DS 598
Introduction to RL

Xuezhou Zhang

Chapter 4: Value-based RL (Continued)

Solve for Q^* from data

Given a dataset $\mathrm{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^n$.

Fitted Q iteration (FQI)

1. Initialize $Q^{(0)}$ arbitrarily.
2. For $t = 1, \ldots T$
   - $Q^{(i)}(s,a) = \mathrm{argmin}_{f \in \mathcal{F}} \sum_{i=1}^n \left(f(s_i, a_i) - r_i - \gamma \max_{a'} Q^{(i-1)}(s_i', a') \right)^2$
3. Return $Q^{(T)}$.

Bellman Error

FQI solves for the equation ∂BE=0.

Summary

- FQI requires storing all historical data, which is memory inefficient.

Q-Learning: a streaming algorithm

- At time step t,
- Observes transition tuple (s_t, a_t, r_t, s_t')
- Q-learning:
- $Q^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \max_{a'} Q^{(t)}(s_t', a') - Q^{(t)}(s_t, a_t)\right)$

- Q-learning is taking **one gradient step** w.r.t. the **FQI objective** with step size $\alpha_t(s_t, a_t)$.

Summary

- FQI requires storing all historical data, which is memory inefficient.
- Q-learning converges just fine in the tabular setting.

What happens beyond the tabular setting?

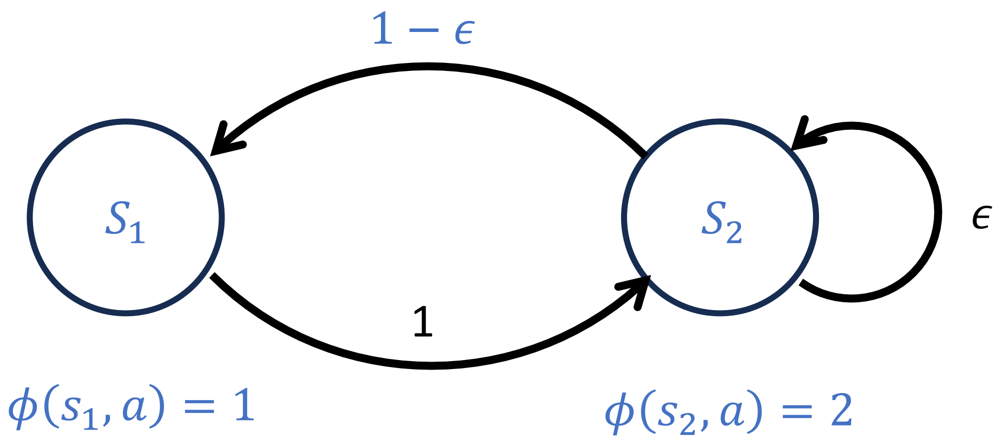
- Value-based RL may fail:

1. They might not converge (algorithm-specific).

2. They might not converge to the correct solution (all value-based RL).

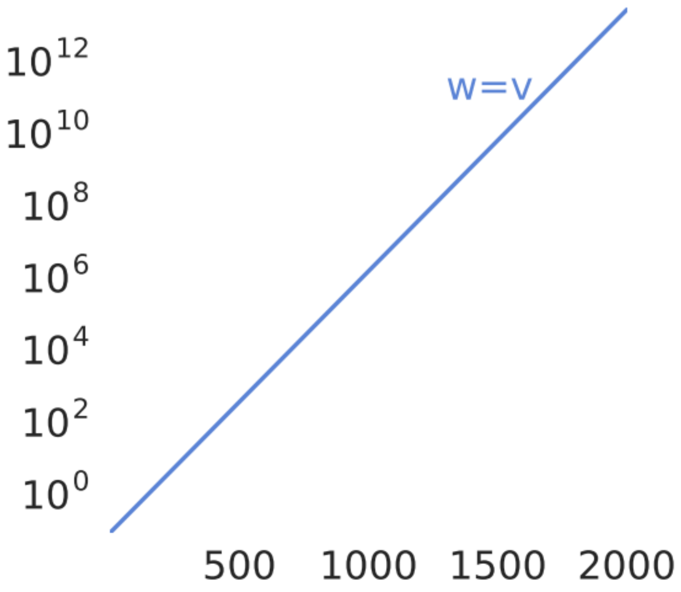
A Failure Example

- MDP: 2 states, 1 action.

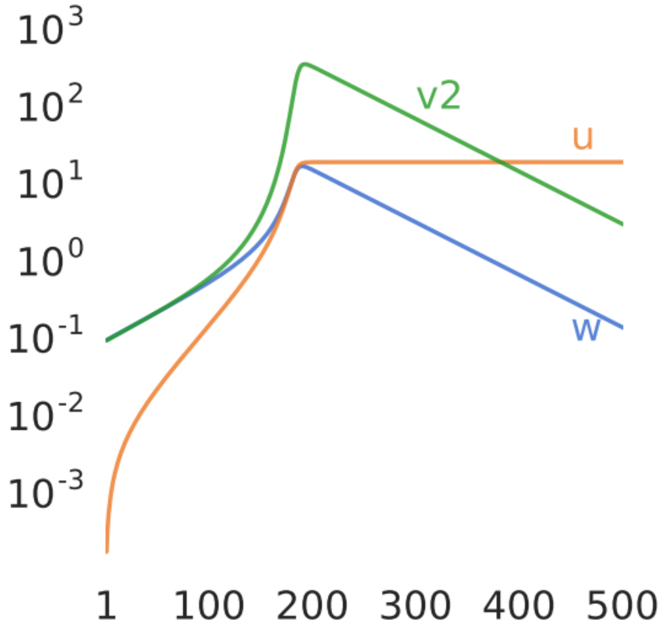


- **Realizable** Linear Function Approximation: $Q(s,a) = \phi(s,a)^\top w$.

A Failure Example



(b) $v(s) = w\phi(s)$ diverges.



(c) $v(s) = w(\phi(s) + u)$ converges.

The sad story of Bellman-Completeness

- A Q function class $\mathcal{F}$ is **Bellman-Complete** if
- For any $f \in \mathcal{F}$, there exists $g \in \mathcal{F}$, such that

$$g(s,a) = (\mathcal{T}f)(s,a) = r(s,a) + \mathbb{E}_{s' \sim P(s'|s,a)}\left[\max_{a'} f(s',a')\right]$$

- In other words, $\mathcal{F}$ is **closed** under Bellman operator $\mathcal{T}$.
- Completeness is **not monotone**, so having a rich function class won't help.

The sad story of Bellman-Completeness

(a) (b)

Bellman-complete

not Bellman-complete

The sad story of Bellman-Completeness

- Theorem (Foster et al., 2022). Value-based method can fail without Bellman-Completeness.
- They contrasted an failure example, where any algorithm require at least $|S|^{1/3}$ samples to learn a good policy.
- This is an algorithm-independent result.

Foster et al. Offline Reinforcement Learning: Fundamental Barriers for Value Function Approximation. 2022.

Summary

- FQI requires storing all historical data, which is memory inefficient.

- Q-learning converges just fine in the tabular setting.

- When using function approximation,
  1) Value-based RL can converge to the wrong solution (FQI, Q-learning)
  2) Value-based RL may not even converge (Q-learning)

Can we (at least) make Q-learning converge?

- Trick #1: Target network (two time-scale update rule)

$$Q^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \max_{a'} Q^{(t)}(s_t', a') - Q^{(t)}(s_t, a_t)\right)$$

Can we (at least) make Q-learning converge?

- Trick #1: Target network (two time-scale update rule)

$$Q^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \max_{a'} T^{(t)}(s_t', a') - Q^{(t)}(s_t, a_t)\right)$$

$$T^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \beta_t(s_t, a_t)\left(Q^{(t)}(s_t, a_t) - T^{(t)}(s_t, a_t)\right)$$

(A slowly updating target network)

Can we (at least) make Q-learning converge?

- Trick #2: Double Q-learning

$$Q^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma\, T^{(t)}(s_t', a') - Q^{(t)}(s_t, a_t)\right)$$

$$T^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \beta_t(s_t, a_t)\left(Q^{(t)}(s_t, a_t) - T^{(t)}(s_t, a_t)\right)$$

$$a' = \mathrm{argmax}_a\, Q^{(t)}(s_t', a)$$

Can we (at least) make Q-learning converge?

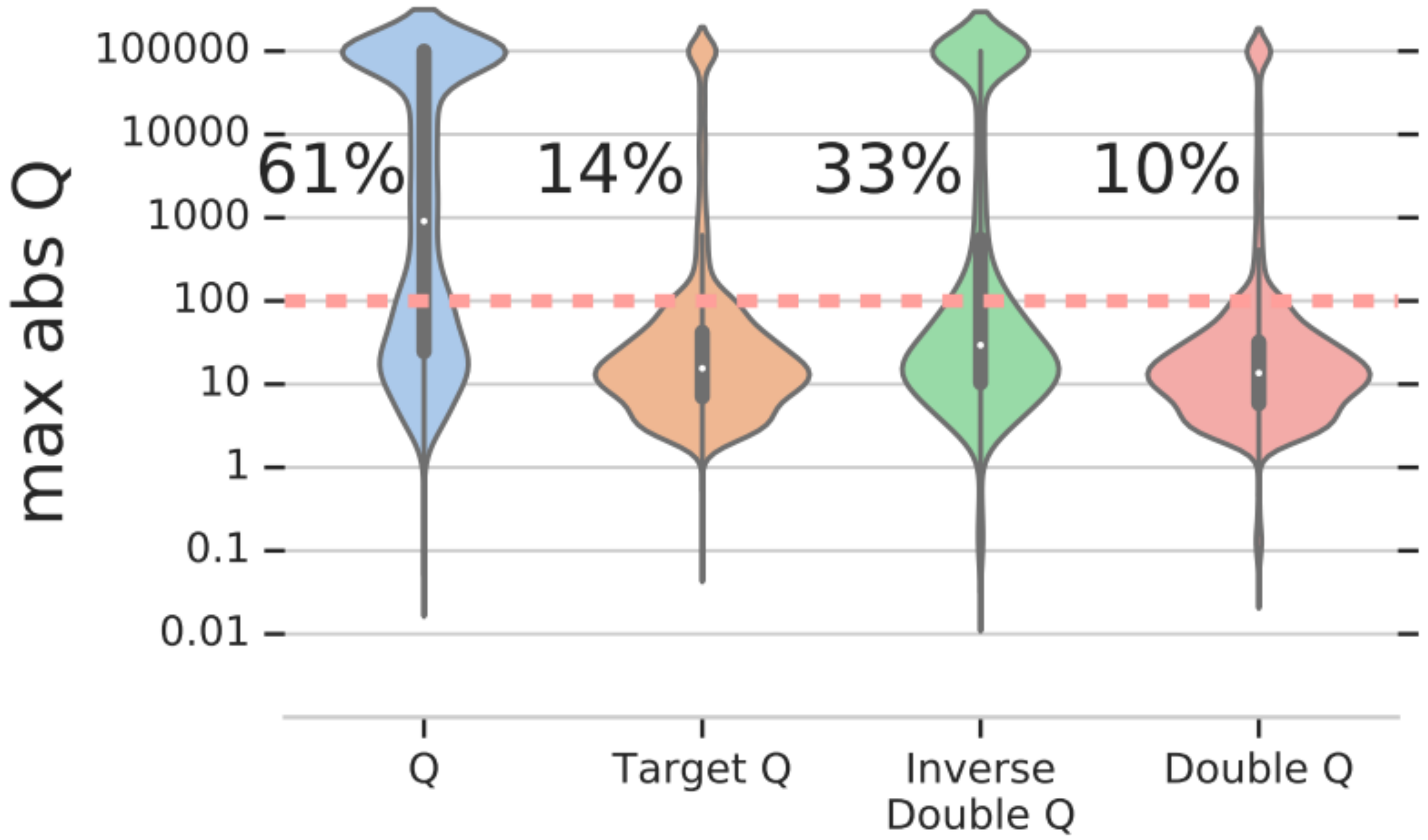
- Baseline 3: Inverse double-Q learning

$$Q^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma\, Q^{(t)}(s_t', a') - Q^{(t)}(s_t, a_t)\right)$$

$$T^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \beta_t(s_t, a_t)\left(Q^{(t)}(s_t, a_t) - T^{(t)}(s_t, a_t)\right)$$

$$a' = \mathrm{argmax}_a\, T^{(t)}(s_t', a)$$

Do they work?


100000
10000
1000
100
10
1
0.1
0.01
max abs Q
61%
14%
33%
10%
Q
Target Q
Inverse Double Q
Double Q

Combine FQI and Q-learning

- Maintain a reasonably sized memory of historical data.

- a.k.a “replay buffer”

DQN (Mnih et al. 2013)

"classic" deep Q-learning algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps

Double-DQN: $a'_j = \mathbf{argmax}_a Q_\phi(s'_j, a)$

Multi-step Return (bias/variance trade-off)

- Vanilla Q-learning (single-step return):

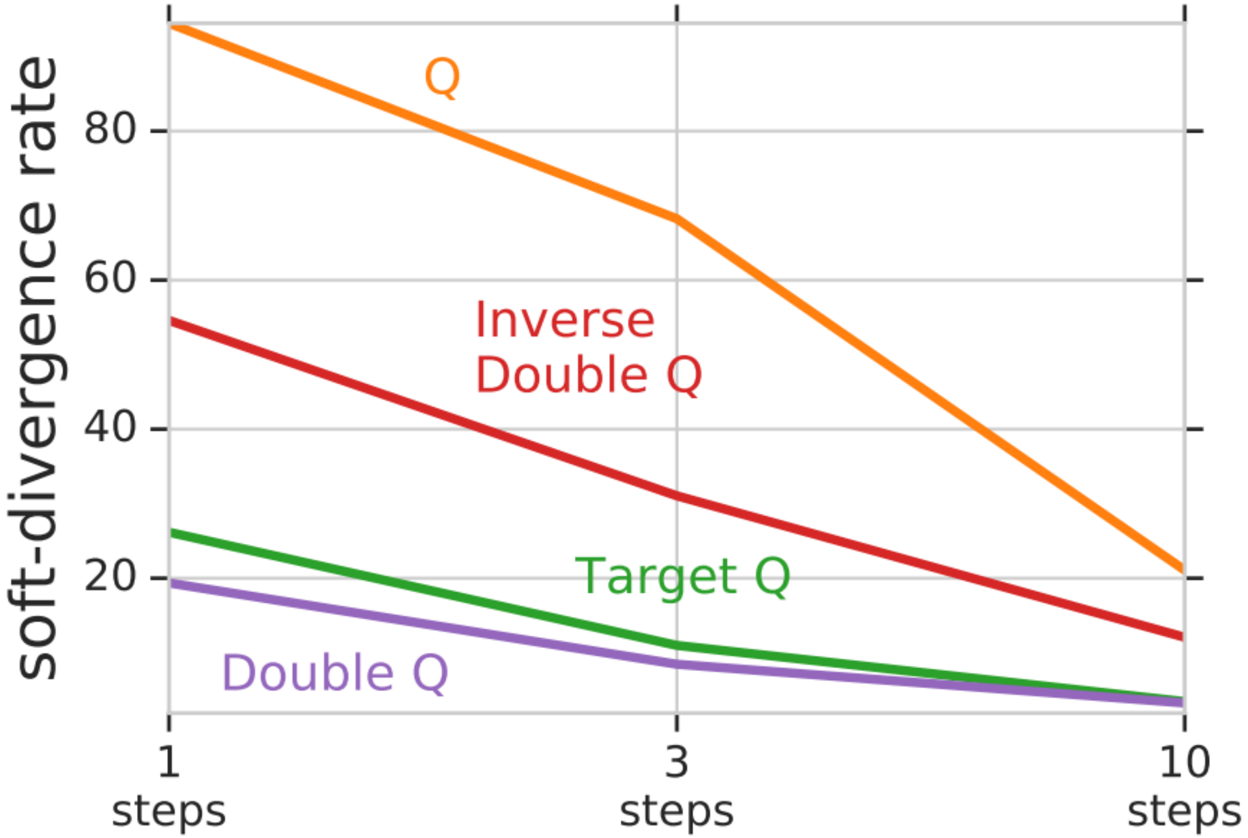
$$Q^{(t+1)}(s_t, a_t) = Q^{(t)}(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_{a'} Q^{(t)}(s_t', a') - Q^{(t)}(s_t, a_t) \right)$$

- Q-learning with multi-step return:

$$Q^{(t+1)}(s_\mathrm{h}, a_h) = Q^{(t)}(s_h, a_h) + \alpha_t \left(r_h + \gamma r_{h+1} \dots + \gamma^\tau r_{h+\tau} + \gamma^{\tau+1} \max_{a'} Q^{(t)}(s_{h+\tau+1}', a') - Q^{(t)}(s_t, a_t) \right)$$

$$\xrightarrow{\tau \to \infty} V^{\pi^{(t)}}(s_h)$$

Multi-step Return



(a) Multi-step returns