

# CS 277: Control and Reinforcement Learning

Winter 2022

## Lecture 4: Deep Q-Learning

Roy Fox

Department of Computer Science

Bren School of Information and Computer Sciences

University of California, Irvine



# Logistics

---

assignments

- Assignments 1 due **next Tuesday**

quizzes

- Quiz 2 to be published soon
- Due **next Wednesday**

# Today's lecture

---

Fitted Q-Iteration

Deep Q-Learning

DQN tricks

# Fitted Value-Iteration (FVI)

---

## Algorithm Value Iteration

---

Initialize some value function  $V$


**repeat**

**for each** state  $s$

Update  $V(s) \leftarrow \max_a (r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p} [V(s')])$

---

- Fitted Value-Iteration (FVI):

$$\theta^{i+1} \leftarrow \arg \min_{\theta} \mathbb{E}_{s \sim \mu} [(\max_a (r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p} [V_{\theta^i}(s')]) - V_{\theta}(s))^2]$$


- ▶ For some state distribution  $\mu$
- ▶ Can use losses other than square

MF

$\theta$

DP

$\pi'$

max

MF

$\theta$

DP

$\pi'$

max

# Fitted Q-Iteration (FQI)

- Action-value iteration:  $Q(s, a) \leftarrow r(s, a) + \gamma \mathbb{E}_{(s'|s, a) \sim p} [\max_{a'} Q(s', a')]$

- Fitted Q-Iteration (FQI):

$$\theta^{i+1} \leftarrow \arg \min_{\theta} \mathbb{E}_{(s, a) \sim \mu} [(r(s, a) + \gamma \mathbb{E}_{(s'|s, a) \sim p} [\max_{a'} Q_{\theta^i}(s', a')]) - Q_{\theta}(s, a)]^2]$$

- ▶ For some state-action distribution  $\mu$
- We can also combine
  - ▶ Policy evaluation: MC with function approximation
  - ▶ Policy improvement: greedy

MF

$\theta$

DP

$\pi'$

max

MF

$\theta$

DP

$\pi'$

max

# Q-Learning

MF

$\theta$

DP

$\pi'$

max

---

## Algorithm Q-Learning

---

Initialize  $Q$

$s \leftarrow$  reset state

**repeat**

    Take some action  $a$

    Receive reward  $r$

    Observe next state  $s'$

    Update  $Q(s, a) \rightarrow \begin{cases} r & s' \text{ terminal} \\ r + \gamma \max_{a'} Q(s', a') & \text{otherwise} \end{cases}$

$s \leftarrow$  reset state if  $s'$  terminal, else  $s \leftarrow s'$

---

# Sampling-based Fitted Q-Iteration

- FQI can be **model-free** by sampling from  $p$ 
  - We can sample using **environment interaction** with some  $\pi'$ , if  $\mu = p_{\pi'}$
  - Or sample using a **simulator** we can reset to any state  $s \sim \mu$
  - Anyway, this is **off-policy** from the greedy policy  $\arg \max_a Q_{\theta}(s, a)$

---

## Algorithm Sampling-based Fitted Q-Iteration

---

Initialize  $\theta$

**repeat**

    Sample a batch  $(\vec{s}, \vec{a}) \sim \mu$

    Feed to simulator to get batch  $(\vec{r}, \vec{s}')$

    Descend  $\mathcal{L}_{\theta} = (\vec{r} + \gamma \max_{\vec{a}'} Q_{\bar{\theta}}(\vec{s}', \vec{a}') - Q_{\theta}(\vec{s}, \vec{a}))^2$

MF

$\theta$

DP

$\pi'$

max

# Today's lecture

---

Fitted Q-Iteration

**Deep Q-Learning**

DQN tricks

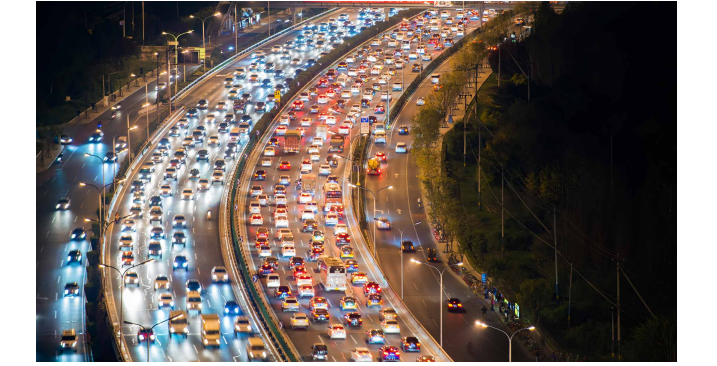


# Experience policy

---

- Which distribution should the **training data** have?
  - The policy may not be good on other distributions / unsupported states
  - $\Rightarrow$  ideally, the **test** distribution  $p_\pi$  for the **final**  $\pi$
- **On-policy methods** (e.g. MC): must use on-policy data (from the **current**  $\pi$ )
- **Off-policy methods** (e.g. Q) can use different policy (or even non-trajectories)
  - But both should eventually use  $p_\pi$  or suffer train–test distribution mismatch

# Exploration policies



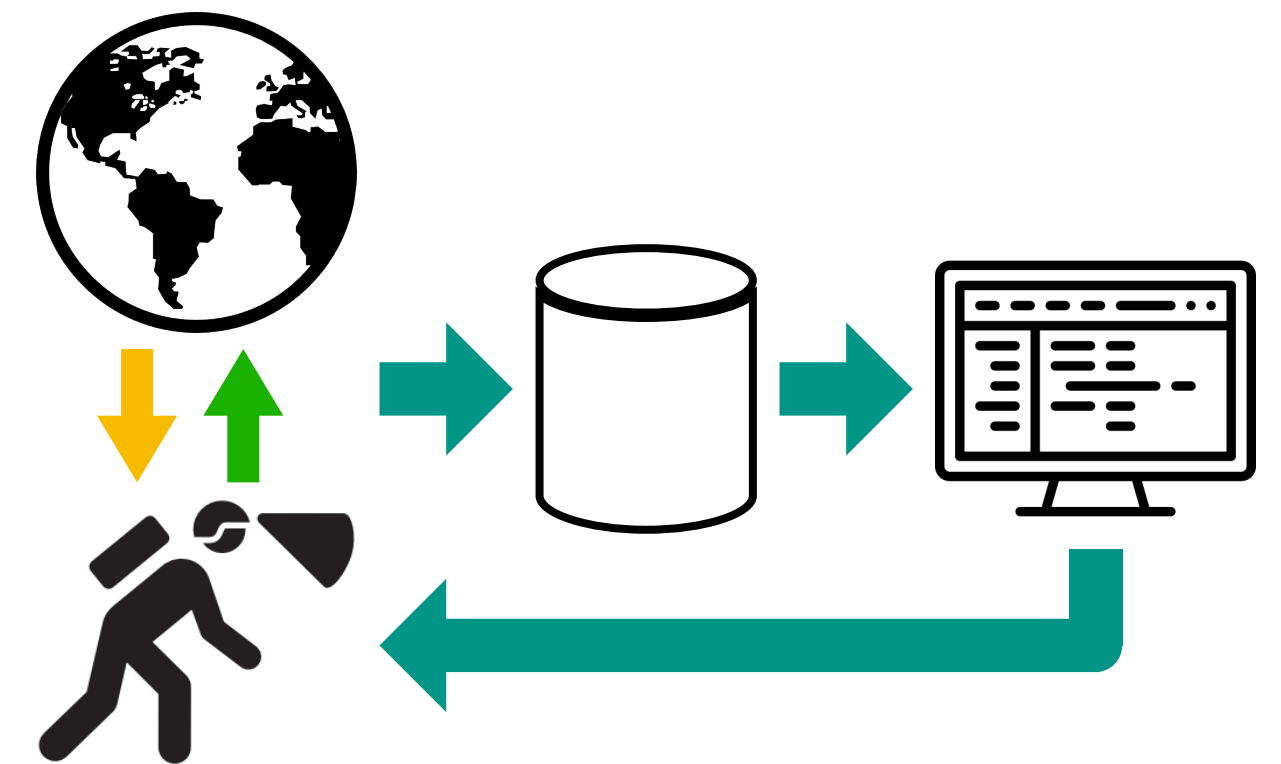
- Example: I tried **route 1**: {40, 20, 30}; **route 2**: {30, 25, 40}
  - Suppose **route 1** really has expected time **30min**, should you commit to it forever?
- To avoid **overfitting**, we must try all actions infinitely often
- **$\epsilon$ -greedy exploration**: select uniform action with prob.  $\epsilon$ , otherwise greedy
- **Boltzmann exploration**:

$$\pi(a | s) = \text{soft max}_a(Q(s, a); \beta) = \frac{\exp(\beta Q(s, a))}{\sum_{\bar{a}} \exp(\beta Q(s, \bar{a}))}$$

- Becomes uniform as the **inverse temperature**  $\beta \rightarrow 0$ , greedy as  $\beta \rightarrow \infty$

# Experience replay

- On-policy methods are **inefficient**: throw out all data with each policy update
- Off-policy methods can keep the data = **experience replay**
  - **Replay buffer**: dataset of past experience
  - **Diversifies** the experience (beyond current trajectory)
- Variants differ on
  - **How often** to add data vs. sample data
  - How to **sample** from the buffer
  - When to **evict** data from the buffer, and which



# Why use target network?

- Fitted-Q loss:  $\mathcal{L}_\theta = (r + \gamma \max_{a'} Q_{\bar{\theta}}(s', a') - Q_\theta(s, a))^2$

no gradient from the target term

- Target network = lagging copy of  $Q_\theta(s, a)$

- ▶ Periodic update:  $\bar{\theta} \leftarrow \theta$  every  $T_{\text{target}}$  steps

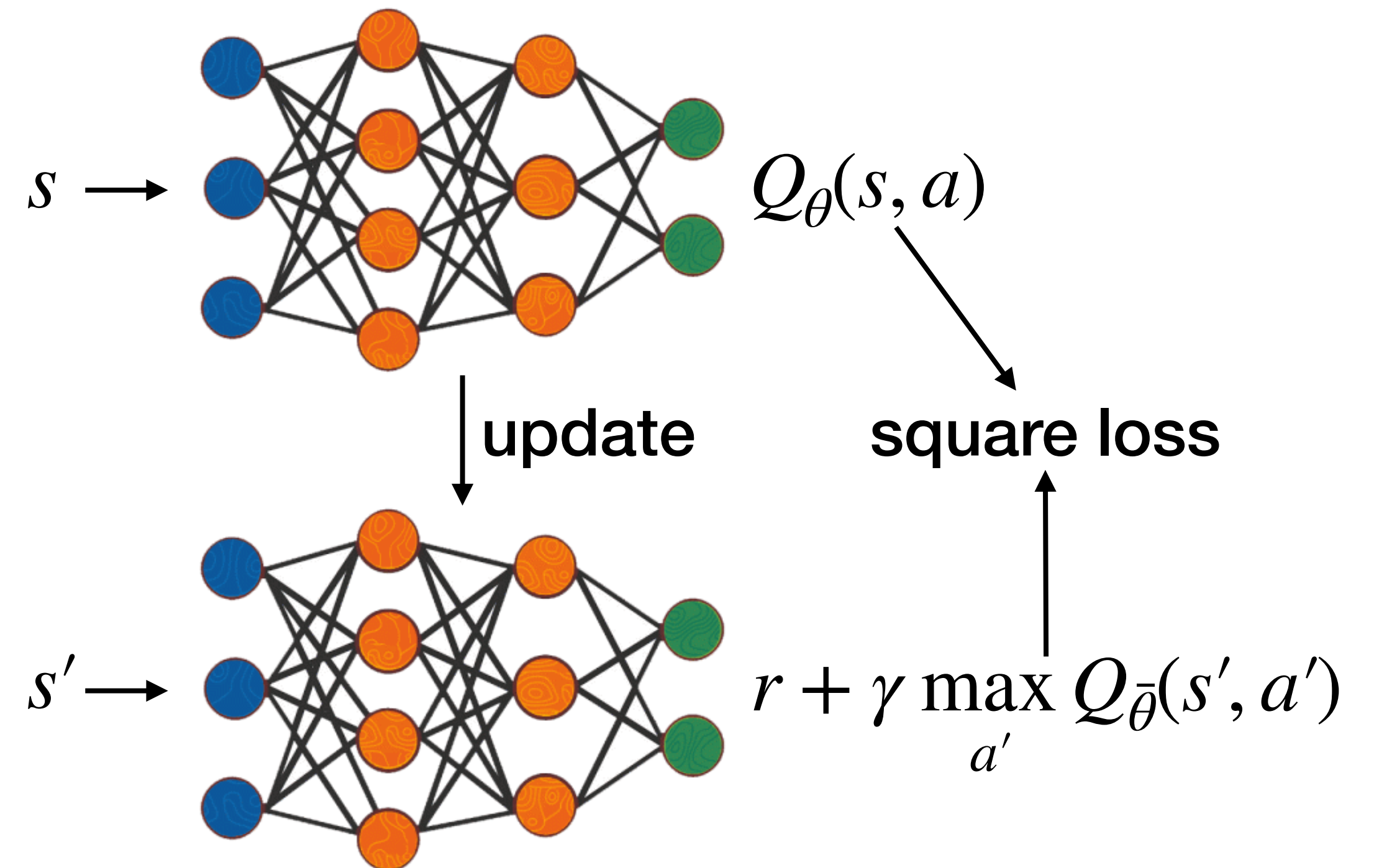
- ▶ Exponential update:  $\bar{\theta} \leftarrow (1 - \eta)\bar{\theta} + \eta\theta$

- $Q_{\bar{\theta}}$  is more stable

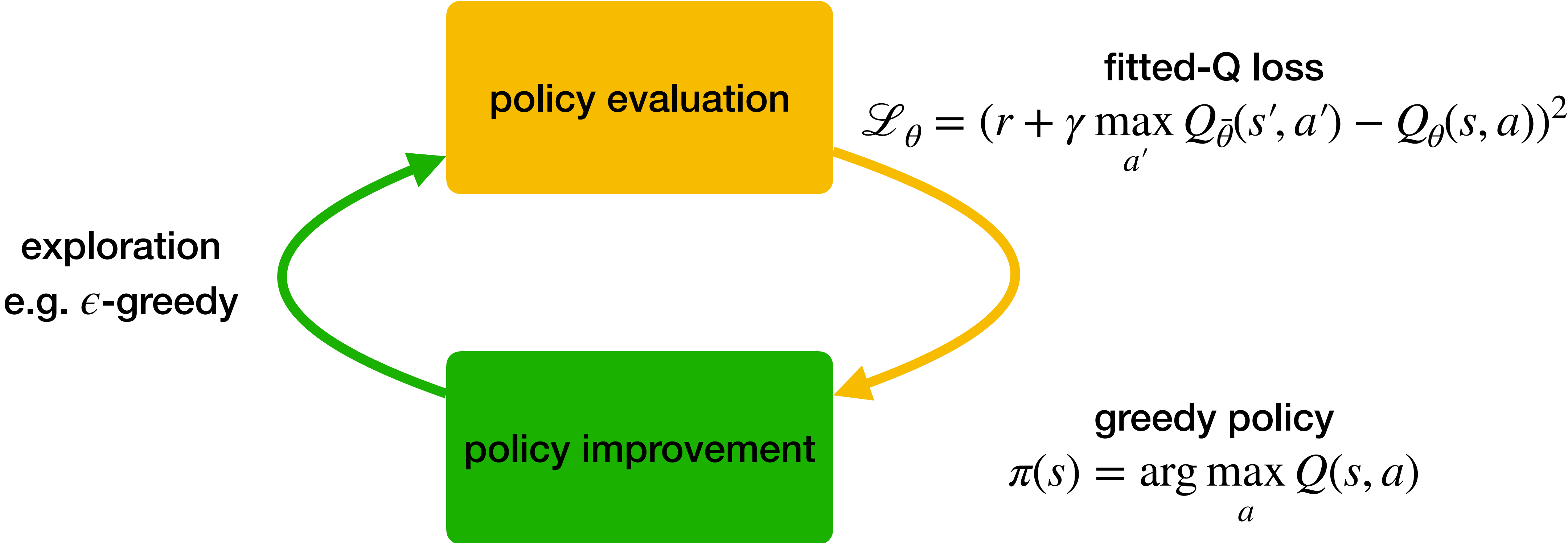
- ▶ Less of a moving target

- ▶ Less sensitive to data  $\Rightarrow$  less variance

- But  $\bar{\theta} \neq \theta$  introduces bias



# Putting it all together: DQN





# Deep Q-Learning (DQN)

---

## Algorithm DQN

---

Initialize  $\theta$ , set  $\bar{\theta} \leftarrow \theta$

$s \leftarrow$  reset state

**for each** interaction step

Sample  $a \sim \epsilon$ -greedy for  $Q_{\theta}(s, \cdot)$

Get reward  $r$  and observe next state  $s'$

Add  $(s, a, r, s')$  to replay buffer  $\mathcal{D}$

Sample batch  $(\vec{s}, \vec{a}, \vec{r}, \vec{s}') \sim \mathcal{D}$

$$y_i \leftarrow \begin{cases} r_i & s'_i \text{ terminal} \\ r_i + \gamma \max_{a'} Q_{\bar{\theta}}(s'_i, a') & \text{otherwise} \end{cases}$$

Descend  $\mathcal{L}_{\theta} = (\vec{y} - Q_{\theta}(\vec{s}, \vec{a}))^2$

every  $T_{\text{target}}$  steps, set  $\bar{\theta} \leftarrow \theta$

$s \leftarrow$  reset state if  $s'$  terminal, else  $s \leftarrow s'$

MF

$\theta$

DP

$\pi'$

max

# Today's lecture

---

Fitted Q-Iteration

Deep Q-Learning

**DQN tricks**

# Value estimation bias

- Q-value estimation is optimistically **biased**
- **Jensen's inequality**: for a random vector  $Q$

$$\mathbb{E}[\max_a Q_a] \geq \max_a \mathbb{E}[Q_a]$$

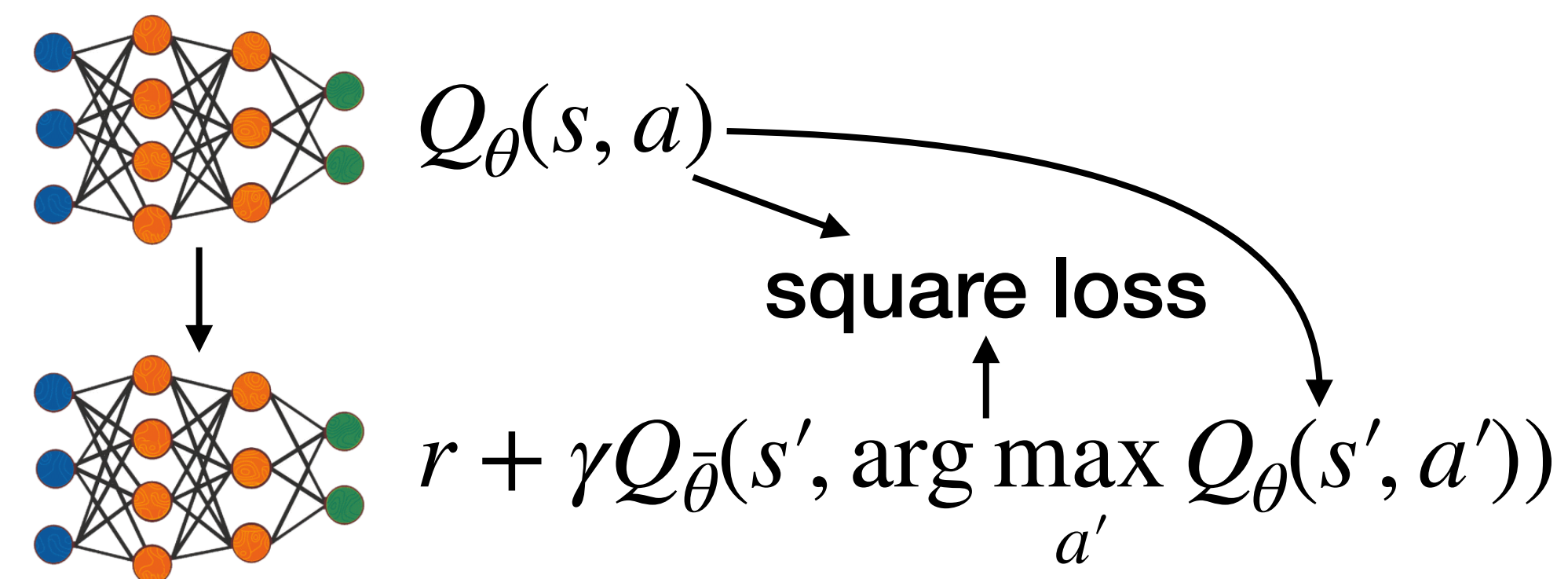
- While there's **uncertainty** in  $Q_{\bar{\theta}}$ ,  $\max_{a'} Q_{\bar{\theta}}(s', a')$  is positively biased
- So how can this **converge**?
  - As certainty increases, the bias of each update decreases
  - Existing bias attenuates with repeated discounting by  $\gamma$



# Double Q-Learning

- **Idea:** keep two value estimates  $Q_1$  and  $Q_2$ 
  - ▶ **Update:**  $Q_i(s, a) \rightarrow r + \gamma Q_{-i}(s', \arg \max_{a'} Q_i(s', a'))$   
    *-i = the other*

- How to use this with DQN?
- **Idea 1:** use target network as the other estimate



- **Idea 2:** Clipped Double Q-Learning

$$Q_{\theta_i}(s, a) \rightarrow r + \gamma \min_{i=1,2} Q_{\bar{\theta}_i}(s', \arg \max_{a'} Q_{\theta_i}(s', a'))$$

# Prioritized Experience Replay

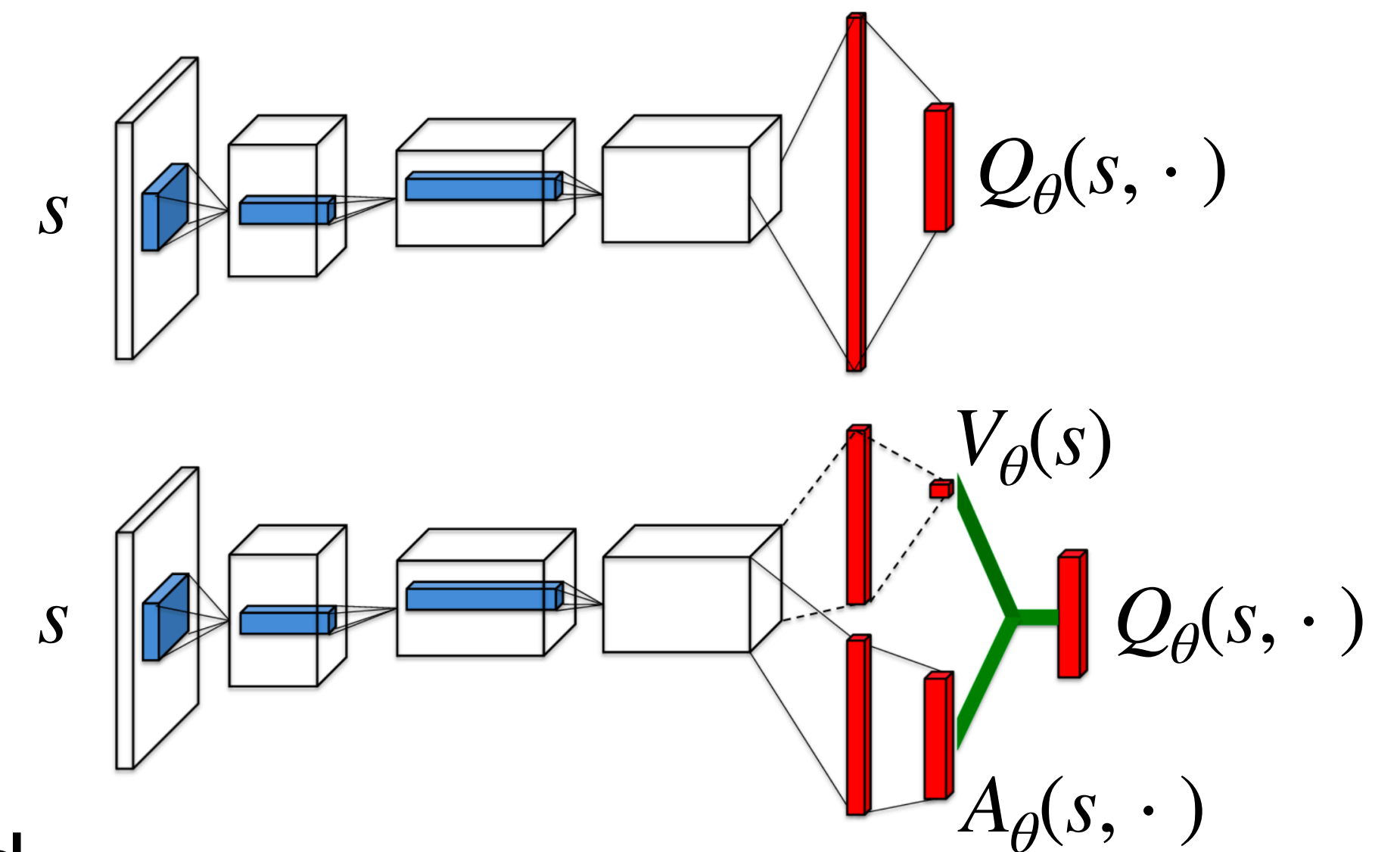
- **Bellman error** (= TD error):  $\delta(s, a, r, s') = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$ 
  - **Optimality**:  $\delta \equiv 0$ ; that's why we usually descend the square loss  $\delta^2$
- Experience with **high error**  $\Rightarrow$  more important to see
- **Prioritized Experience Replay**:
  - Sample instance  $i$  with prob.  $p_i \propto \delta_i^\omega$ ; e.g.  $\omega = 0.6$
  - Update with **Importance Sampling** (IS) weight  $(m \cdot p_i)^{-\beta}$ ; e.g.  $\beta = 0.4$
- $\delta$  is computed during the updates; new experience is weighted  $\max_i \delta_i^\omega$

# Dueling Networks

- Advantage function:  $A_{\pi}(s, a) = Q_{\pi}(s, a) - V_{\pi}(s)$
- $A_{\pi}(s, a)$  can be more consistent across states with similar effect of actions

- ▶ Even if their value  $V_{\pi}(s)$  is very different

- $V_{\pi}(s)$  is a scalar, which can be easier to learn



- Issue:  $Q = (V + c) + (A - c)$  is underdetermined

- ▶ Stabilize with  $Q(s, a) = V(s) + \left( A(s, a) - \text{mean}_{\bar{a}} A(s, \bar{a}) \right)$

# Multi-step Q Learning

- MC is **high variance** but **unbiased**:  $Q(s_t, a_t) \rightarrow R_{\geq t} = \sum_{t' \geq t} \gamma^{t'-t} r_{t'}$
- TD is **lower variance** but **biased**:  $Q(s_t, a_t) \rightarrow r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ 
  - ▶ Because  $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$  isn't really the next-step value, while still learning
- Let's trade them off,  **$n$ -step Q-Learning**:

$$Q(s_t, a_t) \rightarrow r_t + \gamma r_{t+1} + \dots + \gamma^{n-1} r_{t+n-1} + \gamma^n \max_{a_{t+n}} Q(s_{t+n}, a_{t+n})$$



# Rainbow DQN

- **Rainbow DQN** = DQN + a powerful combination of tricks
  - ▶ Double Q-Learning
  - ▶ Prioritized Experience Replay
  - ▶ Dueling Networks
  - ▶ Multi-step Q-Learning
  - ▶ Distributional RL
  - ▶ Noisy Nets



# Recap

---

- RL algorithms can be implemented with **function approximation**
- There are (at least) 2 important policies
  - The **learner policy** — should be the best possible (e.g. greedy)
  - The **experience policy** — should explore (e.g.  $\epsilon$ -greedy)
- **Replay buffer**: store data for longer (off-policy), diversify
- **Target network**: reduce variance, stabilize the target
- In practice, add lots of **tricks** and heuristics to the theory