

Approximate Value Iteration

Chen-Yu Wei

Review: Value Iteration

$V^*(s)$:= maximum expected total reward starting from state s

$Q^*(s, a)$:= maximum expected total reward starting from state s and taking action a **for one step**, and then following the optimal strategy

Bellman Equations

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s')$$

$$V^*(s) = \max_a Q^*(s, a)$$

Value Iteration (algorithm to find Q^*/V^*)

Initialize $Q_0(s, a) \leftarrow 0, V_0(s) \leftarrow 0$ for all (s, a)

For $i = 1, 2, \dots$

$$Q_i(s, a) \leftarrow R(s, a) + \gamma \sum_{s'} P(s'|s, a) V_{i-1}(s') \quad \text{for all } (s, a)$$

$$V_i(s) = \max_a Q_i(s, a) \quad \text{for all } s$$

If $|Q_i(s, a) - Q_{i-1}(s, a)| \leq \epsilon$ for all (s, a) : **break**

Review: Value Iteration

For $k = 1, 2, \dots$

$$\forall s, a, \quad Q_k(s, a) \leftarrow \underbrace{R(s, a)}_{\text{unknown}} + \gamma \sum_{s'} \underbrace{P(s'|s, a)}_{\text{unknown}} \max_{a'} Q_{k-1}(s', a')$$

Idea: In each iteration, use multiple samples to estimate the right-hand side.

Value Iteration with Samples

For $k = 1, 2, \dots$

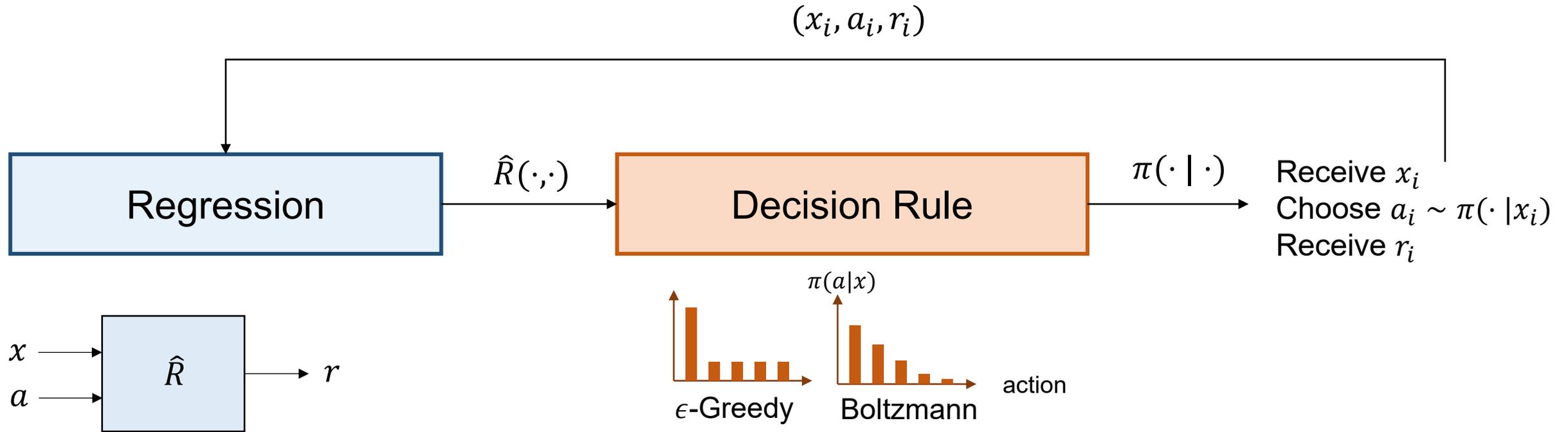
Obtain N samples $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$ where $\mathbb{E}[r_i] = R(s_i, a_i)$, $s'_i \sim P(\cdot | s_i, a_i)$

Perform **regression** on $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$ to find Q_k such that

$$\forall s, a, \quad Q_k(s, a) \approx R(s, a) + \gamma \sum_{s'} P(s' | s, a) \max_{a'} Q_{k-1}(s', a')$$

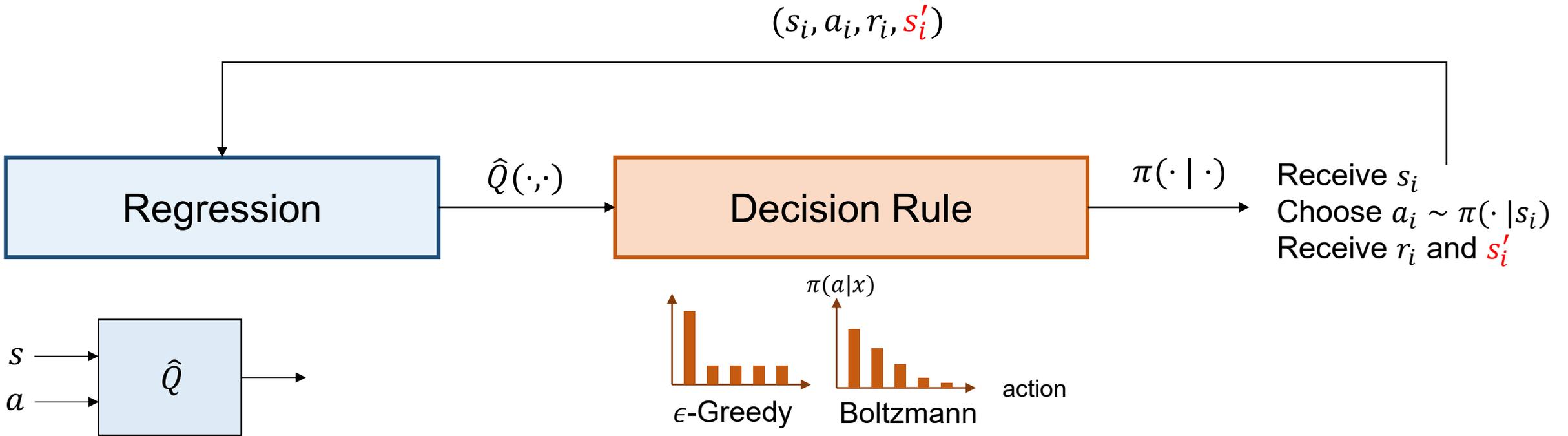
Perform one iteration
of Value Iteration

Review: Contextual Bandits with Regression

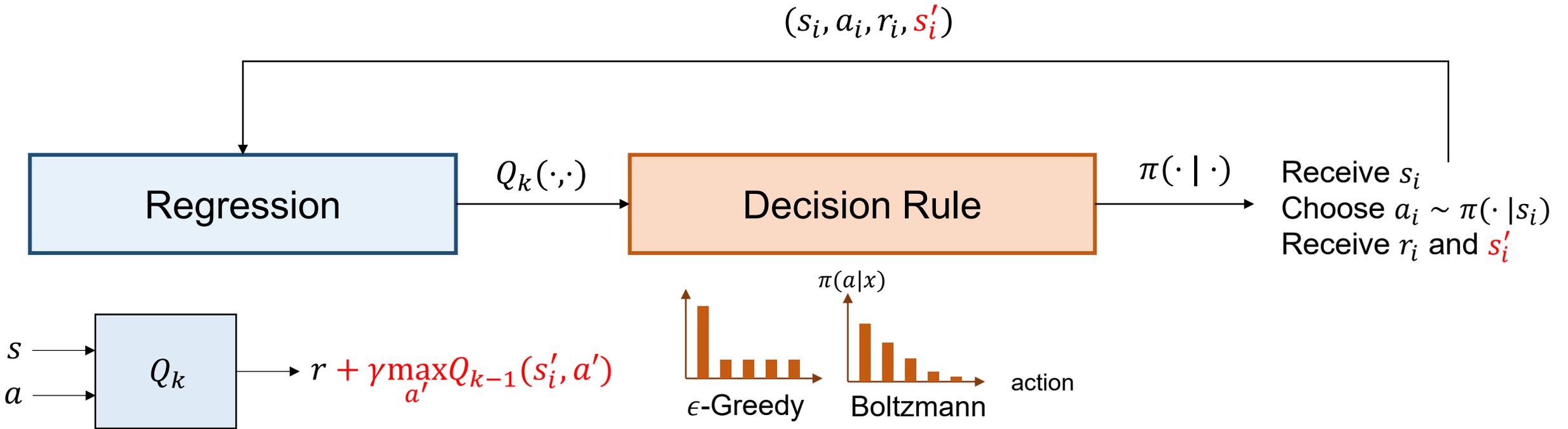


Train \hat{R} such that $\hat{R}(x_i, a_i) \approx r_i$

Value Iteration with Regression



Value Iteration with Regression



Train Q_k such that $Q_k(s_i, a_i) \approx r_i + \gamma \max_{a'} Q_{k-1}(s'_i, a')$

Value Iteration with Samples

For $k = 1, 2, \dots$

For $i = 1, 2, \dots, N$:

Choose action $a_i \sim \text{EG}(Q_{\theta_k}(s_i, \cdot))$ or BE

Receive reward $r_i \sim R(s_i, a_i)$ and $s'_i \sim P(\cdot | s_i, a_i)$

$s_{i+1} = s'_i$ if episode continues, $s_{i+1} \sim \rho$ if episode ends

$$\theta_{k+1} \leftarrow \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^N \left(Q_{\theta}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\theta_k}(s'_i, a') \right)^2$$

Data collection

Perform one iteration
of Value Iteration

Value Iteration with Samples

For $k = 1, 2, \dots$

For $i = 1, 2, \dots, N$:

Choose action $a_i \sim \text{EG}(Q_{\theta_k}(s_i, \cdot))$ or BE

Receive reward $r_i \sim R(s_i, a_i)$ and $s'_i \sim P(\cdot | s_i, a_i)$

$s_{i+1} = s'_i$ if episode continues, $s_{i+1} \sim \rho$ if episode ends

Data collection

$\theta \leftarrow \theta_k$

For $m = 1, 2, \dots, M$:

Randomly sample a batch B from $\{1, 2, \dots, N\}$

$$\theta \leftarrow \theta - \alpha \frac{1}{|B|} \sum_{(s, a, r, s') \in B} \nabla_{\theta} \left(Q_{\theta}(s, a) - r - \gamma \max_{a'} Q_{\theta_k}(s', a') \right)^2$$

Perform one iteration of Value Iteration

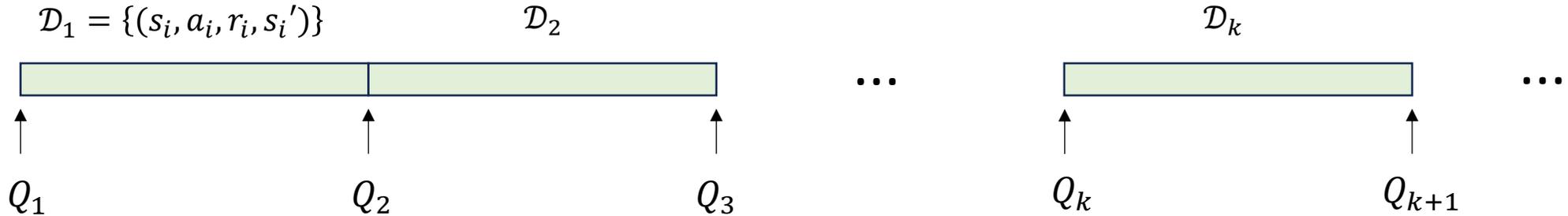
$\theta_{k+1} \leftarrow \theta$

↑
Target network

Target Network

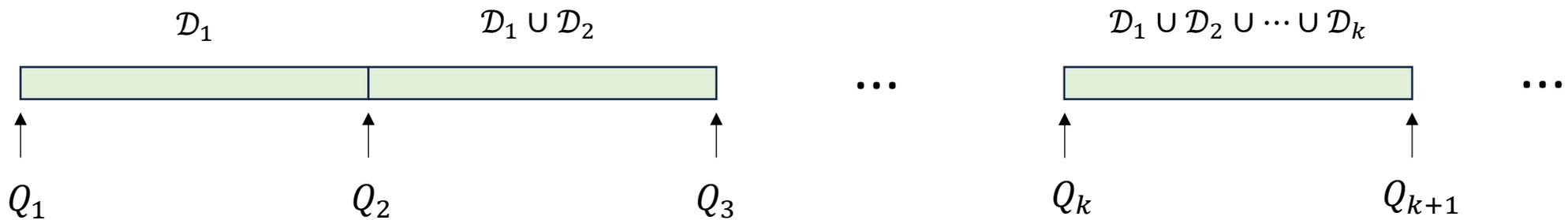
Reusing Samples

(e.g., using ϵ -greedy)



The algorithm in the previous slide only use \mathcal{D}_k to train θ_{k+1} .

However, as the reward function R and transition P remains unchanged, it is valid (actually, even better) to reuse samples:



Benefits of Reusing Samples

- Improving data efficiency
 - Every sample is used multiple times in training – just like we usually go through multiple epochs for supervised learning tasks.
- The buffer \mathcal{B} will consist of a **wider range** of state-actions
 - It allows better approximation of

$$\forall s, a, \quad Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q_k(s', a')$$

Value Iteration with Reused Samples (= Deep Q-Learning or DQN)

Initialize $\mathcal{RB} = \{\}$ ← Replay buffer

For $k = 1, 2, \dots$

For $i = 1, 2, \dots, N$:

Choose action $a_i \sim \text{EG}(Q_{\theta_k}(s_i, \cdot))$ or BE

Receive reward $r_i \sim R(s_i, a_i)$ and $s'_i \sim P(\cdot | s_i, a_i)$

$s_{i+1} = s'_i$ if episode continues, $s_{i+1} \sim \rho$ if episode ends

Push (s_i, a_i, r_i, s'_i) to \mathcal{RB}

$\theta \leftarrow \theta_k$

For $m = 1, 2, \dots, M$:

Randomly sample a batch B from \mathcal{RB}

$$\theta \leftarrow \theta - \alpha \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \nabla_{\theta} \left(Q_{\theta}(s, a) - r - \gamma \max_{a'} Q_{\theta_k}(s', a') \right)^2$$

$\theta_{k+1} \leftarrow \theta$

HW3 task

Data collection

Perform one iteration
of Value Iteration

↑
Target network

Value Iteration with Reused Samples (= Deep Q-Learning or DQN)

Initialize $\mathcal{RB} = \{\}$ ← Replay buffer

For $k = 1, 2, \dots$

For $i = 1, 2, \dots, N$:

Choose action $a_i \sim \text{EG}(Q_{\theta_k}(s_i, \cdot))$ or BE

Receive reward $r_i \sim R(s_i, a_i)$ and $s'_i \sim P(\cdot | s_i, a_i)$

$s_{i+1} = s'_i$ if episode continues, $s_{i+1} \sim \rho$ if episode ends

Push (s_i, a_i, r_i, s'_i) to \mathcal{RB}

For $m = 1, 2, \dots, M$:

Randomly sample a batch B from \mathcal{RB}

$$\theta \leftarrow \theta - \alpha \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \nabla_{\theta} \left(Q_{\theta}(s, a) - r - \gamma \max_{a'} Q_{\bar{\theta}}(s', a') \right)^2$$

$$\bar{\theta} \leftarrow (1 - \tau)\bar{\theta} + \tau\theta$$

↑
Target network

HW3 task

Data collection

Perform one iteration
of Value Iteration

Deep Q-Learning Elements

- Target Network
 - Keep the Q/V value from the previous iteration of value iteration
- Replay Buffer
 - Allow reusing samples collected from previous policies

When Does DQN Succeed (In Theory)?

DQN tries to approximate **Value Iteration** by solving

$$\theta_{k+1} = \operatorname{argmin}_{\theta} \sum_{i \in B} \left(Q_{\theta}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\theta_k}(s'_i, a') \right)^2 \quad (1)$$

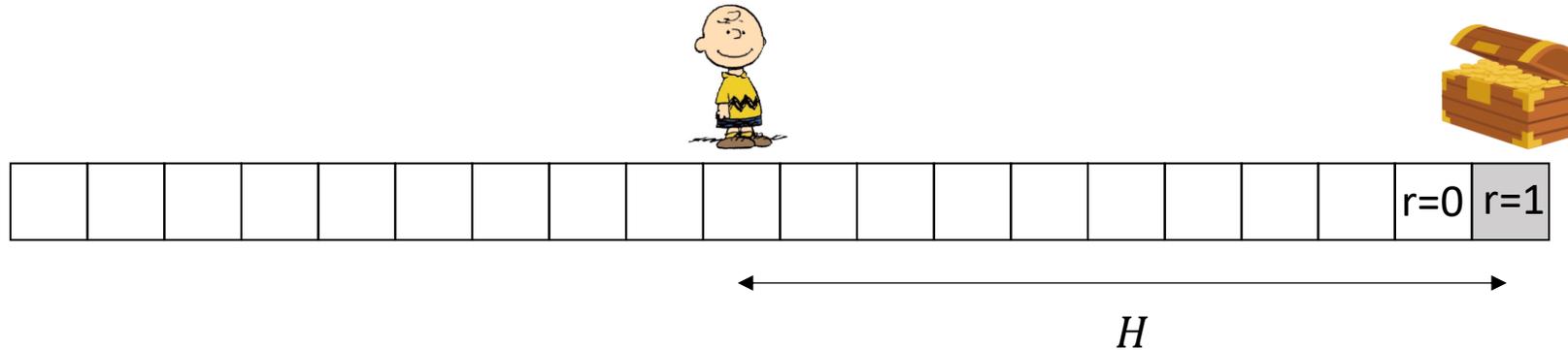
The true Value Iteration:

$$\forall \mathbf{s}, \mathbf{a}, \quad Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q_k(s', a') \quad (2)$$

Under what conditions can (1) well approximate (2)?

- B should contain a wide range of state-action pairs (a challenge of **exploration**)

Exploration in MDPs (Not Easy)



Environment:

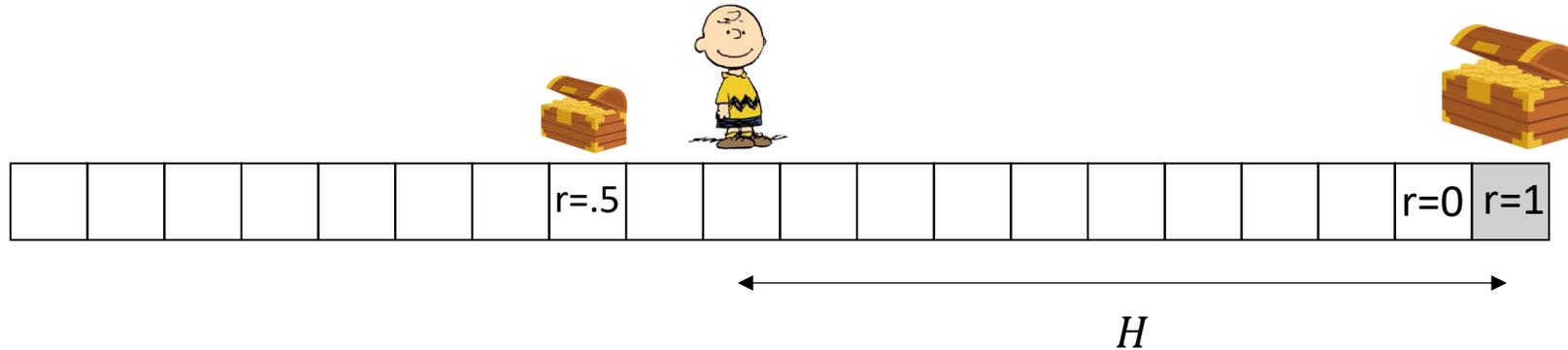
- Fixed-horizon MDP with episode length H
- Initial state at 0
- A single rewarding state at state H
- Actions: Go LEFT or RIGHT

Suppose we perform DQN with ϵ -greedy with random initialization

⇒ On average, we need 2^H episodes to see the reward

(before that, we won't make any meaningful update and will just do random walk around state 0)

Exploration in MDPs (Not Easy)



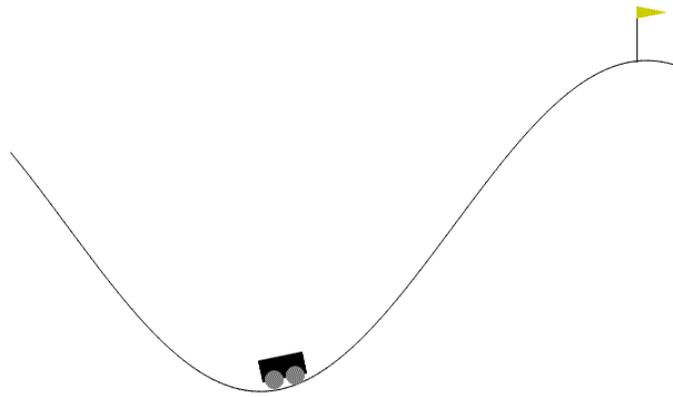
Key issue:

- The ϵ -greedy strategy (or BE) performs **action-space** exploration but not **state-space** exploration.
- This problem becomes more severe when the reward signal is **sparse** and the horizon length is **long**.
- To solve this, we usually require the **exploration bonus** (like UCB, TS), or a better **reward design**. (We will discuss them much later in the course)

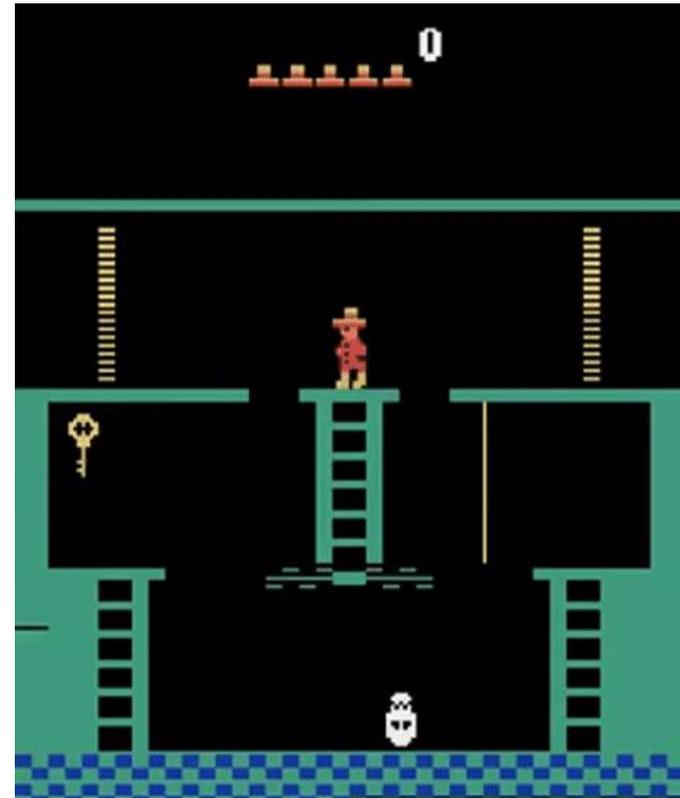
At this point (for the discussion of DQN), we pretend that EG, BE will lead to sufficient exploration over the **state space**.

Exploration in MDPs (Not Easy)

Classic sparse-reward environments:



Mountain Car



Montezuma's Revenge

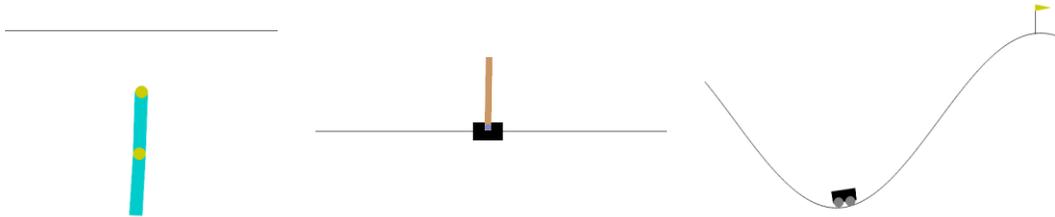
Deep Q-Learning

- DeepMind. [Playing Atari with Deep Reinforcement Learning](#). 2013.

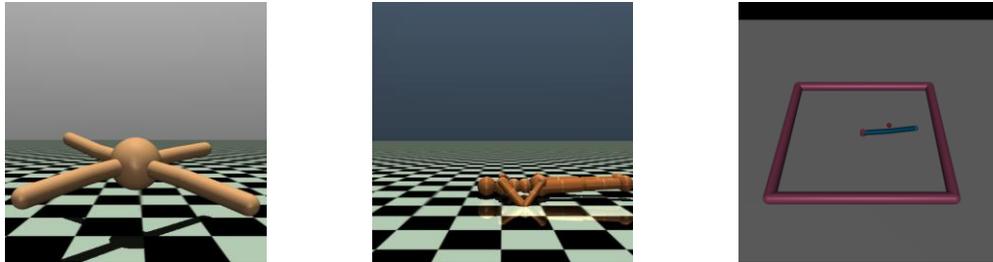


OpenAI Gym (used in HW3)

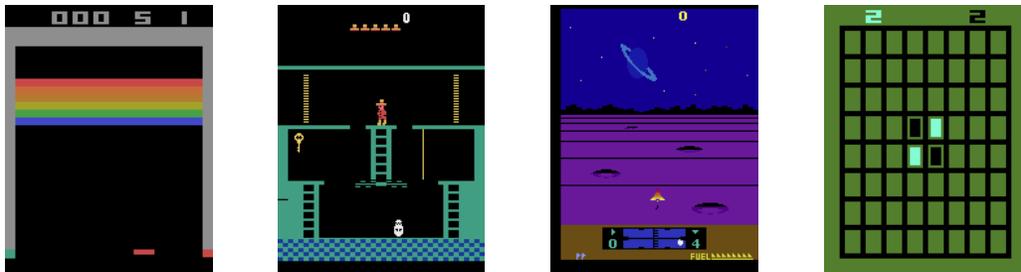
- An RL library that provides benchmark environments with a unified interface



Classic control

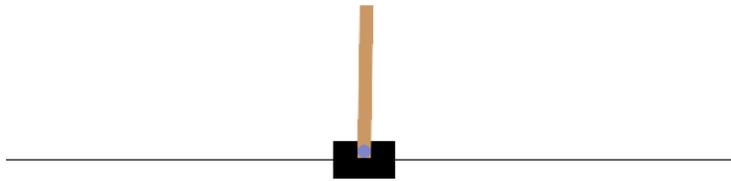


Mujoco



Atari games

HW3: DQN in CartPole



Left and Right

Action Space	Discrete(2)
Observation Space	Box([-4.8 -inf -0.41887903 -inf], [4.8 inf 0.41887903 inf], (4,), float32)

Cart Position Cart Velocity Pole Angle Pole Angle Velocity

HW3: DQN in CartPole



Handling the Non-Ideal Case

When DQN cannot well-approximate VI

In practice,

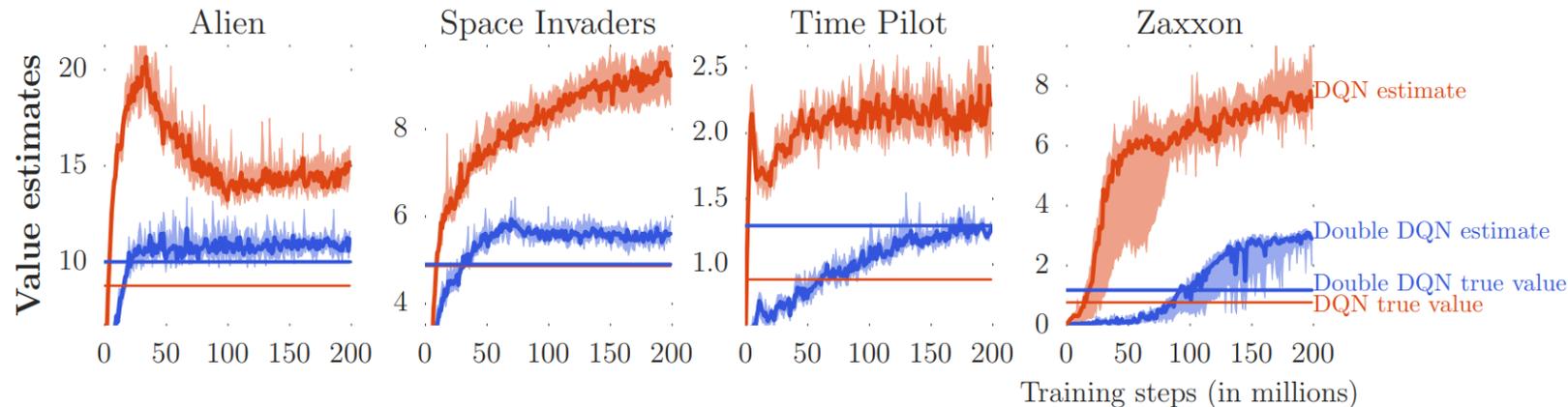
- We may not be able to collect sufficiently rich state-actions

Thus, we may not have

$$\forall s, a \quad Q_{\theta_{k+1}}(s, a) \approx R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q_{\theta_k}(s', a')$$

When DQN cannot well-approximate VI

In this case, $Q_{\theta_k}(s, a)$ tends to **overestimate** $Q^*(s, a)$, and the greedy policy $\hat{\pi}(s) = \operatorname{argmax}_a Q_{\theta_k}(s, a)$ could be very bad.



When DQN cannot well-approximate VI

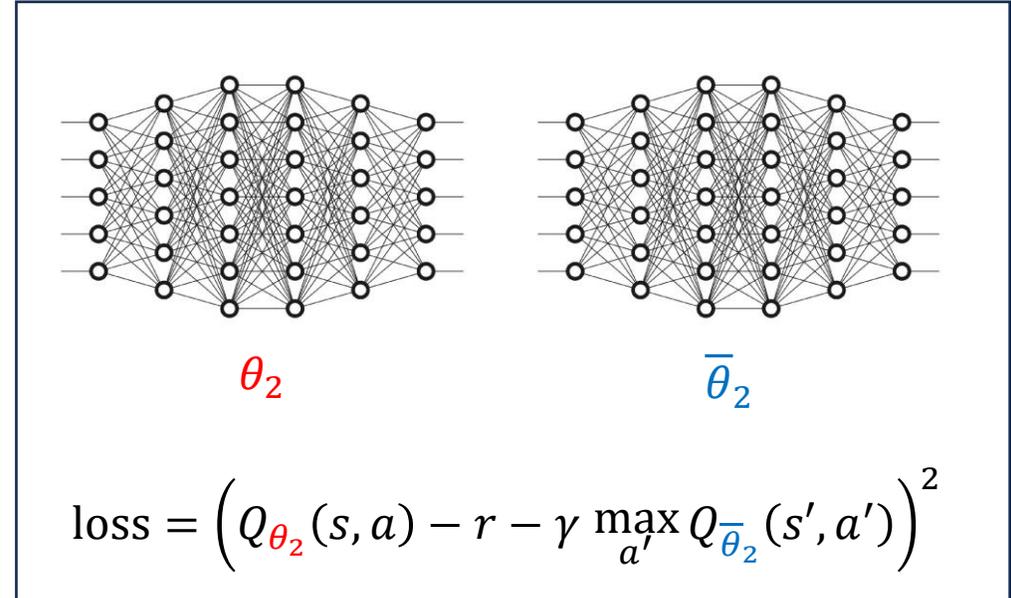
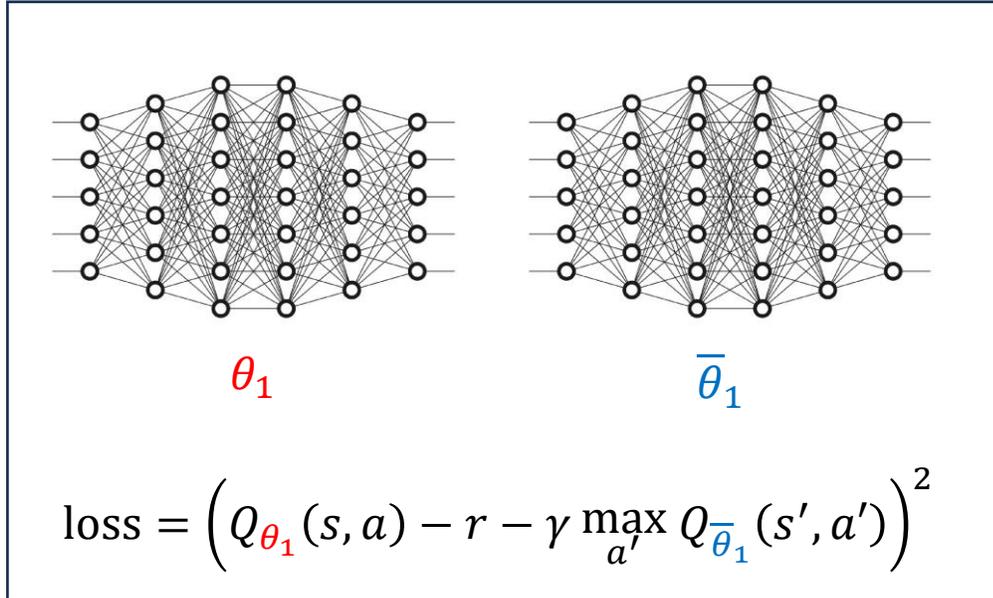
Such “seeking the error” behavior is due to “**bootstrapping**”

- “**Bootstrapping**” in DQN: perform estimation (i.e., $Q(s, a)$) using other estimation (i.e., $r + \gamma \max_{a'} Q(s', a')$)
- An issue only in MDP but not in bandits

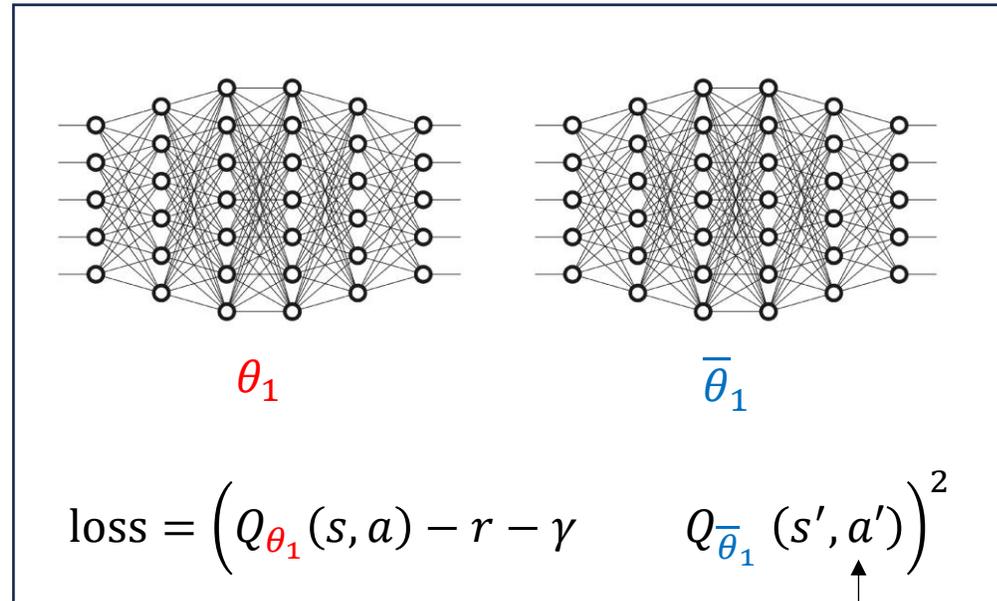
To prevent overestimation, a common technique is

- Double Q-learning: decorrelating the choice of argmax action and the error of the value function

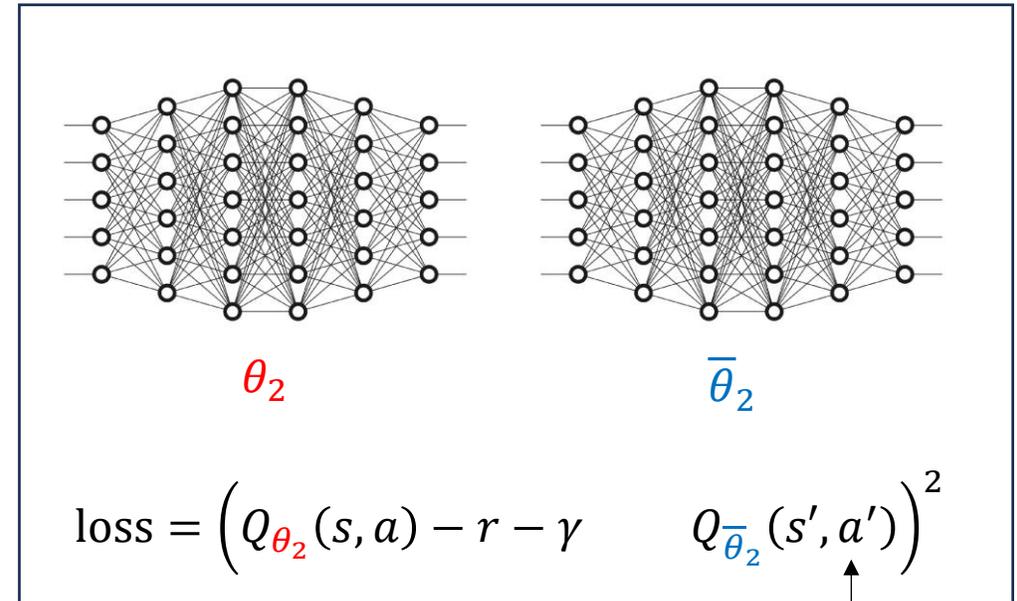
Double DQN (v1)



Double DQN (v1)

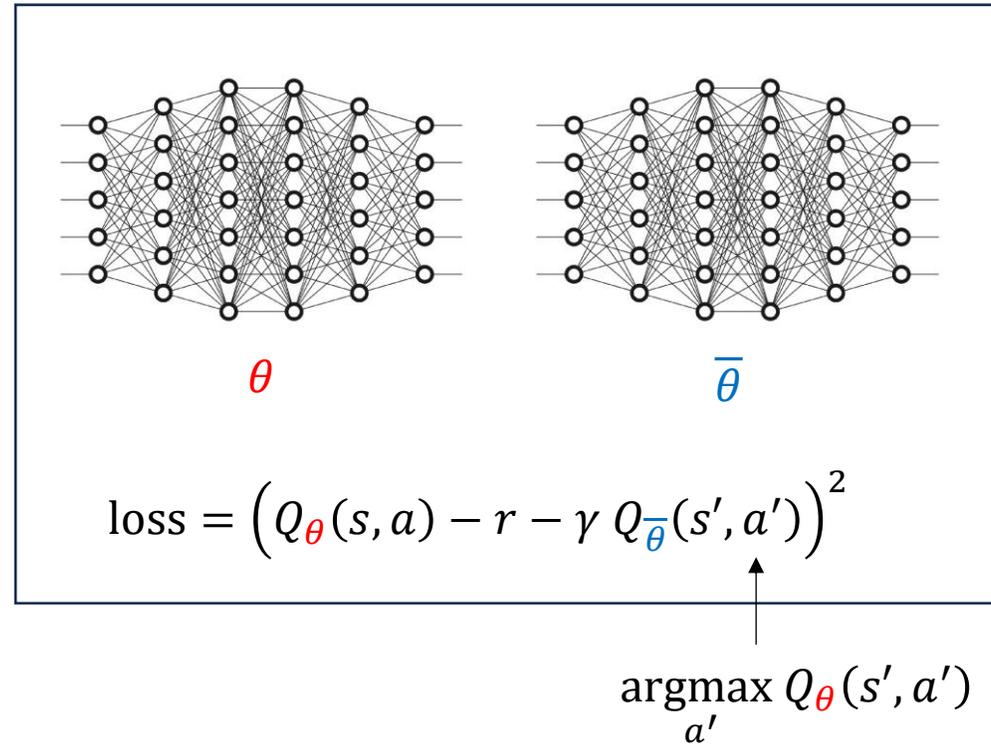


\uparrow
 $\operatorname{argmax}_{a'} Q_{\bar{\theta}_2}(s', a')$



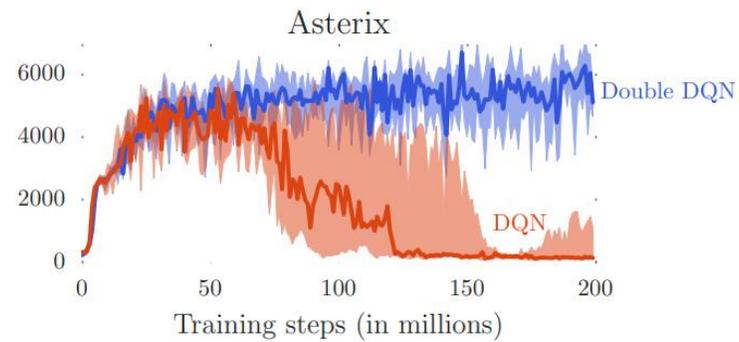
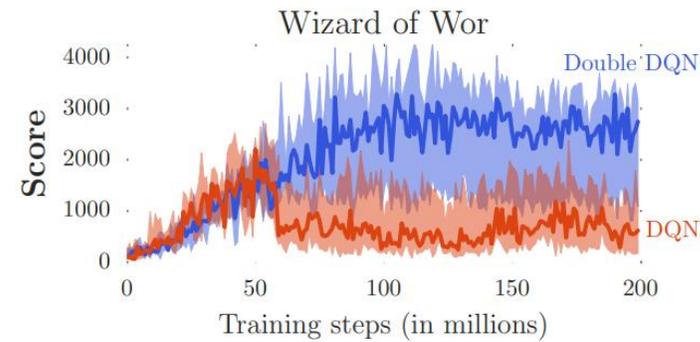
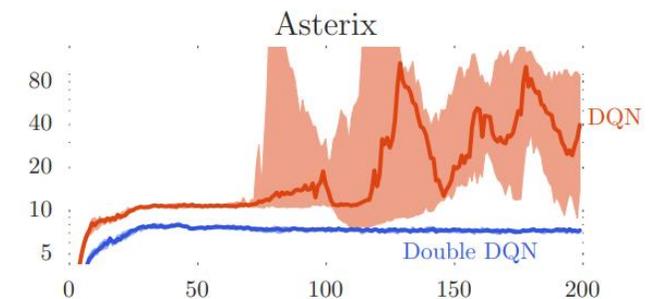
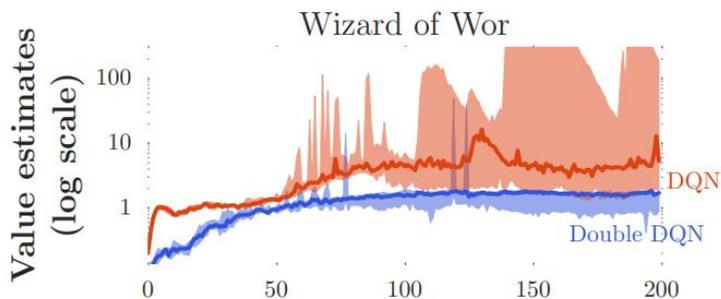
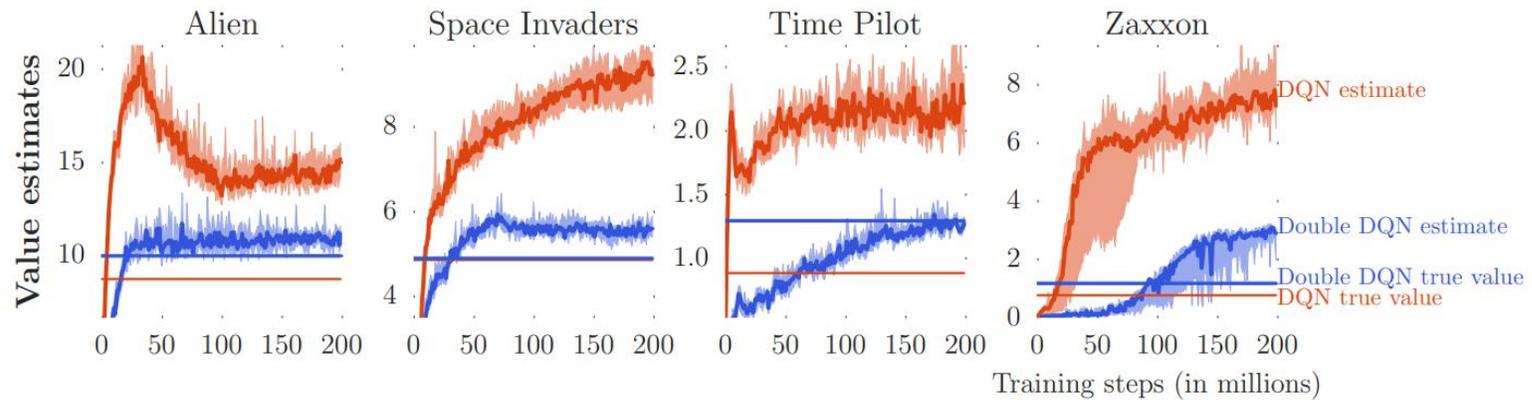
\uparrow
 $\operatorname{argmax}_{a'} Q_{\bar{\theta}_1}(s', a')$

Double DQN (v2)



DeepMind. [Deep Reinforcement Learning with Double Q-learning](#). 2015.

Double DQN



Summary for DQN

- Motivation: approximating Value Iteration using **samples** and **function approximation**
- Standard elements: target network, replay buffer
- Work better if the replay buffer has wide state-action coverage
- If not (which is usually the case), it tends to overestimate Q values and suggest arbitrary actions
- Solutions
 - Double Q-learning

Other Variants that Fail

An Unstable Variant

DQN without target network

For $k = 1, 2, \dots$

Randomly pick an i (or a mini-batch) from \mathcal{B}

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \left(Q_{\theta}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\bar{\theta}}(s'_i, a') \right)^2$$

$$\bar{\theta} \leftarrow \theta$$

cf. DQN with target network

For $k = 1, 2, \dots$

Randomly pick an i (or a mini-batch) from \mathcal{B}

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \left(Q_{\theta}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\bar{\theta}}(s'_i, a') \right)^2$$

$$\bar{\theta} \leftarrow (1 - \tau) \bar{\theta} + \tau \theta$$

For $k = 1, 2, \dots$

$$\theta \leftarrow \bar{\theta}$$

For $m = 1, \dots, M$:

Randomly pick an i (or a mini-batch) from \mathcal{B}

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \left(Q_{\theta}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\bar{\theta}}(s'_i, a') \right)^2$$

$$\bar{\theta} \leftarrow \theta$$

The Effect of Target Network

Let $KN = 100000$

For $k = 1, 2, \dots, K$

$$\theta_k \leftarrow \theta$$

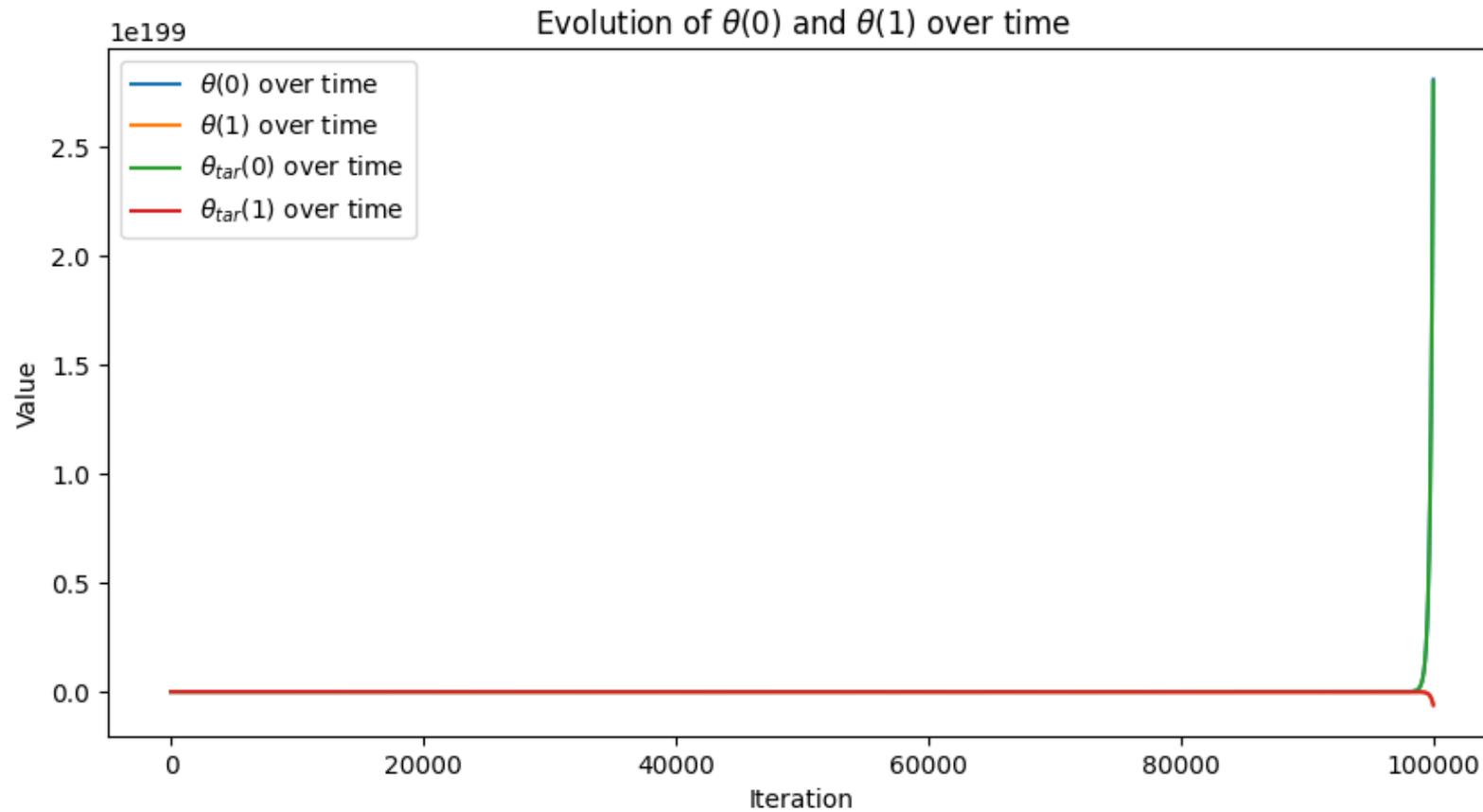
For $i = 1, \dots, N$:

Sample $(s, a, r, s') \sim \text{Uniform} \{(s_1, a, 1, s_2), (s_2, a, 0, s_2)\}$

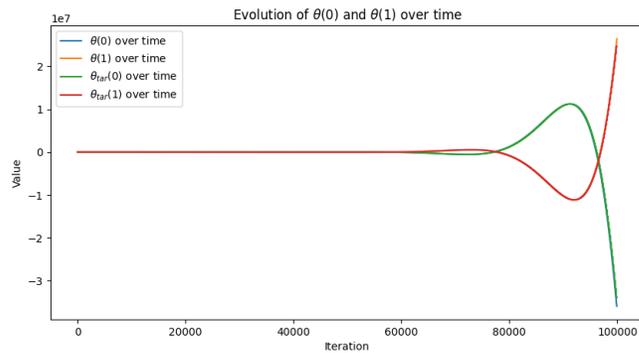
$$\theta \leftarrow \theta - \alpha \left(\phi(s, a)^\top \theta - r - \gamma \phi(s', a)^\top \theta_k \right) \phi(s, a)$$

$$\theta_{k+1} \leftarrow \theta$$

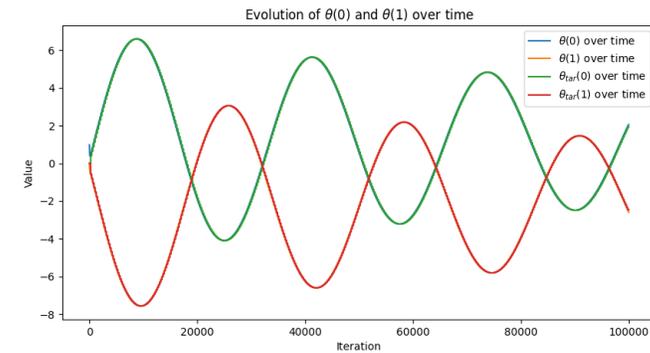
The Effect of Target Network



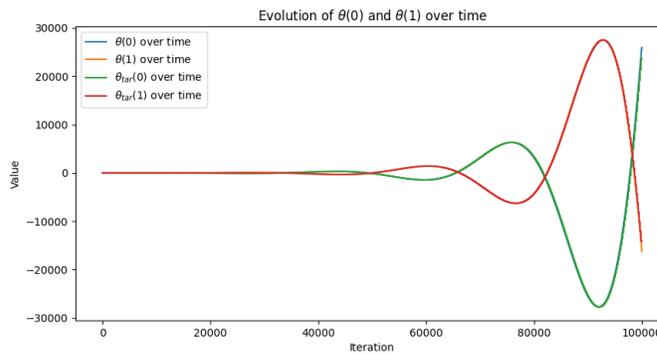
N=1



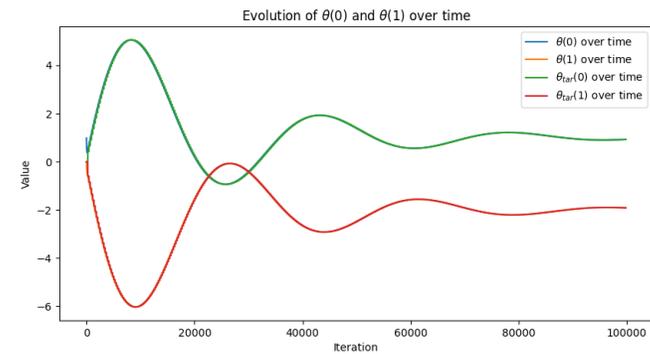
N=150



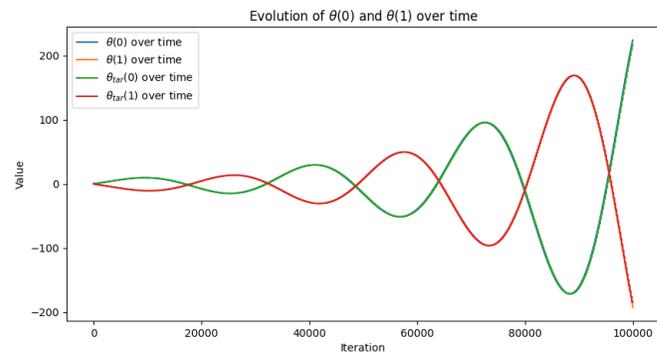
N=210



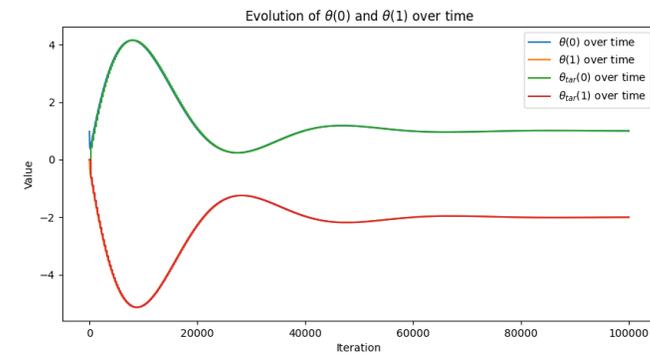
N=170



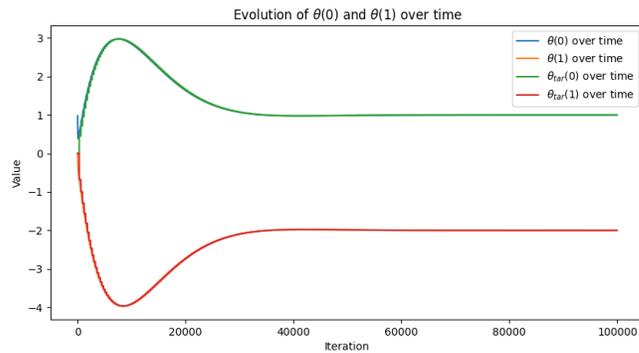
N=230



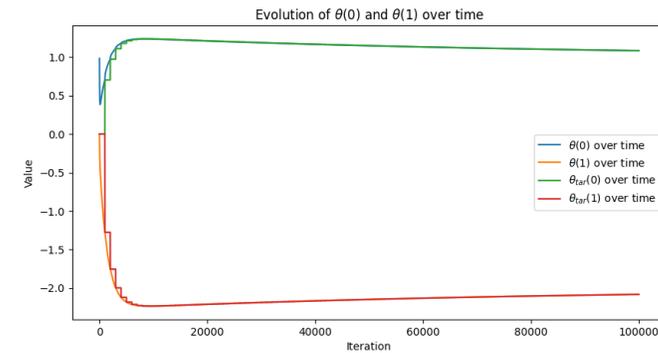
N=190



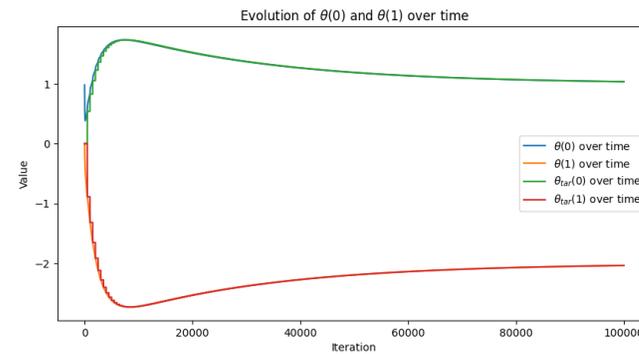
N=250



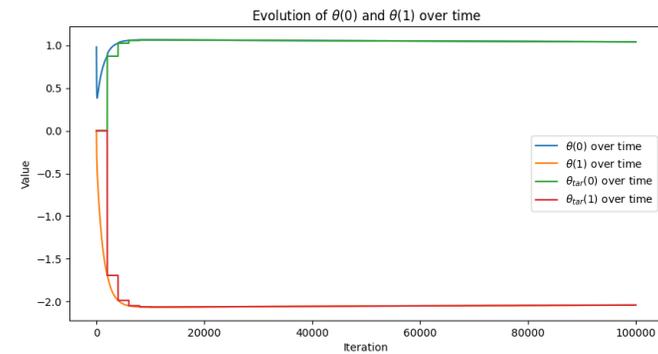
N=300



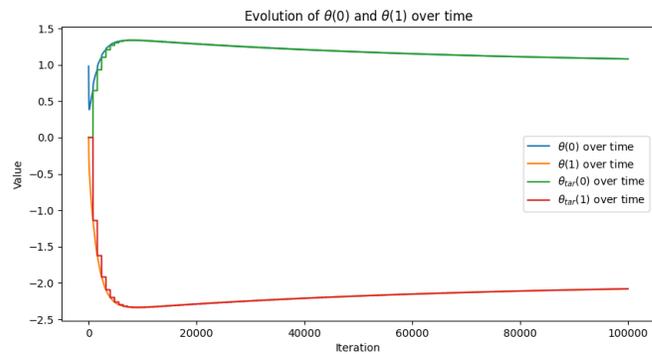
N=1000



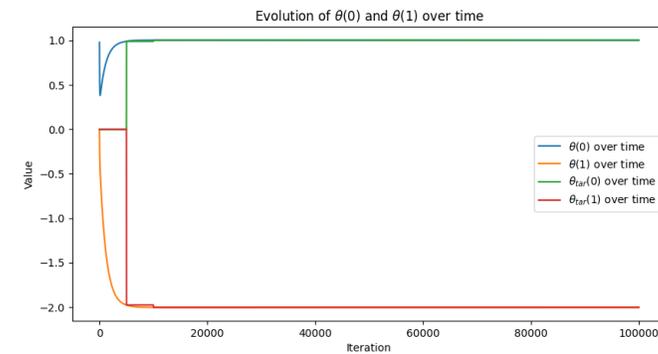
N=500



N=2000



N=800



N=5000

A Biased Variant

Residual Gradient (DQN without **stop gradient**)

For $k = 1, 2, \dots$

Randomly pick an i (or a mini-batch) from \mathcal{B}

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \left(Q_{\theta}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\theta}(s'_i, a') \right)^2$$

cf. standard DQN

For $k = 1, 2, \dots$

Randomly pick an i (or a mini-batch) from \mathcal{B}

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \left(Q_{\theta}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\bar{\theta}}(s'_i, a') \right)^2$$

$$\bar{\theta} \leftarrow (1 - \tau) \bar{\theta} + \tau \theta$$

For $k = 1, 2, \dots$

$$\theta \leftarrow \bar{\theta}$$

For $m = 1, \dots, M$:

Randomly pick an i (or a mini-batch) from \mathcal{B}

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} \left(Q_{\theta}(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\bar{\theta}}(s'_i, a') \right)^2$$

$$\bar{\theta} \leftarrow \theta$$

A Biased Variant

This variant will converge (as it is similar to standard SGD), but the solution it converges to is usually not desirable.