Lookahead-Bounded Q-learning

Ibrahim El Shar, Daniel Jiang

University of Pittsburgh

IE Department

Presented at ICML 2020

Vienna, Austria

June 11, 2020

University of Pittsburgh

Introduction

- **Q-learning and its variants** are known to be challenging to apply to real world settings, due to the cost of collecting experience.

- **Information relaxation** is a framework for obtaining upper bounds on MDPs by assuming the future is "known" and solving a related problem.

- In this work, we propose a new algorithm, **lookahead-bounded Q-learning**, that leverages information relaxation to make Q-learning more effective.

University of Pittsburgh

Infinite Horizon Markov Decision Problem

- Consider a γ-discounted infinite horizon problem with finite state and action spaces, $\mathcal{S}$ and $\mathcal{A}$ respectively.

Definitions:

- **Policy:** a mapping $\pi: \mathcal{S} \rightarrow \mathcal{A}$,
- The **state-action value function** of a policy π is:

$$Q^{\pi}(s,a) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t, a_t) \mid s_0 = s, a_0 = a, \pi\right]$$

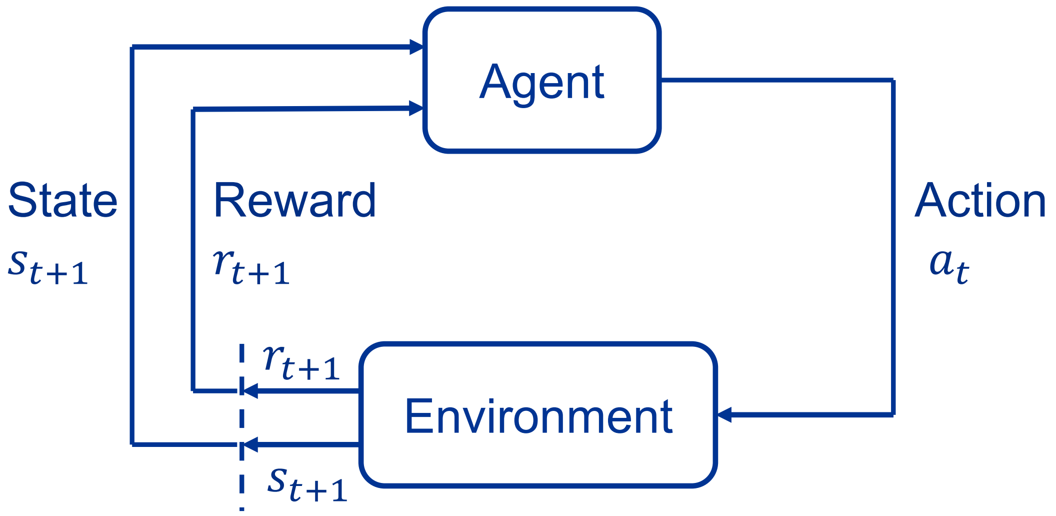
- An **optimal policy** selects actions according to

$$\pi^*(s) = \text{argmax}_{a \in \mathcal{A}}\, Q^*(s,a)$$

where $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$ is the optimal action-value function.

University of Pittsburgh

RL Setting



Popular solution:

Q-learning (Watkins, 1989)

- most widely-used
- conceptual simplicity
- ease of implementation, and convergence guarantees

However…

University of Pittsburgh

Q-learning

Q-learning is hard to use in real-world problems

(expensive simulations/real-world interactions)

$$Q_{n+1}(s,a) = Q_n(s,a) + \alpha_n(s,a)\left[r(s,a) + \gamma \max_a Q(s',a) - Q(s,a)\right]$$

Goal: make better use of collected experience

Approach: use upper and lower bounds derived using information relaxation techniques

But how…

University of Pittsburgh

Q-learning

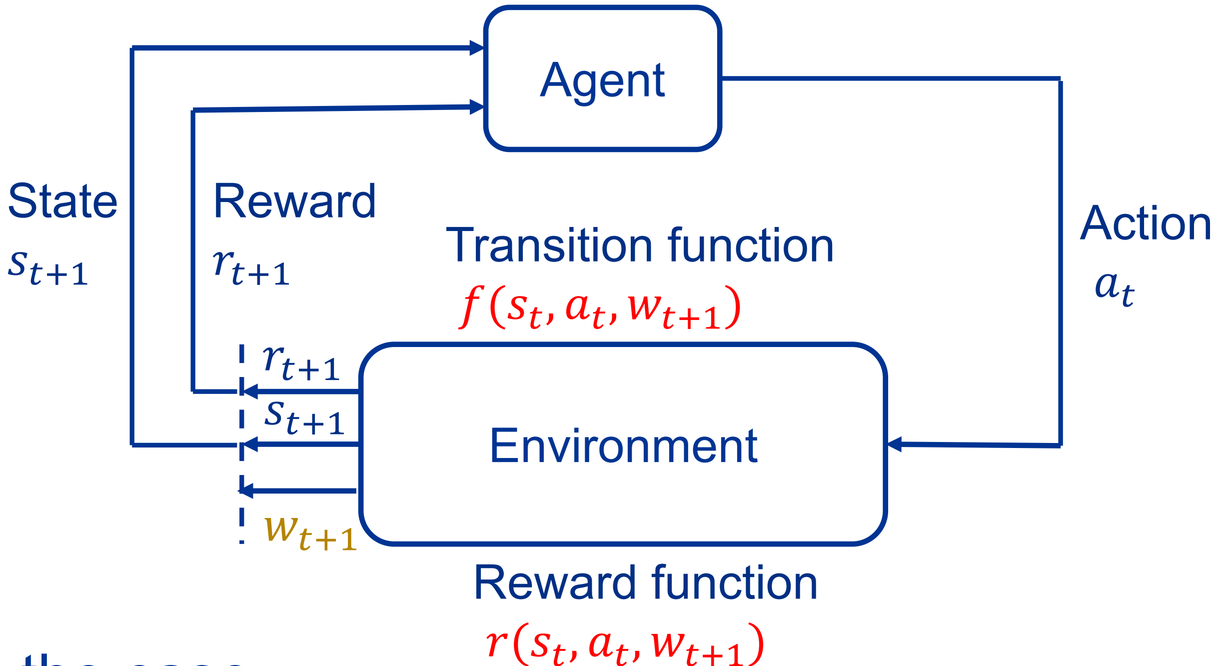
- **Question 1:** If we knew upper and lower bounds on Q*, could we improve this algorithm?
- **Question 2:** Can we dynamically estimate upper and lower bounds that get better over time?

Other Usages of Bounds in RL

- Optimistic RL with Certificates (Dann et al., 2018) and Episodic Upper Lower Exploration in RL (Zanette & Brunskill, 2019)
  - Study finite horizon problems
  - Main goal: achieve better exploration
- Bounded RTDP (McMahan et al., 2005), Bayesian RTDP (Sanner et al., 2009) & Focused RTDP (Smith & Simmons, 2006)
  - Largely use heuristics to obtain bounds
- Faster Deep RL by Optimality Tightening (He et al., 2016)
  - Exploit multistep returns to construct bounds
  - No theoretical guarantees are provided

University of Pittsburgh

Partially Known Dynamics



It is often the case

$$\left.\begin{array}{l}\text{Transition function } f(s_t, a_t, w_{t+1}) \\ \text{Reward function } r(s_t, a_t, w_{t+1})\end{array}\right\}\ \textbf{known form}$$

What is often not known is the random noise w_{t+1}

range of values? distribution?

But it is *observable…*

Partially Known Dynamics

- Typical assumption in the **OR** and **control theory** literature
- Backed by abundant **real-world applications**
- Examples:
  - Inventory problems
  - Car-sharing problems
  - Vehicle routing
  - Renewable energy
  - Maintenance problems
  - Production and scheduling problems
  - Pricing options and trading stocks

$$f(s_t, a_t, d_{t+1}) = s_t + a_t - d_{t+1}$$

- Significant practical interest

University of Pittsburgh

Information Relaxation and Duality Theory

- Proposed by Brown et al. 2010, 2017
- A flexible framework to compute upper bounds on DP values
- **Idea:** relaxing non-anticipativity constraint
- Perfect information relaxation
- Solve a deterministic DP for each sample path $\boldsymbol{W} = \{w_1, w_2, w_3, \ldots\}$:

$$V^*(s) \le \mathbf{E}\left[\max_{\boldsymbol{a}} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\right]$$

Recall: $V^*(s) = \max_{\pi} \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \mid s_0 = s]$

- Absorption time formulation: $\boldsymbol{W} = \{w_1, w_2, , \ldots, w_\tau\}$, $\tau \sim \mathrm{Geom}(1-\gamma)$

$$V^*(s) \le \mathbf{E}\left[\max_{\boldsymbol{a}} \sum_{t=0}^{\tau} r(s_t, a_t) \mid s_0 = s\right] \quad \text{(undiscounted)}$$

University of Pittsburgh

Perfect Information Relaxation is Too Loose

- Brown et al. propose the use of penalties to create a tighter bound.

Perfect information relaxation

Optimal Value

Perfect Information Relaxation is Too Loose

- Brown et al. propose the use of penalties to create a tighter bound.

Perfect information relaxation

Optimal Value
(penalty that uses Q*)

Perfect Information Relaxation is Too Loose

- Brown et al. propose the use of penalties to create a tighter bound.

Perfect information relaxation

Penalized perfect information relaxation
(penalties that use approx. of Q*)

Optimal Value
(penalty that uses Q*)

University of Pittsburgh

Information Relaxation and Duality Theory

- Given φ an approx. of Q* we can construct penalties
- We use penalties of the form

$$\boxed{s_{t+1} = h(s_t, a_t, w_{t+1})}$$

$$\zeta_t^{\pi}(s_t, a_t, w_{t+1} | \varphi) := \gamma^{t+1}\Big(\varphi\big(s_{t+1}, \pi(s_{t+1})\big) - \mathbf{E}\Big[\varphi\Big(h(s_t, a_t, w), \pi\big(h(s_t, a_t, w)\big)\Big)\Big]\Big)$$

- **Weak duality:** for any feasible π and bounded φ:

$$\boxed{\pi_\varphi = \text{greedy}(\varphi)}$$

$$Q^{\pi}(s_0, a_0) \le \mathbf{E}\Bigg[\underbrace{\max_{\boldsymbol{a}} \sum_{t=0}^{\tau-1} r(s_t, a_t) - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} | \varphi)}_{\text{inner problem}}\Bigg]$$

- **Strong duality:**

$$Q^*(s_0, a_0) = \inf_{\boldsymbol{\varphi}} \mathbf{E}\Bigg[\max_{\boldsymbol{a}} \sum_{t=0}^{\tau-1} r(s_t, a_t) - \zeta_t^{\pi_\varphi}(s_t, a_t, w_{t+1} | \varphi)\Bigg]$$

with the infimum attained at $\varphi = Q^*$

University of Pittsburgh

Information Relaxation and Duality Theory

- **Empirical penalty:** given $\{w_{t+1}^1, w_{t+1}^2, \dots, w_{t+1}^K\}$ (black box simulator)

$$\hat{\zeta}_t^{\pi}(s_t, a_t, w_{t+1}|\varphi) := \gamma^{t+1}\left(\varphi(s_{t+1}, \pi(s_{t+1})) - \frac{1}{K}\sum_{k=1}^{K} \varphi\left(h(s_t, a_t, w_{t+1}^k), \pi\left(h(s_t, a_t, w_{t+1}^k)\right)\right)\right)$$

- Given a sample path $\boldsymbol{W} = \{w_1, w_2, , \dots, w_\tau\}$ **:**
- **Noisy Upper Bound:** solve a sampled "inner" DP via the backward recursion

$$\hat{Q}_t^U(s_t, a_t) = r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1}|\varphi) + \max_a \hat{Q}_{t+1}^U(s_{t+1}, a)$$

for $t = \tau - 1, \dots, 0$ with $s_{t+1} = h(s_t, a_t, w_{t+1})$ and $\hat{Q}_\tau^U \equiv 0$

- **Noisy Lower Bound:** inner DP given by π_φ and the recursion

$$\hat{Q}_t^L(s_t, a_t) = r(s_t, a_t) - \hat{\zeta}_t^{\pi_\varphi}(s_t, a_t, w_{t+1}|\varphi) + \hat{Q}_{t+1}^L(s_{t+1}, \pi_\varphi(s_{t+1}))$$

for $t = \tau - 1, \dots, 0$ with $s_{t+1} = h(s_t, a_t, w_{t+1})$ and $\hat{Q}_\tau^U \equiv 0$

University of Pittsburgh

Our Approach…

IR Theory + Q-learning

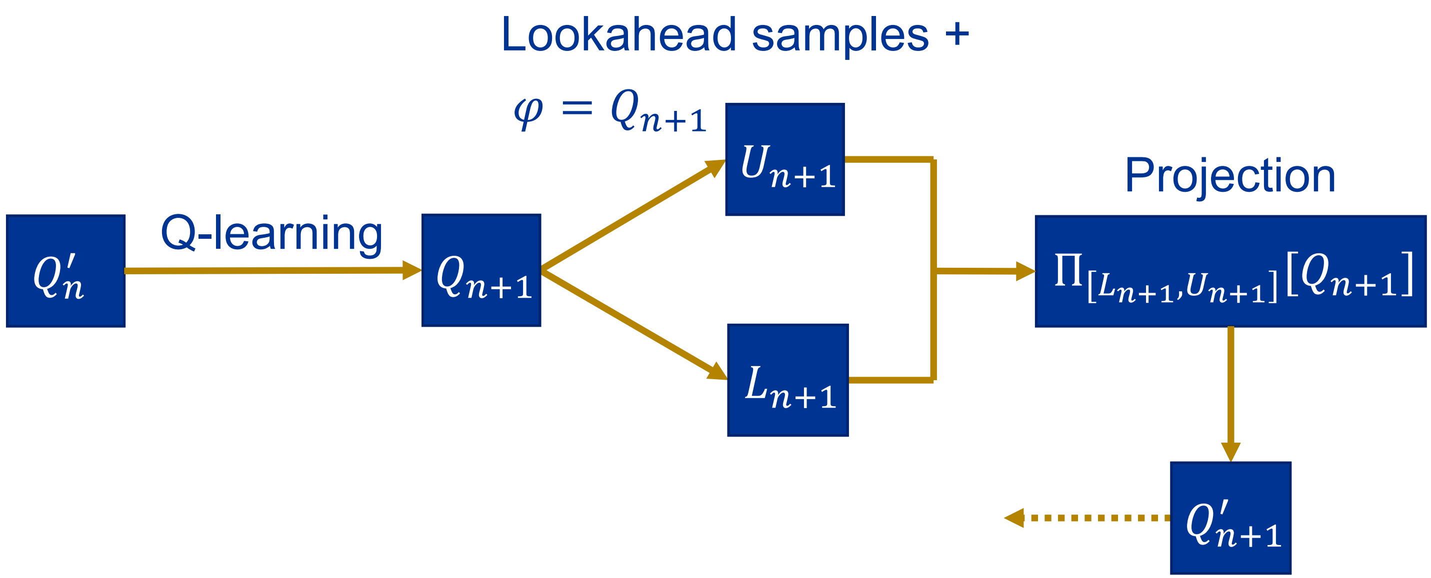
↓

LBQL

Systematic approach to obtain the optimal policy

Q-learning with Lookahead Upper and Lower Bounds

Main idea: Generate improving upper & lower bounds such that the Q-iterates are “squeezed” toward optimality by setting φ to the current Q-iterate.



Q improves ⇒ bounds improve ⇒ Q improves more ⇒ bounds improve more

University of Pittsburgh

Convergence Guarantees

Theorem (Convergence of LBQL). Suppose that:

1. $\sum_{n=0}^{\infty} \alpha_n(s,a) = \infty,\ \sum_{n=0}^{\infty} \alpha_n^2(s,a) < \infty$
2. $\sum_{n=0}^{\infty} \beta_n(s,a) = \infty,\ \sum_{n=0}^{\infty} \beta_n^2(s,a) < \infty$
3. Each state $s \in \mathcal{S}$ is visited infinitely often w.p. 1

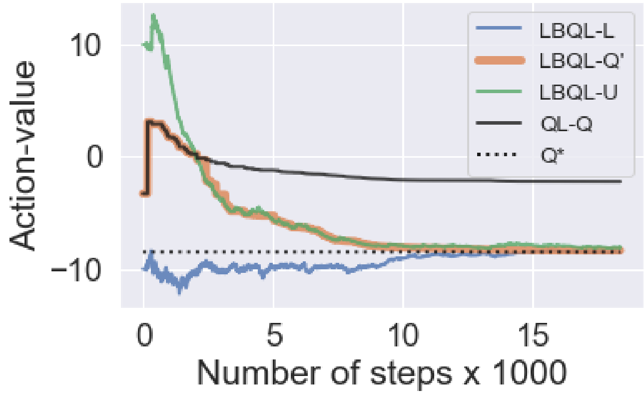
The following hold:

1. With probability one, $Q_n'(s,a)$ converges to the optimal action-value function $Q^*(s,a)$ for all state-action pairs (s,a).
2. If the penalty terms were computed exactly, then w.p. 1, the iterates $L_n(s,a), Q_n'(s,a), U_n(s,a)$ converges to the optimal action-value function $Q^*(s,a)$ for all state-action pairs (s,a).

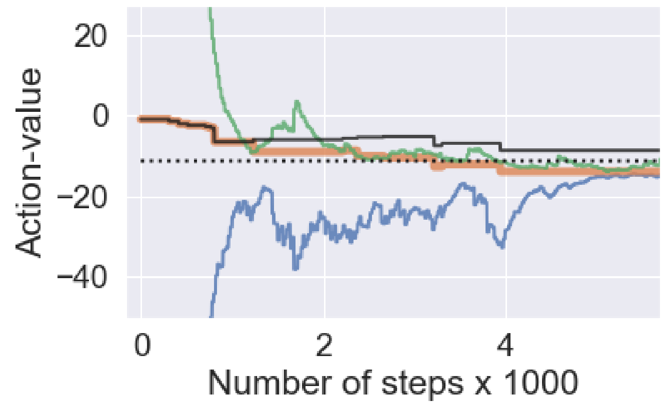
University of Pittsburgh

Behavior of LBQL

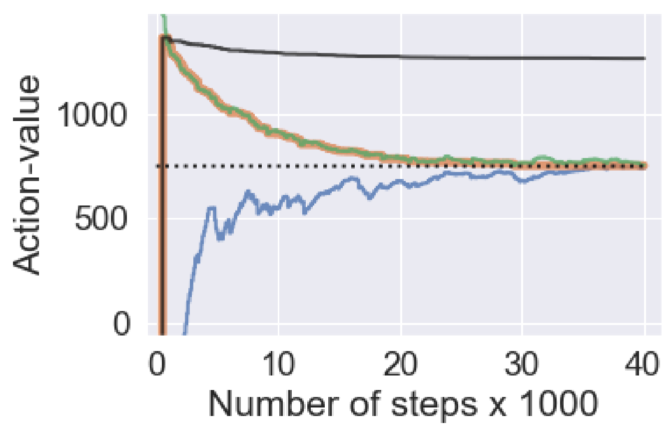
Q improves ⇒ bounds improve ⇒ Q improves more ⇒ bounds improve more



(A) Windy Gridworld



(B) Stormy Gridworld



(C) Pricing for car-sharing (2 stations)

University of Pittsburgh

LBQL with Experience Replay

Main changes:

- A noise buffer $\mathcal{B}$ is used to record observed w. The buffer $\mathcal{B}$ is then used to generate the sample path and the batch samples.

- Similar convergence results are obtained.

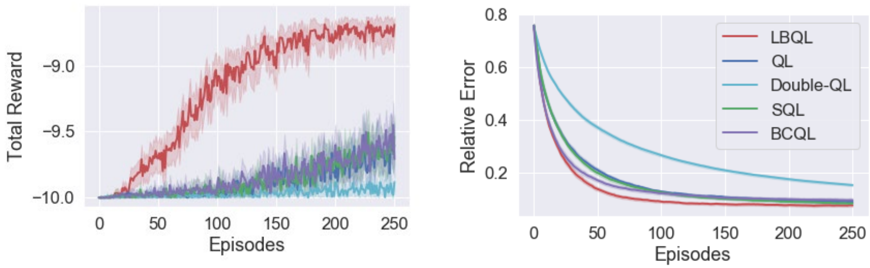
- Need to account for the additional bias due to sampling from the buffer.

Numerical Illustrations

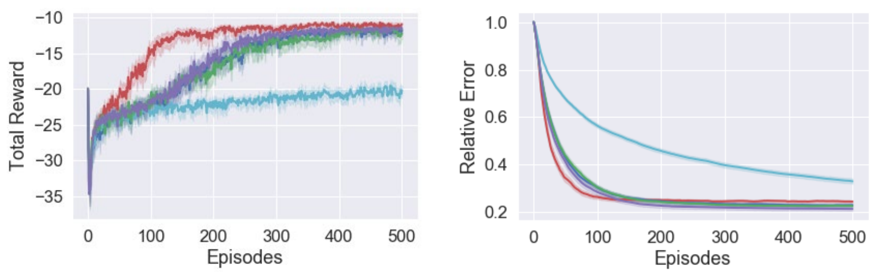
- **Windy Gridworld (Sutton & Barto, 2018)**
  - Gridword + upward wind with random intensity
  - Agent is affected by the wind
- **Stormy Gridworld**
  - Windy gridworld + additional complexity of random rain (negative reward) and multi-directional wind
- **Repositioning in Two Location Car-sharing (He et al., 2019)**
  - Balance cars between stations by direct repositioning
  - One-way rentals, maximize revenue under lost sales cost
- **Spatial Pricing in Two Location Car-sharing (Bimpikis et al., 2019)**
  - Set a price at each station, which influence stochastic demand for rentals
- **Spatial Pricing in Four Location Car-sharing**
  - Four stations
  - One-way + return rentals
  - Two sources of randomness (demand & rentals distribution)

Comparison to Other Algorithms

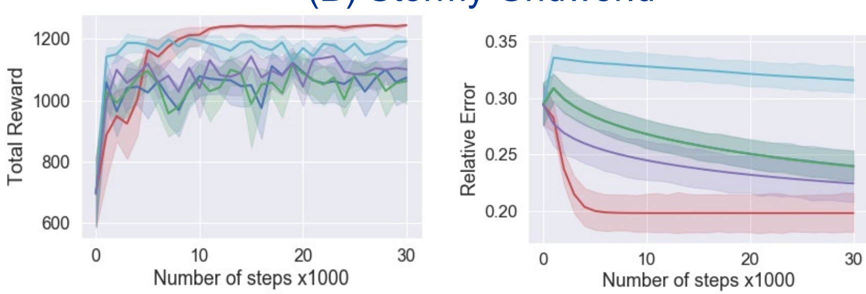
Relative Error = $\|V - V^*\|_2 \, / \|V^*\|_2$



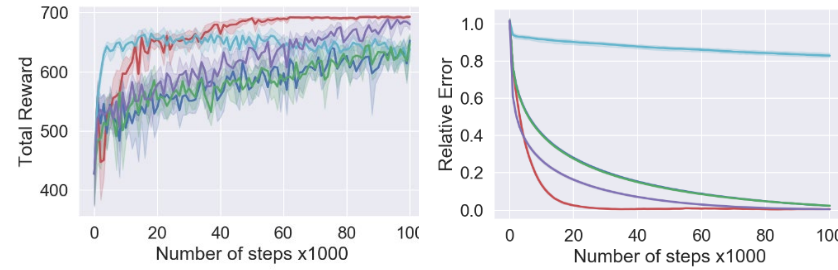
(A) Windy Gridworld



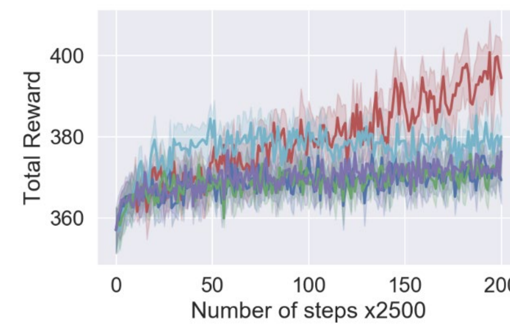
(B) Stormy Gridworld



(C) Repositioning for car-sharing (2 stations)



(D) Pricing for car-sharing (2 stations)



(E) Pricing for car-sharing (4 stations)

University of Pittsburgh

LBQL is Robust to Hyperparameters

Polynomial learning rate: $\alpha_n(s,a) = 1/\nu_n(s,a)^r$

ϵ-greedy exploration strategy: $\epsilon(s) = 1/\nu(s)^e$

$\nu(s,a)$ and $\nu(s)$ are the number of times (s,a) and s, have been visited, respectively

n: average number of iterations

t (s): CPU time

'-' indicates that the % RE was not achieved throughout training

			% Relative error							
	e	*r*	20%		10%		5%		1%	
			n	t (s)	*n*	t (s)	*n*	t (s)	*n*	t (s)
LBQL	0.4	0.5	**9,323.6**	**4.6**	**13,439.0**	**6.5**	**18,456.6**	**8.9**	**33,054.0**	**15.7**
		0.8	**9,321.8**	**4.4**	**14,253.8**	**6.8**	**20,860.0**	**9.9**	**53,752.6**	**25.6**
	0.6	0.5	**7,129.0**	**3.3**	**9,871.8**	**4.6**	**13,046.0**	**6.2**	**23,822.0**	**11.2**
		0.8	**8,431.0**	**4.1**	**11,551.2**	**5.6**	**17,809.2**	**8.5**	**114,032.8**	**54.2**
QL	0.4	0.5	38,114.2	8.2	66,647.2	14.4	93,303.4	20.2	136,851.4	29.6
		0.8	-	-	-	-	-	-	-	-
	0.6	0.5	24,877.2	5.4	44,818.4	9.7	63,777.8	13.7	96,402.0	20.8
		0.8	-	-	-	-	-	-	-	-
SQL	0.4	0.5	37,889.8	9.1	66,583.8	16.0	93,820.0	22.5	141,171.0	33.8
		0.8	-	-	-	-	-	-	-	-
	0.6	0.5	24,989.0	6.0	45,120.2	10.8	64,605.4	15.4	98,554.6	23.6
		0.8	-	-	-	-	-	-	-	-
Double QL	0.4	0.5	-	-	-	-	-	-	-	-
		0.8	-	-	-	-	-	-	-	-
	0.6	0.5	-	-	-	-	-	-	-	-
		0.8	-	-	-	-	-	-	-	-
BCQL	0.4	0.5	22,455.0	7.0	43,866.8	13.6	65,329.0	20.3	107,785.6	33.7
		0.8	-	-	-	-	-	-	-	-
	0.6	0.5	11,639.6	3.6	23,267.0	7.2	35,763.0	11.1	61,249.0	19.0
		0.8	297,368.6	17.9	-	-	-	-	-	-

Conclusions

- LBQL makes more efficient use of the collected experience by additionally using it to estimate bounds.
- LBQL converges almost surely to the optimal action-value function.
- Experiments show that LBQL outperforms other related algorithms and is robust to learning rates and exploration strategies.