'}, a_{h'}) \mid x_h = x \right].$$

$$Q_h^\pi(x, a) := r_h(s, a) + \sum_{h'=h+1}^{H} \mathbb{E}_{(x_{h'}, a_{h'}) \sim (\mathbb{P}, \pi)} \left[r_{h'}(x_{h'}, a_{h'}) \mid x_h = x, a_h = a \right].$$

Value Functions and Optimal Policies

- We use $V_h^\pi : \mathcal{S} \to \mathbb{R}$ and $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to denote the state value function and the action value function at step h under policy π.

$$V_h^\pi(x) := \sum_{h'=h}^{H} \mathbb{E}_{(x_{h'}, a_{h'}) \sim (\mathbb{P}, \pi)} \left[r_{h'}(x_{h'}, a_{h'}) \mid x_h = x \right].$$

$$Q_h^\pi(x, a) := r_h(s, a) + \sum_{h'=h+1}^{H} \mathbb{E}_{(x_{h'}, a_{h'}) \sim (\mathbb{P}, \pi)} \left[r_{h'}(x_{h'}, a_{h'}) \mid x_h = x, a_h = a \right].$$

- There always exists an optimal policy $\pi^\star$ for all states and steps. In detail, it achieves the optimal value $V_h^\star(x) = \sup_\pi V_h^\pi(x) = V_h^{\pi^*}(x)$ for all $x \in \mathcal{S}$ and $h \in [H]$. The associated Bellman equations (BE) are as follows.

$$\begin{cases} V_h^\pi(x) = \mathbb{E}_{a \sim \pi_h(x)}[Q_h^\pi(x, a)] \\ Q_h^\pi(x, a) := (r_h + \mathbb{P}_h V_{h+1}^\pi)(x, a) \\ V_{H+1}^\pi(x) = 0, \quad \forall x \in \mathcal{S} \end{cases} \text{ and } \begin{cases} V_h^\star(x) = \max_{a \in \mathcal{A}} Q_h^\star(x, a) \\ Q_h^\star(x, a) := \left(r_h + \mathbb{P}_h V_{h+1}^\star\right)(x, a) \\ V_{H+1}^\star(x) = 0, \quad \forall x \in \mathcal{S}. \end{cases}$$

Here, $[\mathbb{P}_h V_{h+1}](x, a) := \mathbb{E}_{x' \sim \mathbb{P}_h(\cdot|x,a)} V_{h+1}(x')$.

Q-Learning with Upper Confidence Bounds (UCB) (Jin et.al, 2018)

Q-learning is a typical model-free learning algorithm for RL. It maintains estimated Q, V functions and loops the following procedure.

Q-Learning with Upper Confidence Bounds (UCB) (Jin et.al, 2018)

Q-learning is a typical model-free learning algorithm for RL. It maintains estimated Q, V functions and loops the following procedure.

1. Get a policy π via the estimated Q function and BE.

Q-Learning with Upper Confidence Bounds (UCB) (Jin et.al, 2018)

Q-learning is a typical model-free learning algorithm for RL. It maintains estimated Q, V functions and loops the following procedure.

1. Get a policy π via the estimated Q function and BE.
2. Get an episode $(x_1, a_1, r_1, x_2, a_2 \ldots x_H, a_H, r_H)$ under policy π.

Q-Learning with Upper Confidence Bounds (UCB) (Jin et.al, 2018)

Q-learning is a typical model-free learning algorithm for RL. It maintains estimated Q, V functions and loops the following procedure.

1. Get a policy π via the estimated Q function and BE.
2. Get an episode $(x_1, a_1, r_1, x_2, a_2 \ldots x_H, a_H, r_H)$ under policy π.
3. Update the estimated Q functions:

$$Q_h(x_h, a_h) \leftarrow (1-\alpha_h)Q_h(x_h, a_h) + \alpha_h(r_h(x_h, a_h) + V_{h+1}(x_{h+1}) + b_h), h \in [H].$$

Here, $\alpha_h \in (0, 1)$ is a step size, $b_h > 0$ is the UCB that encourages explorations.

Q-Learning with Upper Confidence Bounds (UCB) (Jin et.al, 2018)

Q-learning is a typical model-free learning algorithm for RL. It maintains estimated Q, V functions and loops the following procedure.

1. Get a policy π via the estimated Q function and BE.
2. Get an episode $(x_1, a_1, r_1, x_2, a_2 \ldots x_H, a_H, r_H)$ under policy π.
3. Update the estimated Q functions:

   $$Q_h(x_h, a_h) \leftarrow (1-\alpha_h) Q_h(x_h, a_h) + \alpha_h(r_h(x_h, a_h) + V_{h+1}(x_{h+1}) + b_h), h \in [H].$$

   Here, $\alpha_h \in (0,1)$ is a step size, $b_h > 0$ is the UCB that encourages explorations.
4. Update the estimated V functions via BE.

$$\text{Regret} = \sum_{\text{all episodes } e} V_h^\star(x_{1,e}) - V_h^{\pi_e}(x_{1,e}).$$

Federated Learning and Reinforcement Learning

Federated Learning and Reinforcement Learning

- Federated Learning (FL) is a distributed machine learning framework, where a large number of clients collectively engage in model training and accelerate the learning process, under the coordination of a central server.

Federated Learning and Reinforcement Learning

- Federated Learning (FL) is a distributed machine learning framework, where a large number of clients collectively engage in model training and accelerate the learning process, under the coordination of a central server.
- We wish to extend the FL principle to the RL setting to allow the agents to collaboratively train their decision-making models with limited information exchange.

Federated Learning and Reinforcement Learning

- Federated Learning (FL) is a distributed machine learning framework, where a large number of clients collectively engage in model training and accelerate the learning process, under the coordination of a central server.
- We wish to extend the FL principle to the RL setting to allow the agents to collaboratively train their decision-making models with limited information exchange.
- In this work, we consider a federated RL setting with a central server and M agents, each interacting with $\mathcal{M}$ independently in parallel.

Contributions

We proposed FedQ-Hoeffding and FedQ-Bernstein which are the first model-free federated RL algorithms for the online setting with the following features.

Contributions

We proposed FedQ-Hoeffding and FedQ-Bernstein which are the first model-free federated RL algorithms for the online setting with the following features.

- Linear Regret Speedup. This means that the accuracy of our algorithm matches the situation that all the episodes are generated from a single agent.

Contributions

We proposed FedQ-Hoeffding and FedQ-Bernstein which are the first model-free federated RL algorithms for the online setting with the following features.

- Linear Regret Speedup. This means that the accuracy of our algorithm matches the situation that all the episodes are generated from a single agent.
- Logarithmic communication cost. Comm. cost is defined as the number of scalars communicated during the whole process.

Contributions

We proposed FedQ-Hoeffding and FedQ-Bernstein which are the first model-free federated RL algorithms for the online setting with the following features.

- Linear Regret Speedup. This means that the accuracy of our algorithm matches the situation that all the episodes are generated from a single agent.
- Logarithmic communication cost. Comm. cost is defined as the number of scalars communicated during the whole process.
- Low memory requirement (model-free algorithm).

Contributions

Table 1: Comparison of Related Algorithms

Type	Algorithm (Reference)	Regret	Communication cost
Model-based	Multi-batch RL (Zhang et al. 2022)	$\tilde{O}(\sqrt{H^2SAMT})$	-
	APEVE (Qiao et al. 2022)	$\tilde{O}(\sqrt{H^4S^2AMT})$	-
	Byzan-UCBVI (Chen et al. 2023)	$\tilde{O}(\sqrt{H^3S^2AMT})$	$O(M^2H^2S^2A^2\log T)$
Model-free	Concurrent Q-UCB2H (Bai et al. 2019)	$\tilde{O}(\sqrt{H^4SAMT})$	$O(MT)$
	Concurrent Q-UCB2B (Bai et al. 2019)	$\tilde{O}(\sqrt{H^3SAMT})$	$O(MT)$
	Concurrent UCB-Advantage (Zhang et al. 2020)	$\tilde{O}(\sqrt{H^2SAMT})$	$O(MT)$
	FedQ-Hoeffding (**this work**)	$\tilde{O}(\sqrt{H^4SAMT})$	$O(M^2H^4S^2A\log(T/M))$
	FedQ-Bernstein (**this work**)	$\tilde{O}(\sqrt{H^3SAMT})$	$O(M^2H^4S^2A\log(T/M))$

H: number of steps per episode; T: total number of steps; S: number of states; A: number of actions; M: number of agents.
-: not discussed.

Our Algorithms: Federated Q-Learning Algorithm

Our algorithm proceeds in round $k = 1, 2 \ldots$. In each round,

Our Algorithms: Federated Q-Learning Algorithm

Our algorithm proceeds in round $k = 1, 2 \ldots$. In each round,

- Central Server decides a policy π via the estimated Q, V functions and BE. It shares the estimated V function and the count function N_h for the total visiting number of (x, a, h) to all the agents.

Our Algorithms: Federated Q-Learning Algorithm

Our algorithm proceeds in round $k = 1, 2 \ldots$. In each round,

- Central Server decides a policy π via the estimated Q, V functions and BE. It shares the estimated V function and the count function N_h for the total visiting number of (x, a, h) to all the agents.
- All local agents collect episodes under π. We apply event-triggered abortion conditions to guarantee that $n_h^m(x, a)$, the visiting number of Agent m to (x, a, h), is limited by $N_h(x, a)$.

Our Algorithms: Federated Q-Learning Algorithm

Our algorithm proceeds in round $k = 1, 2 \ldots$. In each round,

- Central Server decides a policy π via the estimated Q, V functions and BE. It shares the estimated V function and the count function N_h for the total visiting number of (x, a, h) to all the agents.
- All local agents collect episodes under π. We apply event-triggered abortion conditions to guarantee that $n_h^m(x, a)$, the visiting number of Agent m to (x, a, h), is limited by $N_h(x, a)$.
- During a non-early round, agents share the reduced information $n_h^m(x, a)$ and $\text{Mean}_{x,a\to x'}^{m,h}(V_{h+1}(x'))$. With them, central server finds out

$$v_{h+1}(x, a) = \text{Mean}_{x,a\to x'}^{h}(V_{h+1}(x')),$$

which is the mean of the estimated value function at step $h + 1$ on all next states for visits of (x, a, h) in this round.

Our Algorithms: Federated Q-Learning Algorithm

Our algorithm proceeds in round $k = 1, 2 \ldots$. In each round,

- Central Server decides a policy π via the estimated Q, V functions and BE. It shares the estimated V function and the count function N_h for the total visiting number of (x, a, h) to all the agents.
- All local agents collect episodes under π. We apply event-triggered abortion conditions to guarantee that $n_h^m(x, a)$, the visiting number of Agent m to (x, a, h), is limited by $N_h(x, a)$.
- During a non-early round, agents share the reduced information $n_h^m(x, a)$ and $\text{Mean}_{x,a \to x'}^{m,h}(V_{h+1}(x'))$. With them, central server finds out

$$v_{h+1}(x, a) = \text{Mean}_{x,a \to x'}^{h}(V_{h+1}(x')),$$

  which is the mean of the estimated value function at step $h + 1$ on all next states for visits of (x, a, h) in this round.
- The central server updates the estimated Q function for all (x, a, h):

$$Q_h(x, a) \leftarrow (1 - \alpha_h) Q_h(x, a) + \alpha_h(r_h(x, a) + v_{h+1}(x, a) + b_h).$$

Our Algorithms: Federated Q-Learning Algorithm

Our algorithm proceeds in round $k = 1, 2 \ldots$. In each round,

- Central Server decides a policy π via the estimated Q, V functions and BE. It shares the estimated V function and the count function N_h for the total visiting number of (x, a, h) to all the agents.
- All local agents collect episodes under π. We apply event-triggered abortion conditions to guarantee that $n_h^m(x, a)$, the visiting number of Agent m to (x, a, h), is limited by $N_h(x, a)$.
- During a non-early round, agents share the reduced information $n_h^m(x, a)$ and $\mathrm{Mean}_{x,a\to x'}^{m,h}(V_{h+1}(x'))$. With them, central server finds out

$$v_{h+1}(x, a) = \mathrm{Mean}_{x,a\to x'}^{h}(V_{h+1}(x')),$$

  which is the mean of the estimated value function at step $h + 1$ on all next states for visits of (x, a, h) in this round.
- The central server updates the estimated Q function for all (x, a, h):

$$Q_h(x, a) \leftarrow (1 - \alpha_h)Q_h(x, a) + \alpha_h(r_h(x, a) + v_{h+1}(x, a) + b_h).$$

- The central server updates the estimated V function according to BE.

Comments

Comments

- We propose FedQ-Hoeffding and FedQ-Bernstein where they use different ways of finding the UCB b_h.

Comments

- We propose FedQ-Hoeffding and FedQ-Bernstein where they use different ways of finding the UCB b_h.
- Our event-triggered synchronization guarantees that for any $(x, a, h, k)-$tuple,

$$n_h^{m,k}(x, a) \leq \max\left\{1, \left\lfloor \frac{N_h^k(x, a)}{MH(H+1)} \right\rfloor \right\},$$

  and for each $k \in [K]$, there exists at least one agent m such that equality is met for a (x, a, h, m)-tuple. This guarantees that a sufficiently large amount of episodes is generated under the same policy and leads to logarithmic communication cost.

Thank you.