535514: Reinforcement Learning
Lecture 20 — Q-Learning

Ping-Chun Hsieh

May 2, 2024

On-Policy vs Off-Policy Methods

<table>
<tr><th></th><th>Policy Optimization</th><th>Value-Based</th><th>Model-Based</th><th>Imitation-Based</th></tr>
<tr><td>On-Policy</td><td>Exact PG
REINFORCE (w/i baseline)
A2C
On-policy DAC
TRPO
Natural PG (NPG)
PPO-KL & PPO-Clip
RLHF by PPO-KL</td><td>Epsilon-Greedy MC
Sarsa
Expected Sarsa</td><td rowspan="2">Model-Predictive Control (MPC)
PETS</td><td rowspan="2">IRL
GAIL
IQ-Learn</td></tr>
<tr><td>Off-Policy</td><td>Off-policy DPG & DDPG
Twin Delayed DDPG (TD3)</td><td>Q-learning
Double Q-learning
DQN & DDQN
C51 / QR-DQN / IQN
Soft Actor-Critic (SAC)</td></tr>
</table>

Quick Review

- Sarsa?

- Expected Sarsa?

Review: From Expected Sarsa to Q-Learning

Expected Sarsa when π is deterministic:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s', \pi(s')) - Q(s,a)\Big)$$

- Idea: Let's allow both behavior and target policies to improve

Target policy π: Greedy w.r.t. $Q(s,a)$

Behavior policy β: ε-Greedy w.r.t. $Q(s,a)$

Q-Learning under the above π, β:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s', a') - Q(s,a)\Big)$$

Q-learning is an "off-policy" version of Expected Sarsa!

Q-Learning Algorithm (With ε-Greedy Exploration)

- Q-Learning:

Step 1: Initialize $Q(s, a)$ for all (s, a), and initial state s_0

Step 2: For each step $t = 0,1,2,\cdots$

Select a_t using ε-greedy w.r.t $Q(s_t, \cdot\ )$

Observe (r_{t+1}, s_{t+1})

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right)$$

- Question: Is “Q-learning” *on-policy* or *off-policy*?

Convergence of Q-Learning

- **Theorem**: Q-learning converges to the optimal action-value function, i.e., $Q(s, a) \to Q_*(s, a)$, under the following conditions:

  (1) GLIE (2) $\sum_{t=1}^{\infty} \alpha_t(s, a) = \infty, \ \sum_{t=1}^{\infty} \alpha_t(s, a)^2 < \infty$, for all (s, a)

- Proof: Stochastic approximation (similar to the proof for Sarsa)

Watkins and Dayan,"Q-Learning," Machine Learning, 1992

How to Interpret Q-Learning as SA?

Recall: A Popular Variant of SA Theorem

$$\Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$$

- **Stochastic Approximation (Jaakkola, Jordan, Singh, 1993)**:

  If the following conditions are satisfied for all $x \in \mathscr{X}$:

(SA1) $0 \le \alpha_n(x) \le 1, \sum_{n=1}^{\infty} \alpha_n(x) = \infty, \sum_{n=1}^{\infty} \alpha_n(x)^2 < \infty$, w.p.1

(SA2) $\left\| \mathbb{E}\left[\varepsilon_n \,|\, \mathscr{H}_n\right] \right\|_\infty \le \rho \|\Delta_n\|_\infty + c_n, \quad \rho \in [0,1)$ and $c_n \to 0$ w.p.1

(SA3) $\mathbb{V}\left[\varepsilon_n(x) \,|\, \mathscr{H}_n\right] \le C(1 + \|\Delta_n\|_\infty)^2$, where C is some constant

then $\Delta_n \to 0$, w.p.1

Jaakkola et al., "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms", 1993

How to Interpret Q-Learning as SA?

SA Update $\Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$

Q-Learning $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_t + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t))$

$Q_{t+1}(s, a) = Q_t(s, a),$ for other $(s, a) \neq (s_t, a_t)$

$x \Leftrightarrow$

$\Delta_n(x) \Leftrightarrow$

$\alpha_n(x) \Leftrightarrow$

$\varepsilon_n(x) \Leftrightarrow$

$\mathscr{H}_n \Leftrightarrow$

Step 1: Interpret Q-Learning as SA (Formally)

SA Update $\quad \Delta_{n+1}(x) = (1 - \alpha_n(x)) \cdot \Delta_n(x) + \alpha_n(x) \cdot \varepsilon_n(x)$

Q-Learning $\quad Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_t + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t))$

$Q_{t+1}(s, a) = Q_t(s, a),$ for other $(s, a) \neq (s_t, a_t)$

$x \Leftrightarrow (s, a)$ pairs

$\Delta_n(x) \Leftrightarrow Q_t(s, a) - Q^*(s, a)$

$\alpha_n(x) \Leftrightarrow$ For (s_t, a_t): $\alpha_t(s_t, a_t)$

But for other $(s, a) \neq (s_t, a_t)$: 0

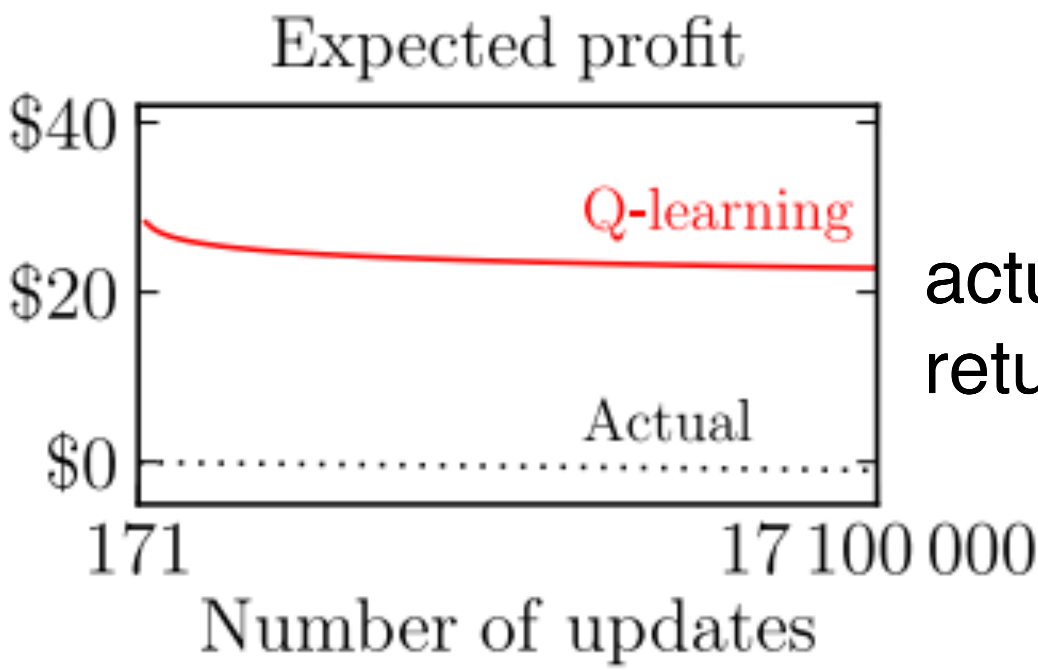
$\varepsilon_n(x) \Leftrightarrow$ For (s_t, a_t): $r_t + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q^*(s_t, a_t) =: \varepsilon_t(s_t, a_t)$

But for other $(s, a) \neq (s_t, a_t)$: 0

$\mathscr{H}_n \Leftrightarrow \{s_0, a_0, r_1, \cdots, s_t, a_t\}$

An Issue With Q-Learning: Overestimation Bias

- Example: Roulette with 1 state and 171 actions (assume \$1 for each bet)



Average action values for Q-learning on roulette with synchronous updates, $\alpha = 1/n$ and $\gamma = 0.95$.

actual expected return = -\$0.053

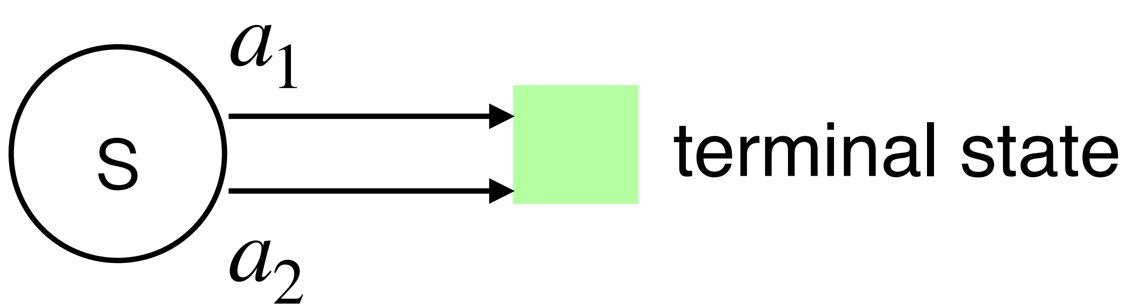
Q-learning after 10^5 trials: Each dollar yields ≈ \$22

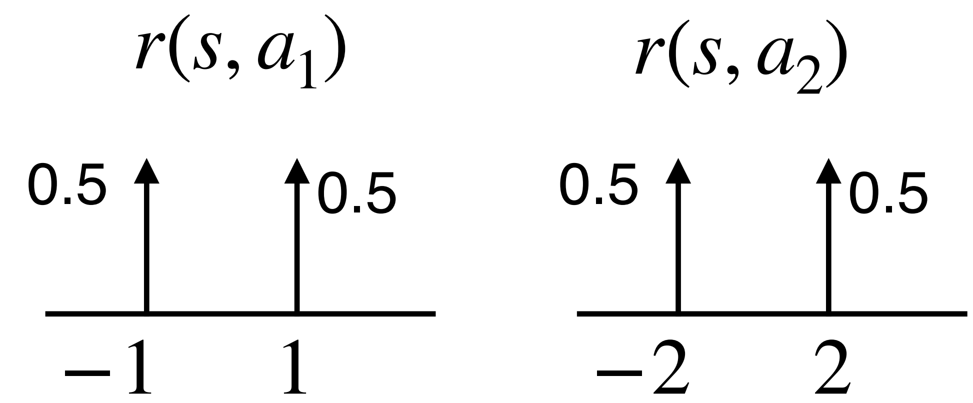
Q-learning can suffer from overestimation (with finite samples)!

Hado van Hasselt et al., Double Q Learning, NIPS 2010

Overestimation Bias: A Motivating Example

- Example: 1-state MDP with 2 actions





- Let $\hat{Q}(s, a_1)$, $\hat{Q}(s, a_2)$ be unbiased estimators (based on 1 reward sample)

- Overestimation:

$$\mathbb{E}\left[\max\{\hat{Q}(s, a_1), \hat{Q}(s, a_2)\}\right] > \max\left\{\mathbb{E}[\hat{Q}(s, a_1)], \mathbb{E}[\hat{Q}(s, a_2)]\right\} = \max_a Q(s, a)$$

$$\max\left\{\mathbb{E}[\hat{Q}(s, a_1)], \mathbb{E}[\hat{Q}(s, a_2)]\right\} =$$

$$\mathbb{E}\left[\max\{\hat{Q}(s, a_1), \hat{Q}(s, a_2)\}\right] =$$

Double Q-Learning

Hado van Hasselt et al., Double Q Learning, NIPS 2010

How to Mitigate Overestimation Bias: Double Estimators

- Observation: Overestimation bias can occur under a greedy policy w.r.t the estimated Q function

- Idea: Avoid using “max of estimates” as “estimate of max”

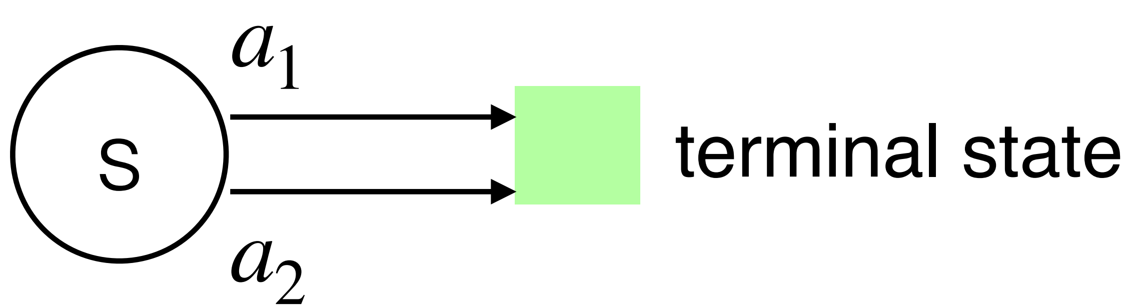
- Create 2 independent unbiased estimates $\hat{Q}_1(s, a)$,$\hat{Q}_2(s, a)$
  - Use one estimate to select action: $a^* = \arg\max_a \hat{Q}_1(s, a)$
  - Use the other estimate for evaluate a^*: $\hat{Q}_2(s, a^*)$
  - Obtain an unbiased estimate: $\mathbb{E}[\hat{Q}_2(s, a^*)] = Q(s, a^*)$

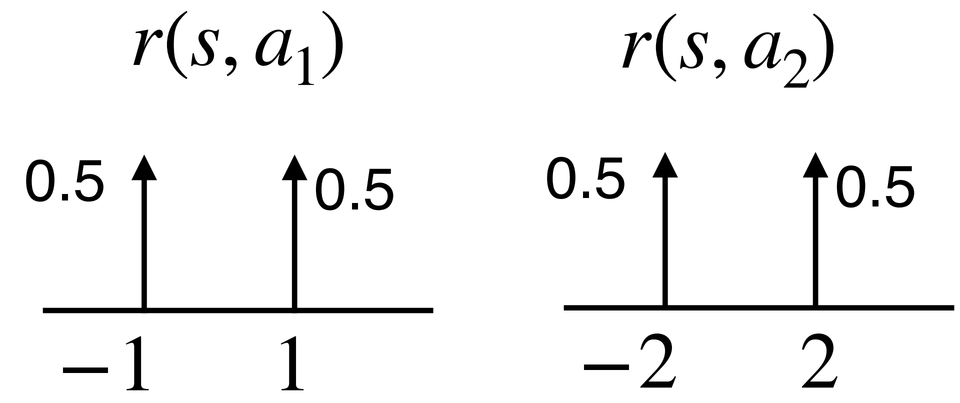
(Unfortunately, unbiased only for $Q(s, a^*)$, not for $\max_a Q(s, a)$)

Hado van Hasselt et al., Double Q Learning, NIPS 2010

Estimation Bias of Double Estimators

- Example: 1-state MDP with 2 actions





- Create 2 independent unbiased estimates $\hat{Q}_A(s,a), \hat{Q}_B(s,a)$
  - Use one estimate to select action: $\bar{a} = \arg\max_a \hat{Q}_A(s,a)$
  - Use the other estimate for evaluate $\bar{a}$: $\hat{Q}_B(s,\bar{a})$
  - Obtain an unbiased estimate: $\mathbb{E}[\hat{Q}_B(s,\bar{a})] = Q(s,\bar{a})$

$$\max_a Q(s,a) = \max\left\{\mathbb{E}[\hat{Q}_{A/B}(s,a_1)], \mathbb{E}[\hat{Q}_{A/B}(s,a_2)]\right\} = 0$$

$$\mathbb{E}[\hat{Q}_B(s,\bar{a})] =$$

How to extend such ideas to general MDPs?

Double Q-Learning Algorithm (Formally)

- Double Q-Learning:

Step 1: Initialize $Q^A(s,a), Q^B(s,a)$ for all (s,a), and initial state s_0

Step 2: For each step $t = 0,1,2,\cdots$

Select a_t using ε-greedy w.r.t $Q^A(s_t,a) + Q^B(s_t,a)$

Observe (r_{t+1}, s_{t+1})

Choose one of the following updates uniformly at random

$$Q^A(s_t,a_t) \leftarrow Q^A(s_t,a_t) + \alpha\left(r_{t+1} + \gamma Q^B(s_{t+1}, \arg\max_a Q^A(s_{t+1},a)) - Q^A(s_t,a_t)\right)$$

$$Q^B(s_t,a_t) \leftarrow Q^B(s_t,a_t) + \alpha\left(r_{t+1} + \gamma Q^A(s_{t+1}, \arg\max_a Q^B(s_{t+1},a)) - Q^B(s_t,a_t)\right)$$

- Question: *Memory & computation per step (compared to Q-learning)*?

van Hasselt et al., Double Q Learning, NIPS 2010

Convergence of Double Q-Learning

- Technical assumptions:

(A1) Both Q^A, Q^B receive infinite number of updates

(A2) Q^A, Q^B are stored in a lookup table

(A3) Each state-action pair is visited an infinite number of times

(A4) Step sizes: $\sum_k \alpha_k = \infty, \sum_k \alpha_k^2 < \infty$ (Robbins-Monro conditions)

- **Convergence Result**:

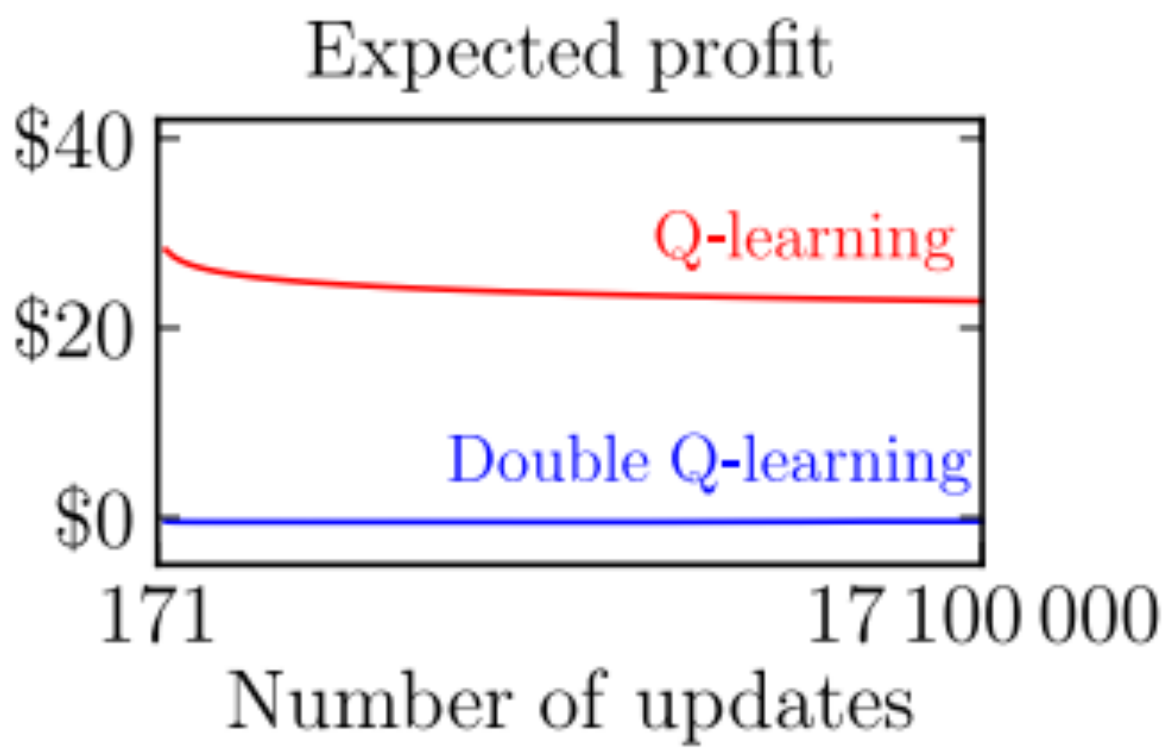
  Under the assumptions (A1)-(A4), both Q^A and Q^B converge to the optimal Q function, almost surely.

- Proof: Stochastic approximation (similar to Q-learning)

van Hasselt et al., Double Q Learning, NIPS 2010

Evaluation: Q-Learning and Double Q-Learning

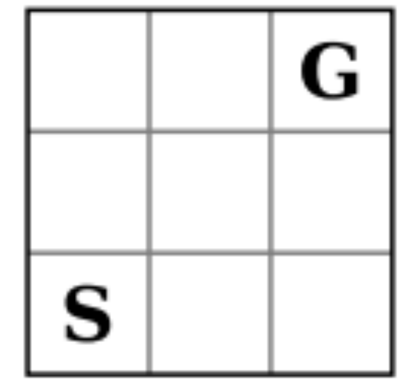
- Example: Roulette with 1 state and 171 actions (assume $1 for each bet)

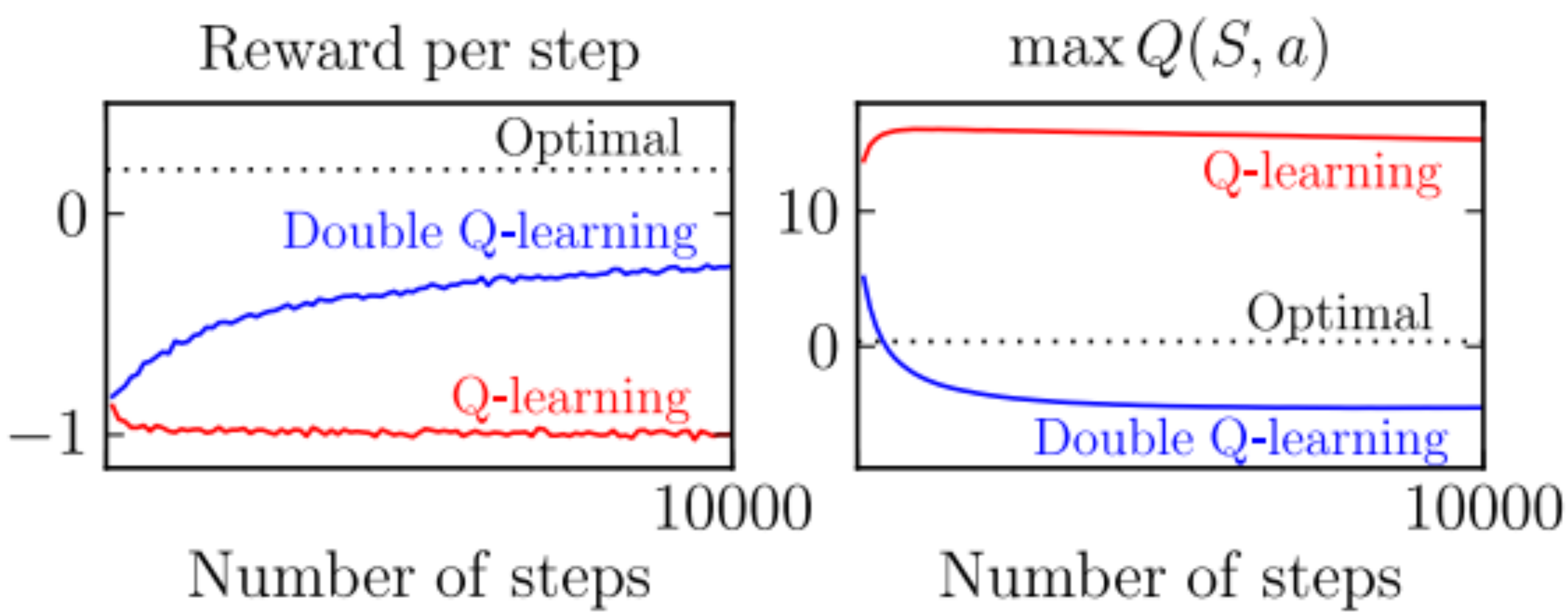


van Hasselt et al., Double Q Learning, NIPS 2010

Evaluation: Q-Learning & Double Q-Learning (Cont.)

- Example: Grid World with 9 states and 4 actions
  - Reward at the terminal state G: +5
  - Reward at any non-terminal state: -12 or +10 (random)
  - Under an optimal policy, an episode ends after 5 actions





Left: average reward per step. Right: maximal value in start state S.
ϵ-greedy exploration, $\epsilon = 1/\sqrt{n}$, $\alpha = 1/n$, $\gamma = 0.95$

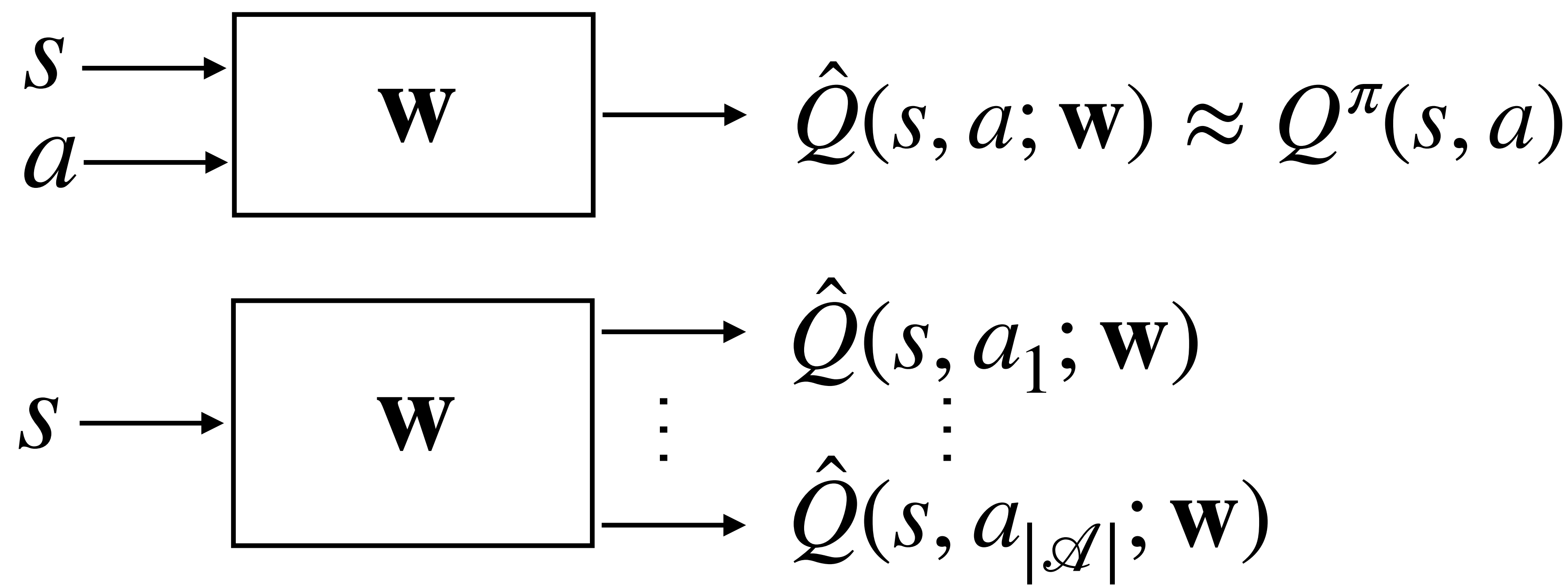
Double Q-learning may lead to "under-estimation" (with finite samples)

van Hasselt et al., Double Q Learning, NIPS 2010

Q-Learning With Value Function Approximation

Q-Learning With VFA

- Idea: $Q^\pi(s,a) \approx \hat{Q}(s,a;\mathbf{w})$ using some parametric function



- Question: Which way is preferred?

- Question: How to learn a proper $\mathbf{w}$?

Minimize MSE between estimated Q-value and a “target”

Q-Value Function Approximation With an Oracle

- Goal: Find $\mathbf{w}$ that minimizes Bellman error w.r.t. $\hat{Q}(s, a; \mathbf{w})$

$$\mathbf{w} = \arg\min_{\mathbf{w}'} \underbrace{\mathbb{E}_{(s,a,r,s')\sim\rho}\left[\left(r + \gamma \max_{a' \in A} \hat{Q}(s', a'; \mathbf{w}') - \hat{Q}(s, a; \mathbf{w}')\right)^2\right]}_{=: F(\mathbf{w}')}$$

- Question: If $\hat{Q}(s, a; \mathbf{w})$ perfectly matches $Q^*(s, a)$, the loss $=$?

- The loss can be minimized by iterative GD / SGD update:

$$\begin{aligned}
\mathbf{w}_{k+1} &= \mathbf{w}_k + \alpha_k \nabla_{\mathbf{w}} F(\mathbf{w}) \\
&= \mathbf{w}_k + \alpha_k \mathbb{E}_{(s,a,r,s')\sim\rho}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}_k) - \hat{Q}(s, a; \mathbf{w}_k)\right) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w}_k)\right] \\
&\approx \mathbf{w}_k + \alpha_k \sum_{(s,a,r,s') \in D} \left[\left(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}_k) - \hat{Q}(s, a; \mathbf{w}_k)\right) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w}_k)\right]
\end{aligned}$$

- Question: How does “off-policy learning” come into play? And ρ?

Q-Learning With Value Function Approximation: A Prototypic "Online" Algorithm

- Q-Learning With Value Function Approximation:

Step 1: Initialize $\mathbf{w}$ for $Q(s, a; \mathbf{w})$ and initial state s_0

Step 2: For each step $t = 0, 1, 2, \cdots$

Select a_t using ε-greedy w.r.t $Q(s_t, a; \mathbf{w})$

Observe (r_{t+1}, s_{t+1})

Update $\mathbf{w}$ as follows:

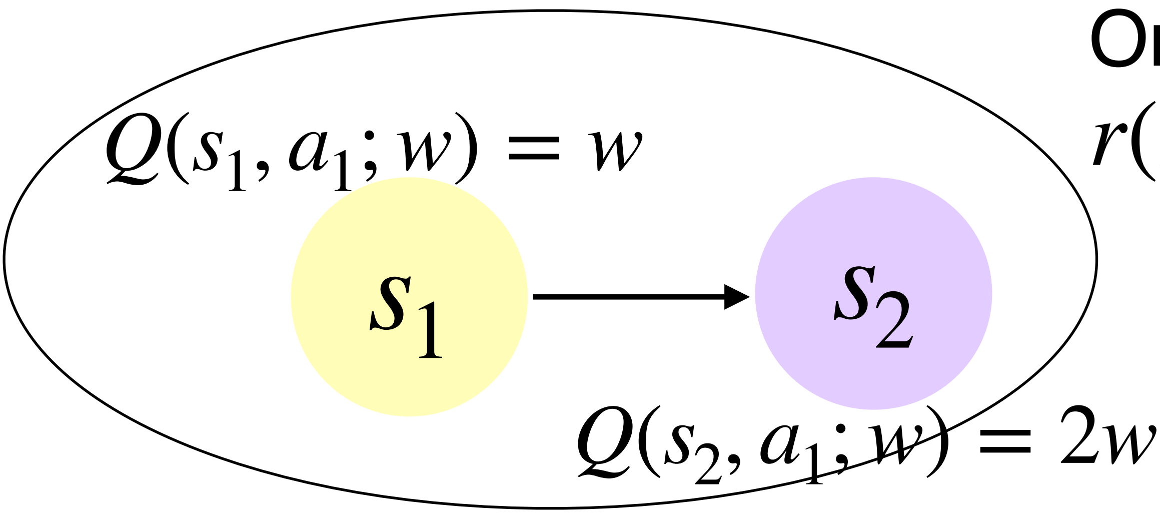
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_k \left[\left(r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \mathbf{w}) - \hat{Q}(s_t, a_t; \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t; \mathbf{w}) \right]$$

Does Q-Learning converge with function approximation?

In general, Q-learning may <u>diverge</u> with VFA
(even with linear VFA)

Divergence of Q-Learning With VFA

- Example: 2 states in a potentially large MDP (with linear VFA)



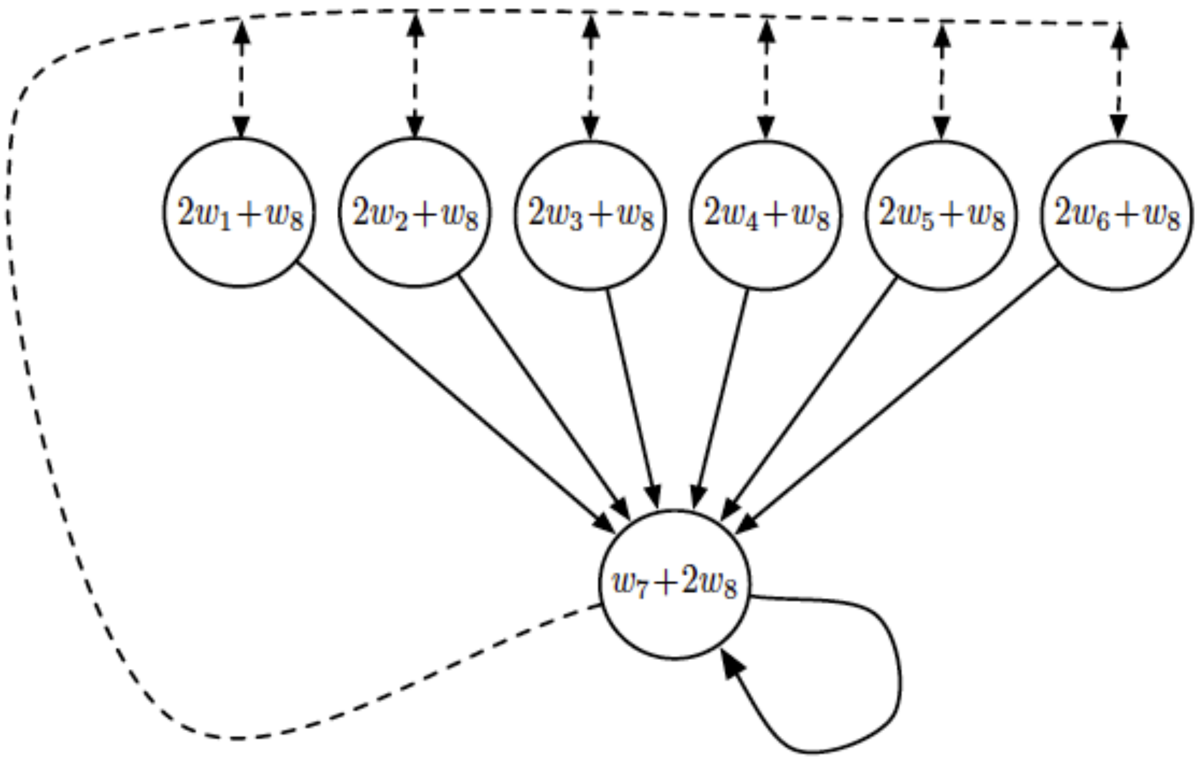
Only 1 action a_1 available at state s_1, and $r(s_1, a_1) = 0$, $P(s_2 \mid s_1, a_1) = 1$

- Question: Given $w_k = 1, \gamma = 0.9$. Under Q-learning, what is w_{k+1}?
- Question: What will happen if we keep using the transition $s_1 \rightarrow s_2$ to update w?

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_k \Big[\Big(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w}) \Big) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w}) \Big]$$

Divergence of Q-Learning: A Classic Example

- Example: Baird's counterexample (with linear VFA)



$(\sqrt{J} = \|\Pi T Q_w - Q_w\|)$

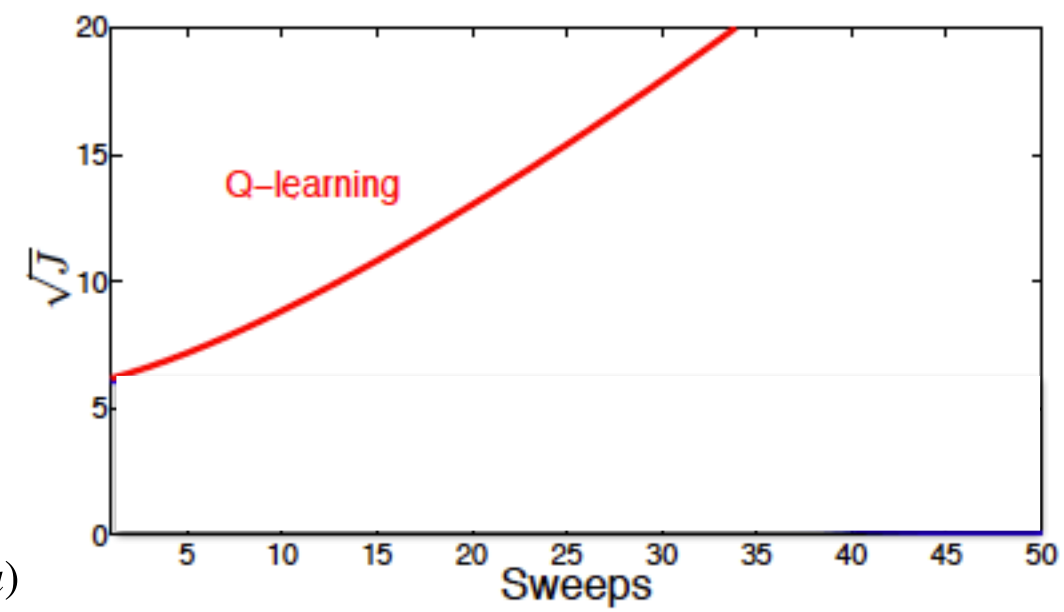
$\pi(\text{solid}|\cdot) = 1$

$b(\text{dashed}|\cdot) = 6/7$

$b(\text{solid}|\cdot) = 1/7$

$\gamma = 0.99$

$R(s,a) = 0, \forall (s,a)$



Baird, Residual Algorithms: Reinforcement Learning with Function Approximation, ICML 1995

Maei et al., Towards Off-Policy Learning Control with Function Approximation, ICML 2010

- Fortunately, Q learning usually converges in practice when target policy π is close to the behavior policy β (e.g., ε-greedy)

(Sutton and Barto, Chapter 11.2)

Why Q-Learning Diverges?

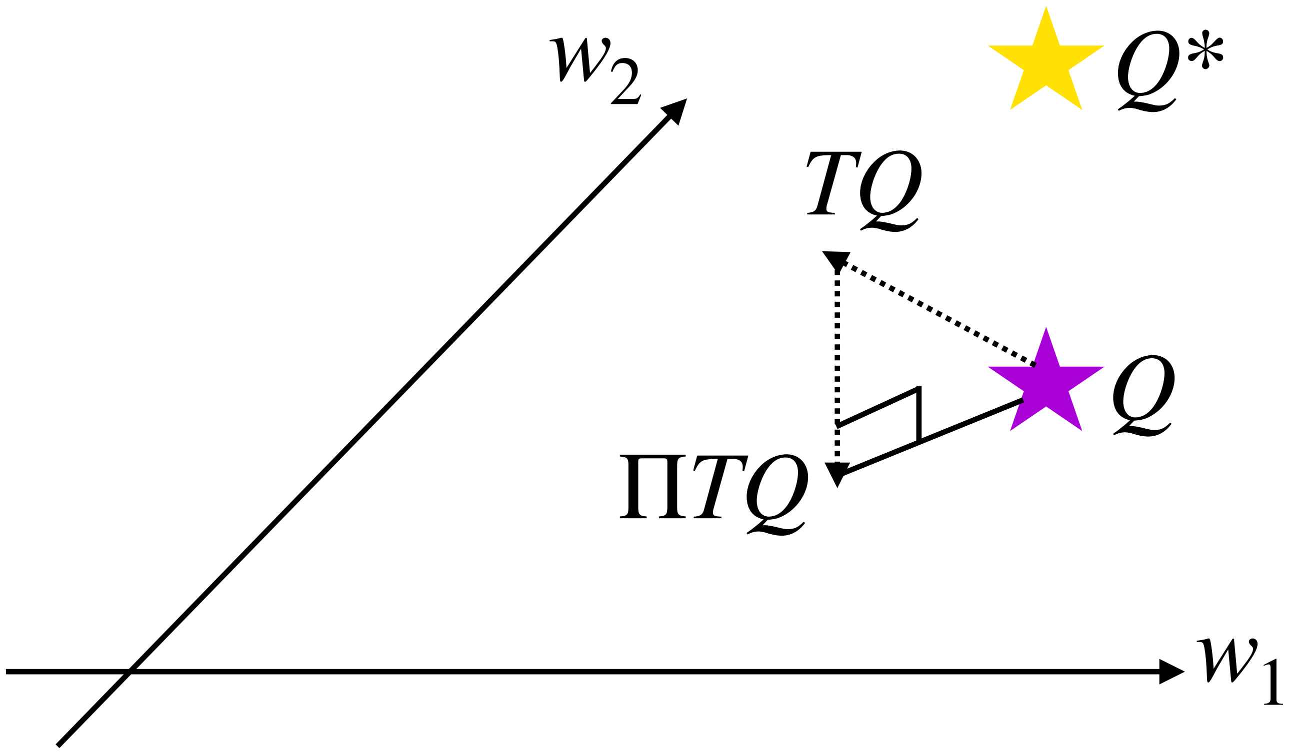
An Illustration of Divergence of Q-Learning With VFA

- **Q-learning with VFA can be equivalently decompose into two steps**:

  Step 1. Let $y_i = r(s_i, a_i) + \gamma \max_a Q_{\mathbf{w}}(s_i', a)$, for each i

  Step 2. Set $\mathbf{w} \leftarrow \arg\min_{\mathbf{w}'} \frac{1}{2} \sum_i \|Q_{\mathbf{w}'}(s_i, a_i) - y_i\|_2^2$

- Bellman operators are <u>contractions</u>, but it can become an <u>expansion</u> with value function approximation fitting



(Sutton and Barto, Chapter 11.4)

Deep Q-Network (DQN)

Mnih et al., Human-level control through deep reinforcement learning, Nature 2015

What is DQN?

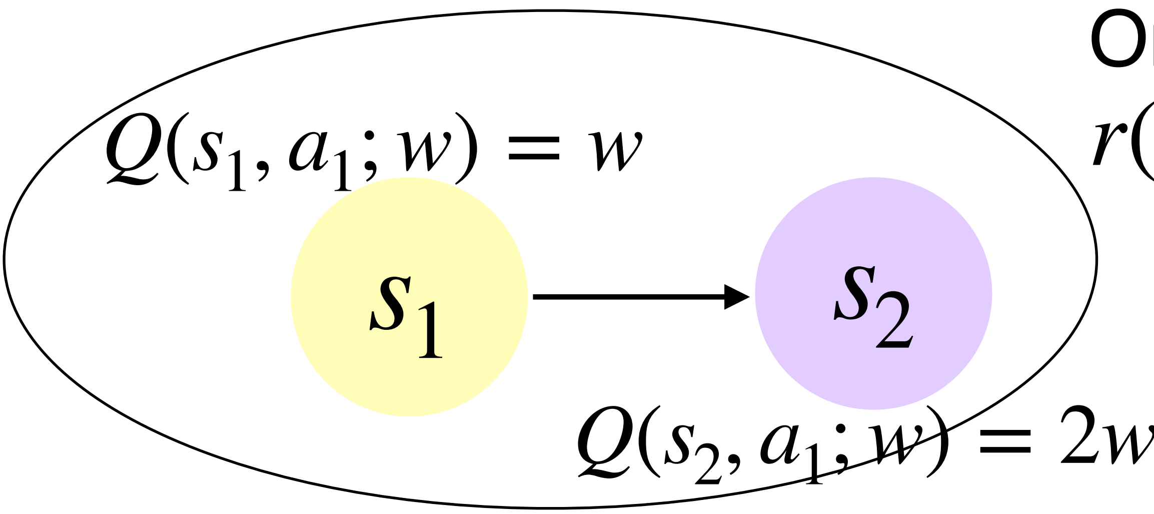
- DQN = Combine Q-Learning with NN-based nonlinear VFA
- Recall: ”Q-learning + VFA + Off-policy learning” has divergence issue
- To tackle the divergence issue, DQN applies two techniques:

  (T1) Experience replay (via a replay buffer)

  (T2) Using 2 networks: Q-network and <u>target network</u>

Recall: Divergence of Q-Learning With VFA

- Example: 2 states in a potentially large MDP (with linear VFA)



Only 1 action a_1 available at state s_1, and $r(s_1, a_1) = 0$, $P(s_2 \mid s_1, a_1) = 1$

- Question: Given $w_k = 1, \gamma = 0.9$. Under Q-learning, what is w_{k+1}?
- Question: What will happen if we keep using the transition $s_1 \rightarrow s_2$ to update w?

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_k \Big[\big(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w}) \big) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w}) \Big]$$

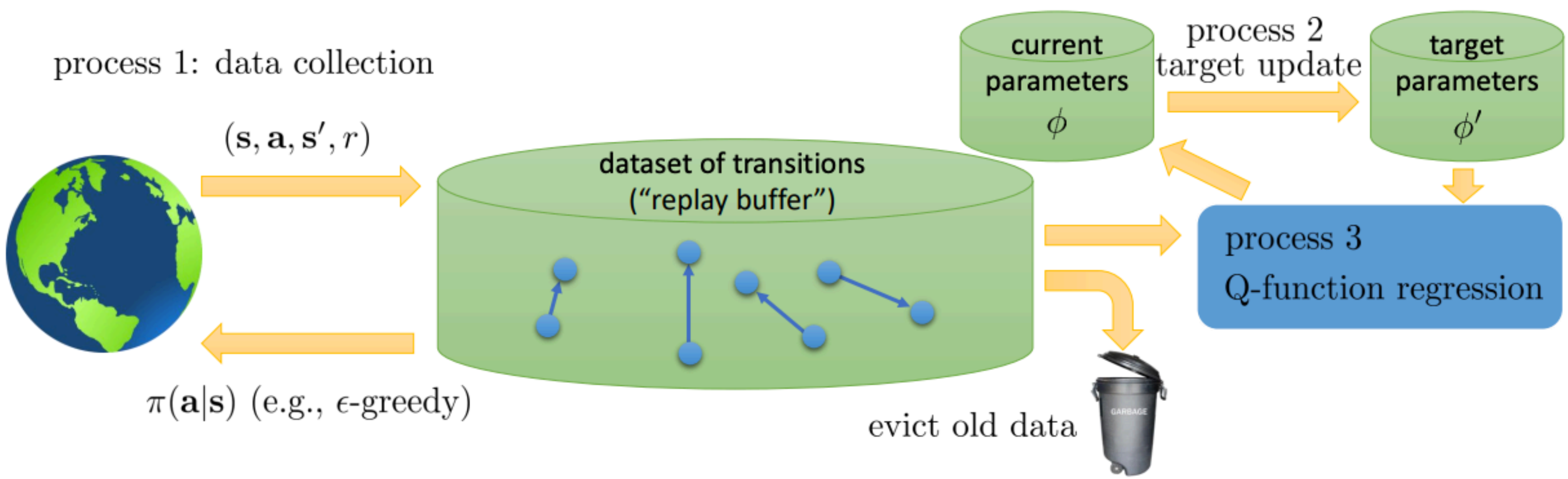
- Insight: Divergence can occur if the following two things happen

1. Keep using the transition $(s_1, a_1, 0, s_2)$ to update Q function

2. Using the latest w_k in the TD target for the update of iteration $(k+1)$

(T1) Experience Replay

- Idea: 1. Store the previous experiences (s, a, s', r) into a buffer
  2. Sample a mini-batch from the buffer at each update
  (similar to mini-batch SGD in supervised learning)



- Purpose:

1. *Stable learning*: Break correlations between successive updates
2. *Data efficiency*: Reuse interactions with environment

(Figure Credit: Sergey Levine)

(T2) Target Network and Q-Network

- Idea: Use a separate target network (denoted by $Q_{\bar{\mathbf{w}}}$) that are updated only periodically

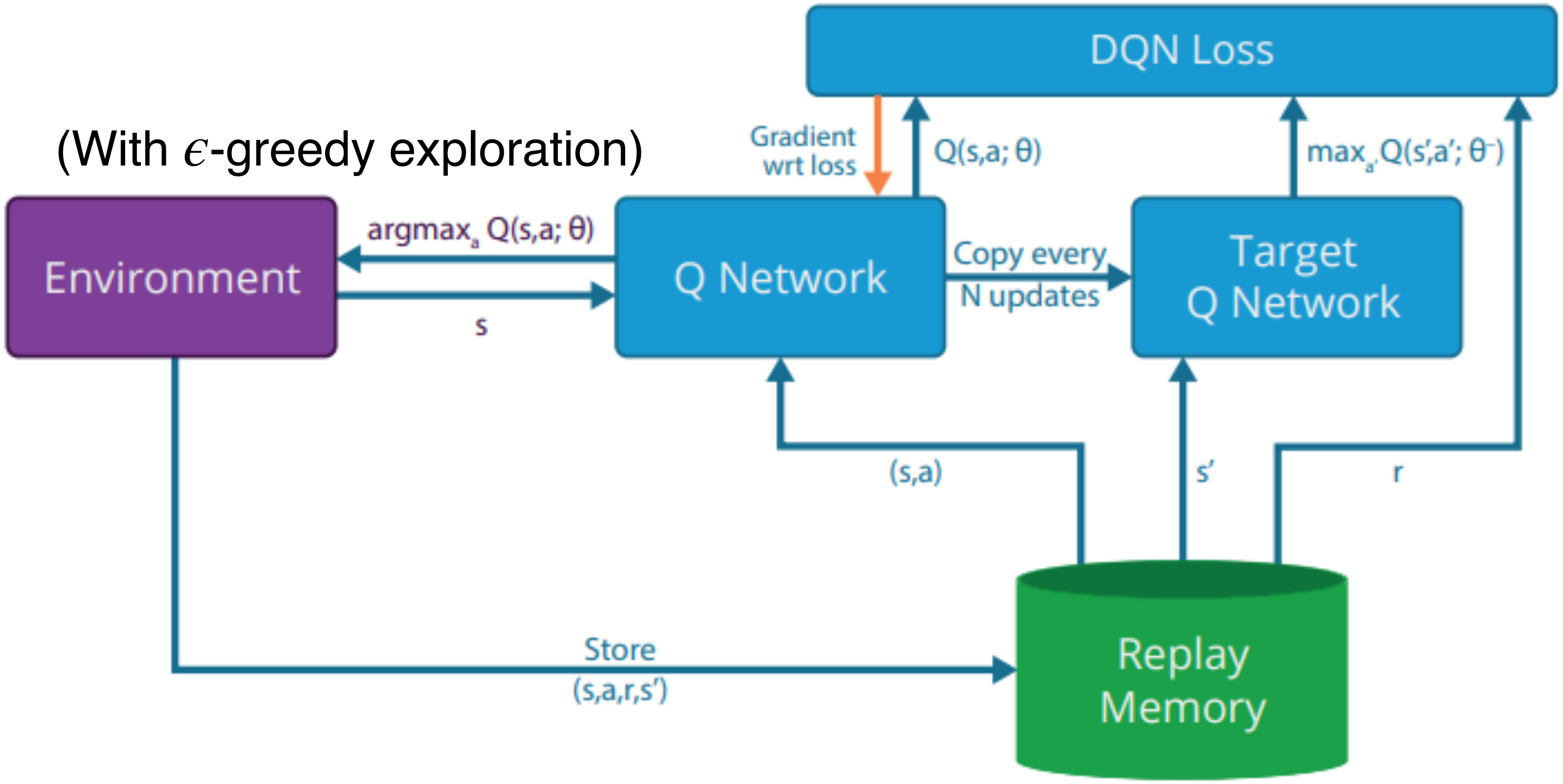
- Update of the Q-network:

$$\Delta \mathbf{w}_k = \alpha_{\mathbf{w}} \cdot \sum_{(s,a,r,s')\in D} \Big(\overbrace{r + \gamma \max_{a'} Q_{\bar{\mathbf{w}}}(s', a')}^{\text{target}} - Q_{\mathbf{w}_k}(s, a) \Big) \nabla_{\mathbf{w}} Q_{\mathbf{w}}(s, a)|_{\mathbf{w}=\mathbf{w}_k}$$

- Purpose: Mitigate divergence

Architecture of Vanilla DQN

(With ϵ-greedy exploration)



$$L(\mathbf{w}) = \sum_{(s,a,r,s')\in D} \frac{1}{2}\left[\left(r + \gamma \max_{a'} \hat{Q}_{\bar{\mathbf{w}}}(s',a') - \hat{Q}_{\mathbf{w}}(s,a)\right)^2\right]$$

$$\nabla_{\mathbf{w}} L(\mathbf{w})\Big|_{\mathbf{w}=\mathbf{w}_k} = \sum_{(s,a,r,s')\in D} \left[\left(r + \gamma \max_{a'} \hat{Q}_{\bar{\mathbf{w}}}(s',a') - \hat{Q}_{\mathbf{w}_k}(s,a)\right) \nabla_{\mathbf{w}} \hat{Q}_{\mathbf{w}_k}(s,a)\right]$$

- Question: How to sample from the replay buffer?
- Question: How to update the replay buffer?

 (Figure Source: https://kaustabpal.github.io/dqn)

Pseudo Code of DQN

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N — 1. Replay buffer

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$ — 2. Target network

For episode $= 1, M$ **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$ — 3. ε-greedy exploration

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

4. Update Q-network by mini-batch SGD

(Need to handle terminal states)

    Every C steps reset $\hat{Q} = Q$

  End For

End For

5. Periodic update of the target network

Mnih et al., Human-level control through deep reinforcement learning, Nature 2015

Some recent results on the convergence of Q-learning with function approximation

[NeurIPS 2020]

A new convergent variant of Q-learning with linear function approximation

Diogo S. Carvalho **Francisco S. Melo** **Pedro A. Santos**
INESC-ID & Instituto Superior Técnico, University of Lisbon
Lisbon, Portugal
{diogo.s.carvalho, pedro.santos}@tecnico.ulisboa.pt
fmelo@inesc-id.pt

Abstract

In this work, we identify a novel set of conditions that ensure convergence with probability 1 of Q-learning with linear function approximation, by proposing a two time-scale variation thereof. In the faster time scale, the algorithm features an update similar to that of DQN, where the impact of bootstrapping is attenuated by using a Q-value estimate akin to that of the target network in DQN. The slower time-scale, in turn, can be seen as a modified target network update. We establish the convergence of our algorithm, provide an error bound and discuss our results in light of existing convergence results on reinforcement learning with function approximation. Finally, we illustrate the convergent behavior of our method in domains where standard Q-learning has previously been shown to diverge.

[NeurIPS 2023]

On the Convergence and Sample Complexity Analysis of Deep Q-Networks with ε-Greedy Exploration

Shuai Zhang
New Jersey Institute of Technology

Hongkang Li
Rensselaer Polytechnic Institute

Meng Wang
Rensselaer Polytechnic Institute

Miao Liu
IBM Research

Pin-Yu Chen
IBM Research

Songtao Lu
IBM Research

Sijia Liu
Michigan State University

Keerthiram Murugesan
IBM Research

Subhajit Chaudhury
IBM Research

Abstract

This paper provides a theoretical understanding of Deep Q-Network (DQN) with the ε-greedy exploration in deep reinforcement learning. Despite the tremendous empirical achievement of the DQN, its theoretical characterization remains underexplored. First, the exploration strategy is either impractical or ignored in the existing analysis. Second, in contrast to conventional Q-learning algorithms, the DQN employs the target network and experience replay to acquire an unbiased estimation of the mean-square Bellman error (MSBE) utilized in training the Q-network. However, the existing theoretical analysis of DQNs lacks convergence analysis or bypasses the technical challenges by deploying a significantly over-parameterized neural network, which is not computationally efficient. This paper provides the first theoretical convergence and sample complexity analysis of the practical setting of DQNs with ε-greedy policy. We prove an iterative procedure with decaying ε converges to the optimal Q-value function geometrically. Moreover, a higher level of ε values enlarges the region of convergence but slows down the convergence, while the opposite holds for a lower level of ε values. Experiments justify our established theoretical insights on DQNs.