Approximate Value Iteration

Chen-Yu Wei

Review: Value Iteration

$V^\star(s) \coloneqq$ maximum expected total reward starting from state s

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

Bellman Equations

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

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

Value Iteration (algorithm to find $Q^\star/V^\star$)

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

For $i = 1, 2, \ldots$

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

$V_i(s) = \max_a Q_i(s,a)$ for all s

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

Review: Value Iteration

For $k = 1,\ 2, \ldots$

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

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

Value Iteration with Samples

For $k = 1,\ 2, \ldots$

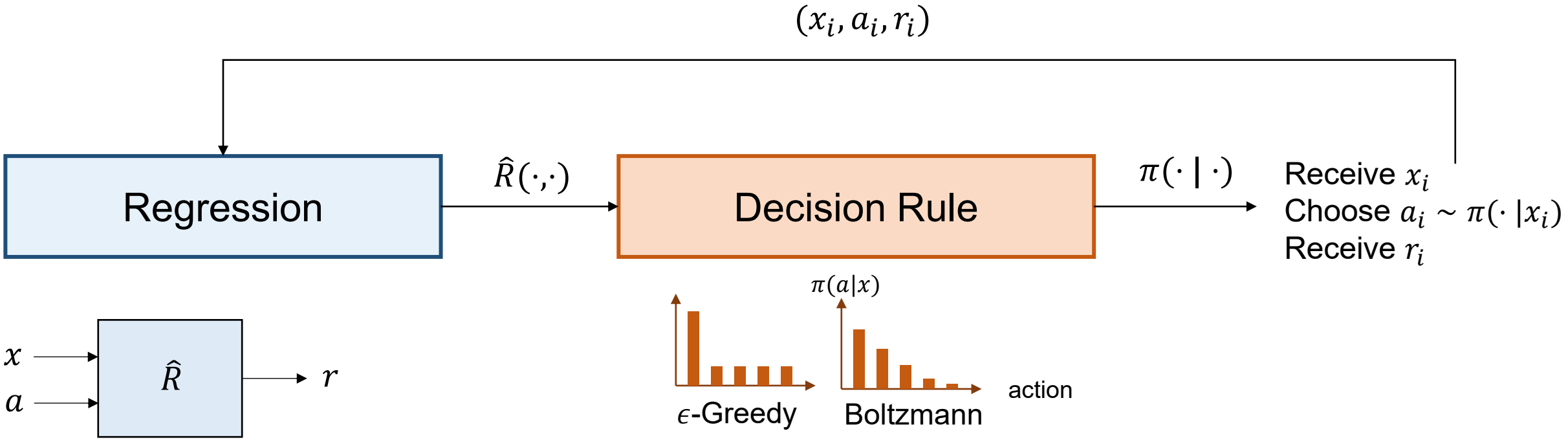
Obtain N samples $\{(s_i, a_i, r_i, s_i')\}_{i=1}^N$ where $\mathbb{E}[r_i] = R(s_i, a_i),\ s_i' \sim P(\cdot\,|s_i, a_i)$

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

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

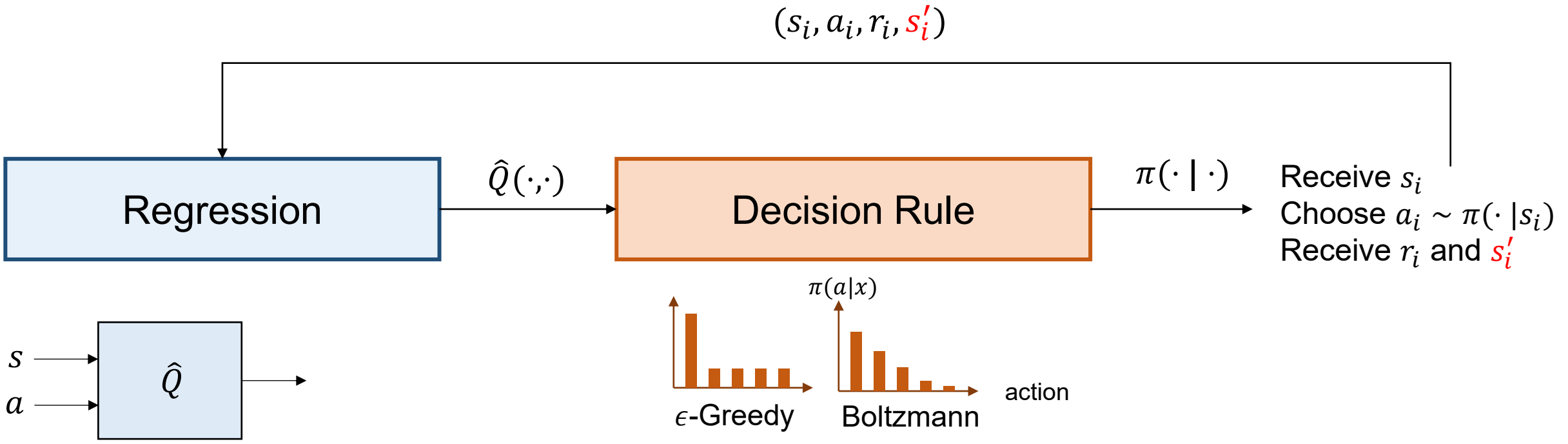
Perform one iteration of Value Iteration

Review: Contextual Bandits with Regression

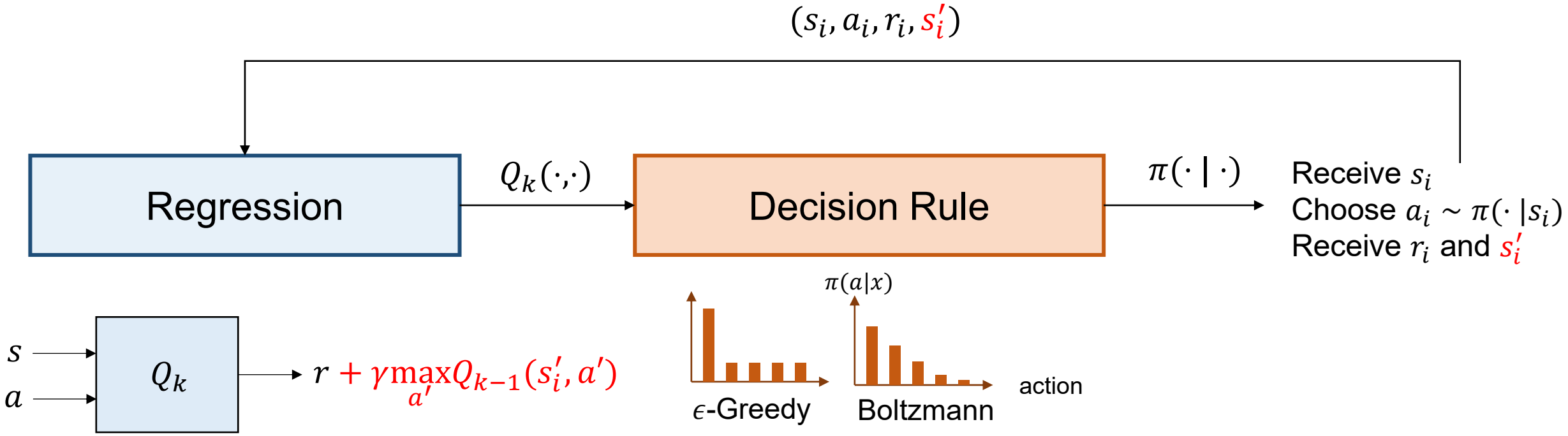


Train $\hat{R}$ such that $\hat{R}(x_i, a_i) \approx r_i$

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 $k = 1,\ 2, \ldots$

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

    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

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

Perform one iteration of Value Iteration

Value Iteration with Samples

For $k = 1,\ 2, \ldots$

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

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 $m = 1, 2, \ldots, M$:

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

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

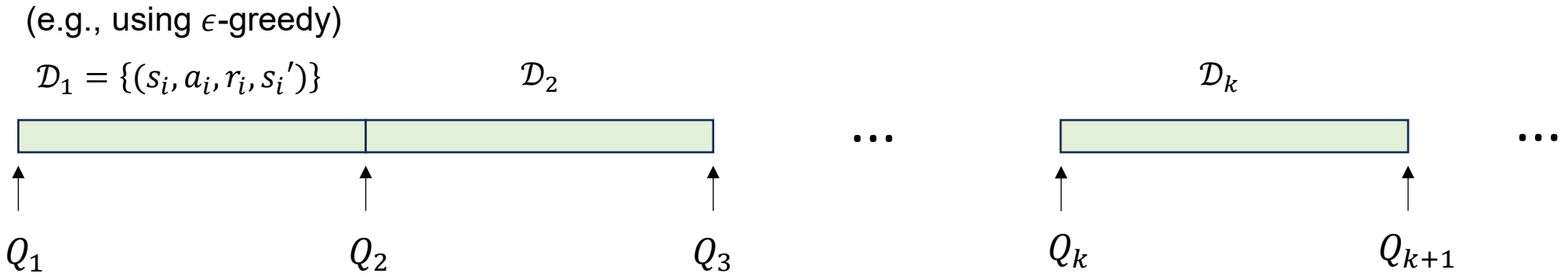
↑ Target network

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

Perform one iteration of Value Iteration

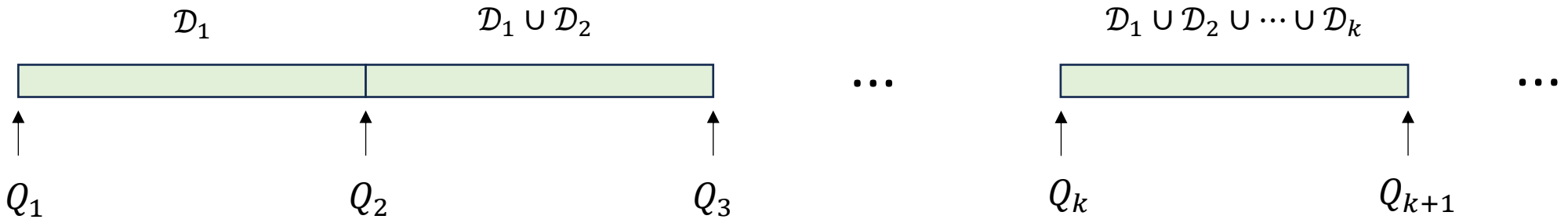
Target Network

Reusing Samples



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

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

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

HW3 task

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

For $k = 1, 2, \ldots$

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

    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 $s_i' \sim P(\cdot \mid s_i, a_i)$

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

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

Data collection

  $\theta \leftarrow \theta_k$

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

    Randomly sample a batch B from $\mathcal{RB}$

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

Target network

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

Perform one iteration of Value Iteration

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

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

HW3 task

For $k = 1, 2, \ldots$

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

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 $s_i' \sim P(\cdot \mid s_i, a_i)$

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

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

Data collection

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

Randomly sample a batch B from $\mathcal{RB}$

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

↑ Target network

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

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

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

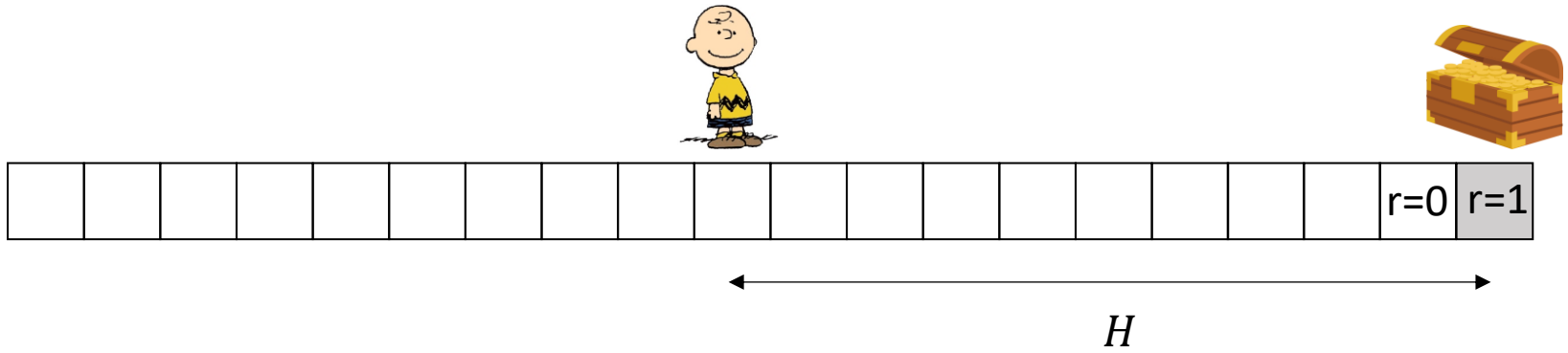
The true Value Iteration:

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

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)

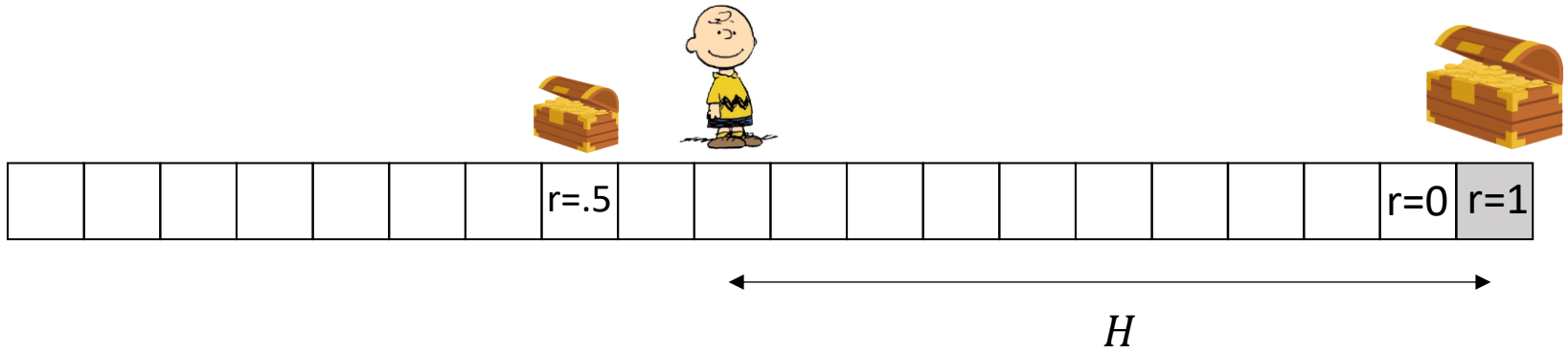


Environment:

- 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)



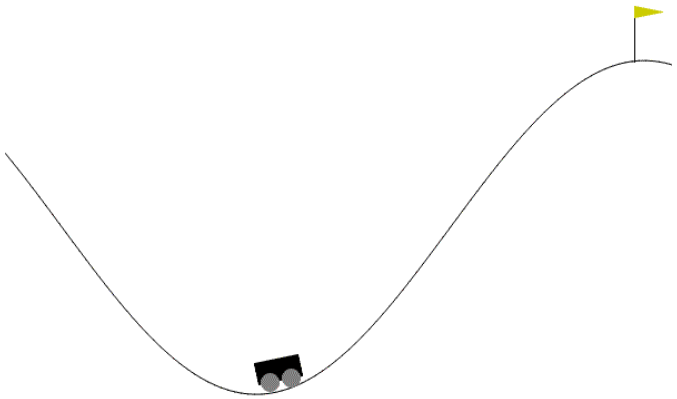
Key issue:

- 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

Handling the Non-Ideal Case

When DQN cannot well-approximate VI

In practice,

- We may not be able to collect sufficiently wide range of 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** $Q^\star(s,a)$, and the greedy policy $\hat{\pi}(s) = \underset{a}{\text{argmax}}\, Q_{\theta_k}(s,a)$ could be very bad.

When DQN cannot well-approximate VI

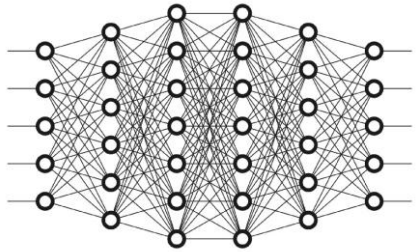
Such “seeking the error” behavior is due to “**bootstrapping**”

- An issue only in MDP but not in bandits

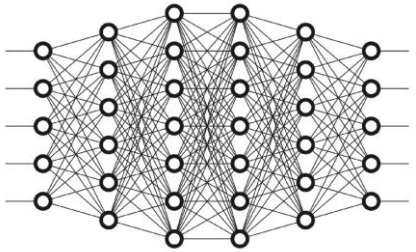
To prevent overestimation, two strategies are

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

Double DQN (v1)

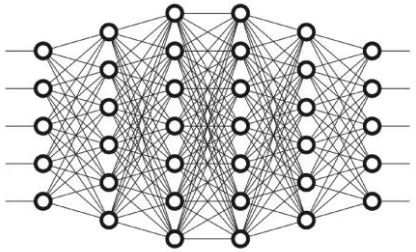


θ_1

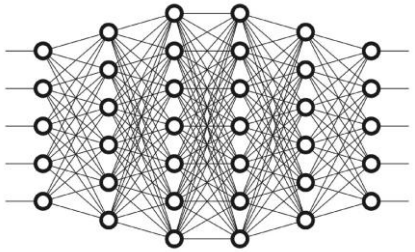


$\overline{\theta}_1$

$$\text{loss} = \left(Q_{\theta_1}(s,a) - r - \gamma \max_{a'} Q_{\overline{\theta}_1}(s',a')\right)^2$$



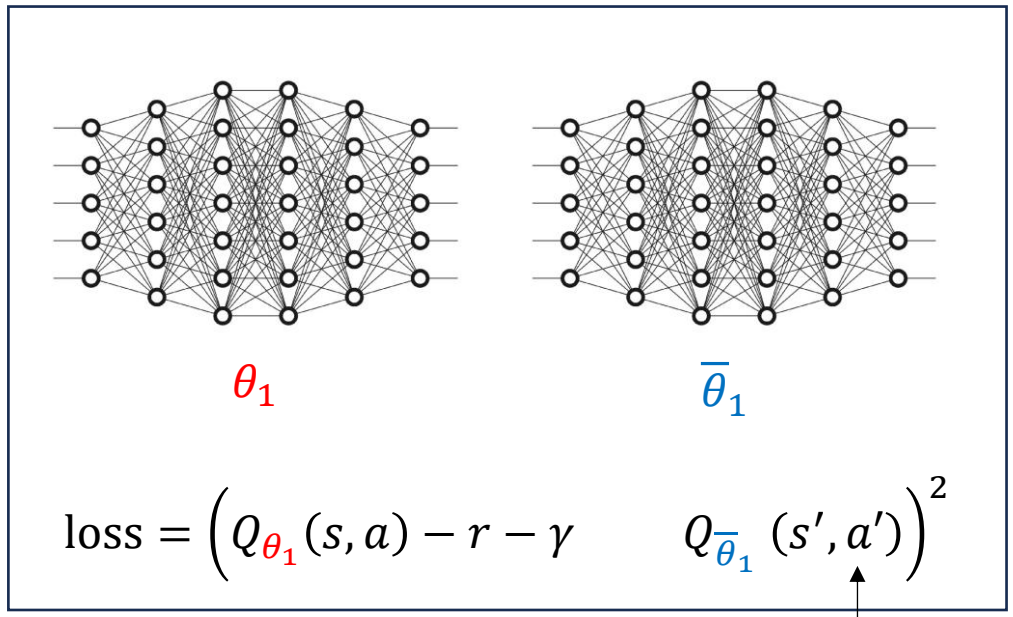
θ_2

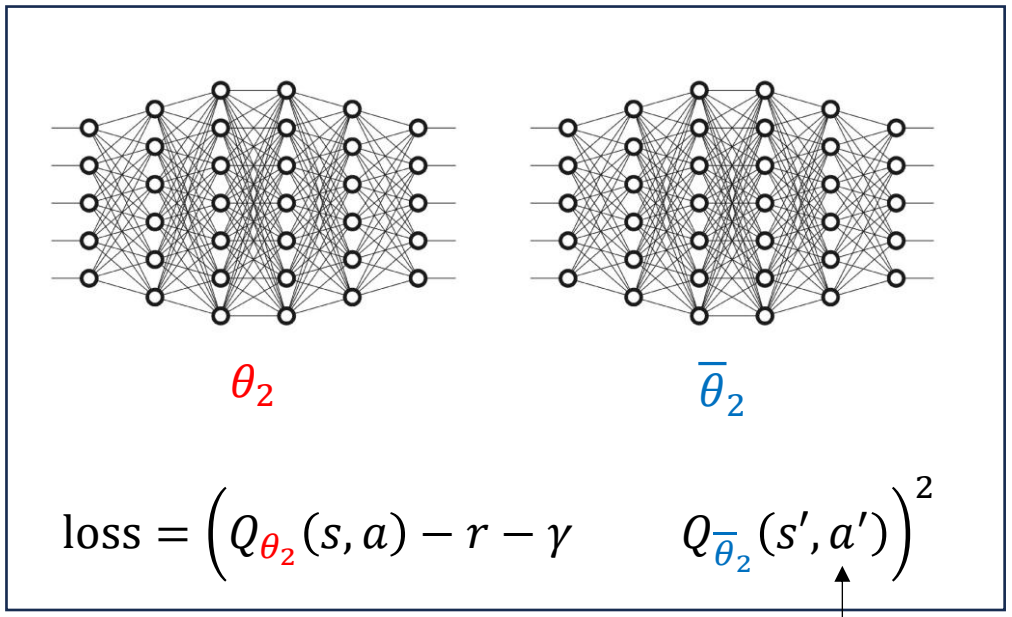


$\overline{\theta}_2$

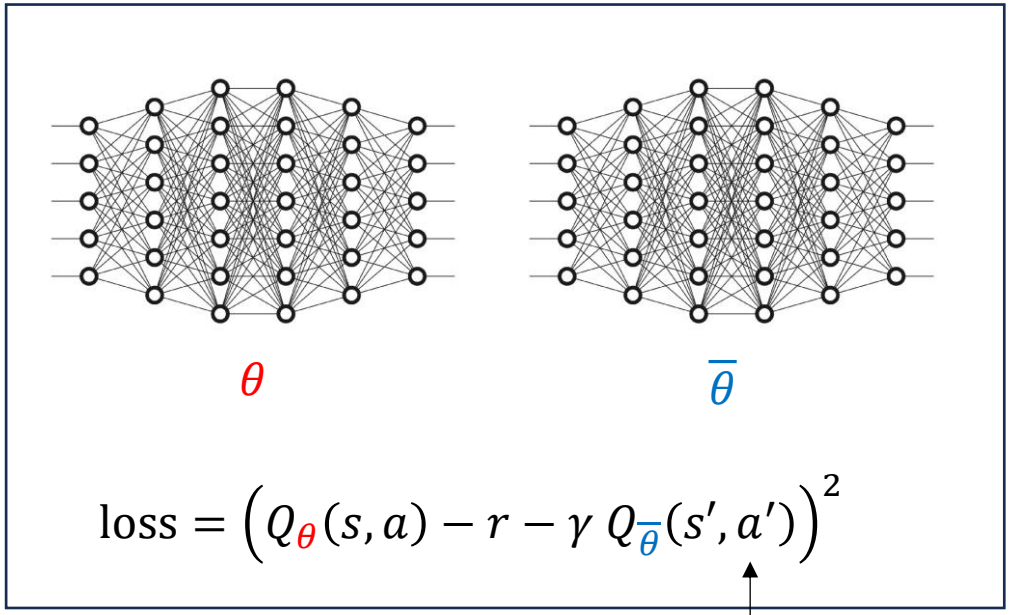
$$\text{loss} = \left(Q_{\theta_2}(s,a) - r - \gamma \max_{a'} Q_{\overline{\theta}_2}(s',a')\right)^2$$

Double DQN (v1)





Double DQN (v2)



Summary for DQN

- Motivation: approximating Value Iteration using **samples** and **function approximation**
- Standard elements: target network, replay buffer
- Work as desired when both of the following conditions hold:
  - The learner is able to obtain exploratory data (online or offline)
  - Neural network is sufficiently expressive: Bellman completeness
- When the conditions above do not hold
  - 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 $k = 1,\ 2, \ldots$

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

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

$$\overline{\theta} \leftarrow \theta$$

cf. DQN with target network

For $k = 1,\ 2, \ldots$

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

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

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

For $k = 1,\ 2, \ldots$

$\theta \leftarrow \overline{\theta}$

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

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

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

$$\overline{\theta} \leftarrow \theta$$

The Effect of Target Network

Let $KN = 100000$

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

$\theta_k \leftarrow \theta$

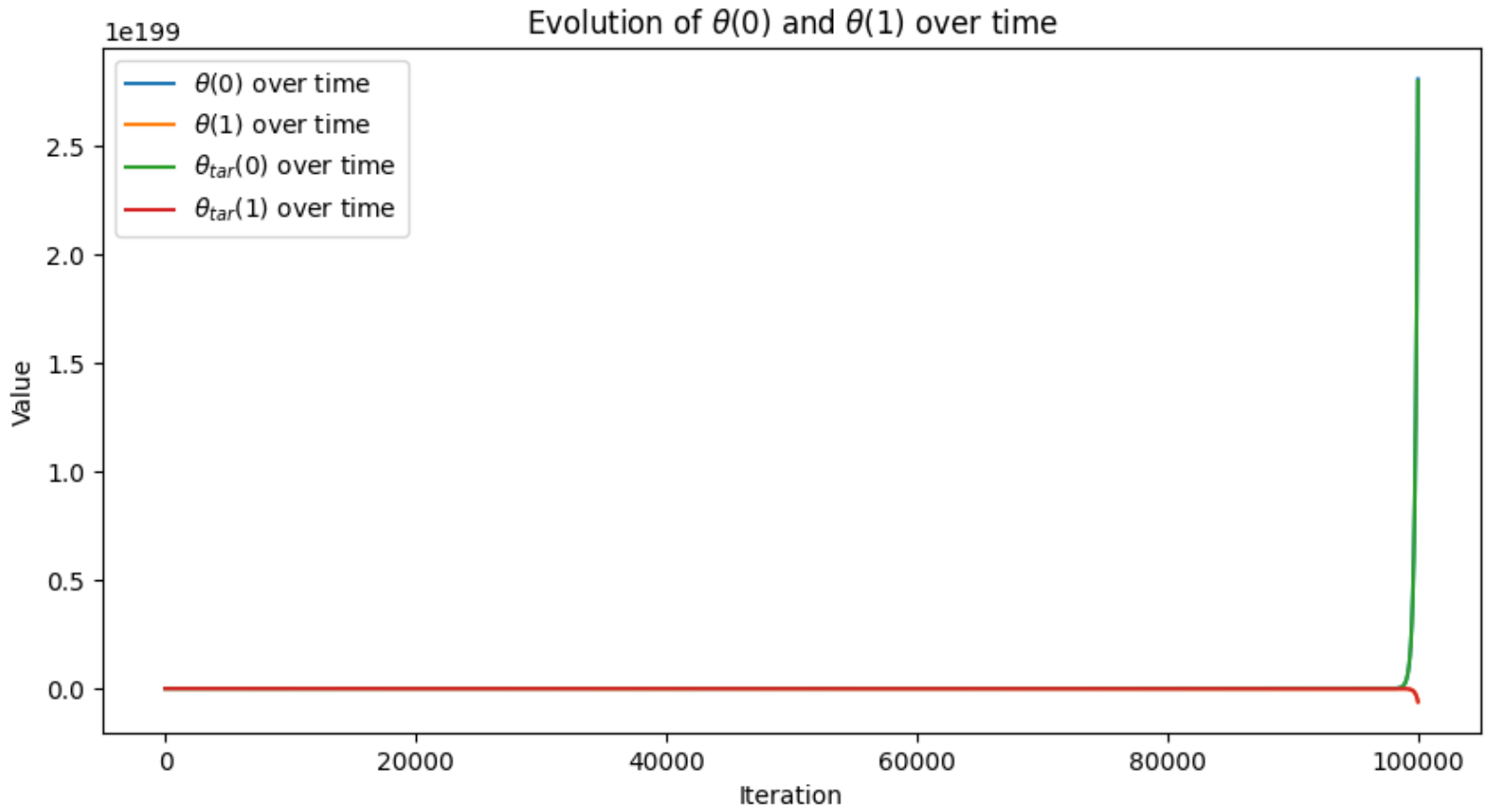
For $i = 1, \ldots, N$:

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

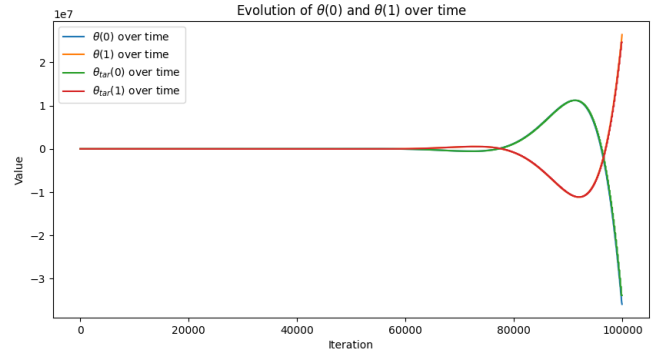
$$\theta \leftarrow \theta - \alpha \left(\phi(s,a)^\top \theta - r - \gamma \phi\left(s', a\right)^\top \theta_k \right) \phi(s,a)$$

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

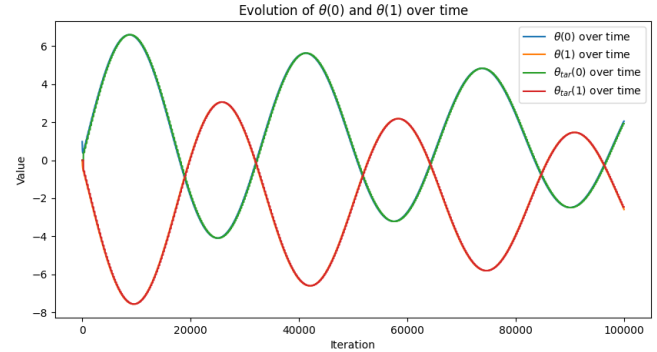
The Effect of Target Network



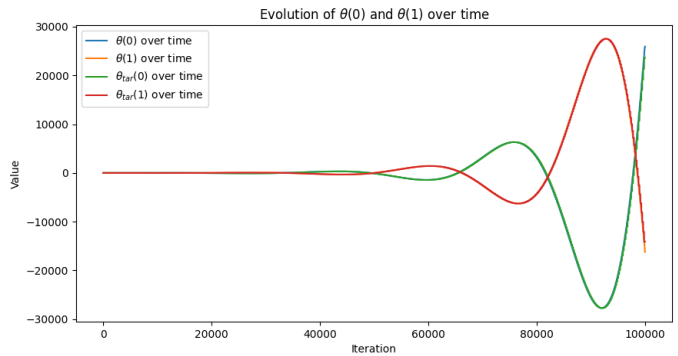
N=1



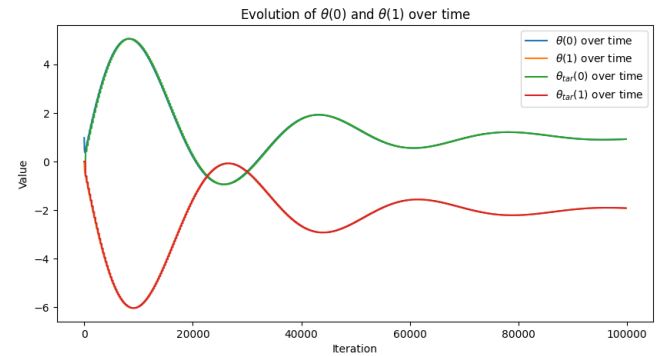
N=150



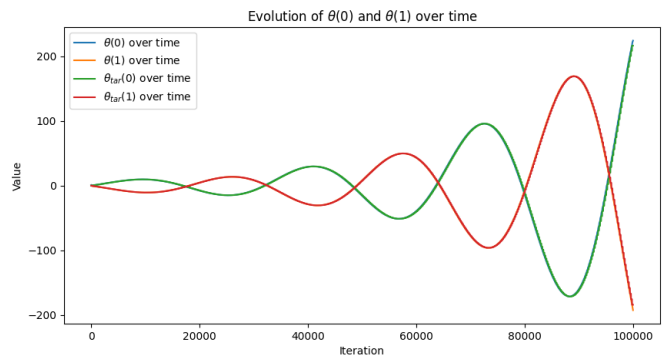
N=210



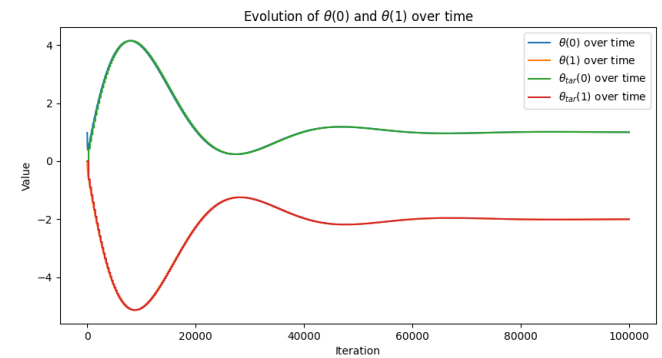
N=170



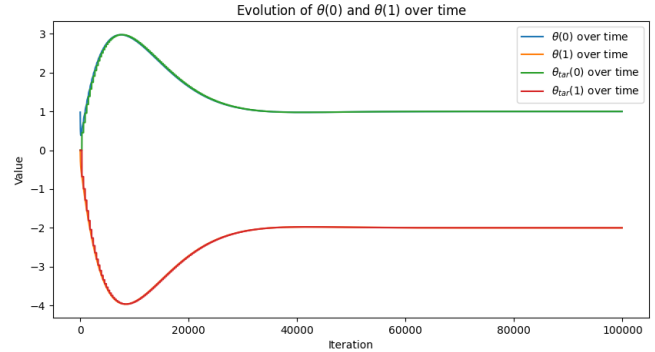
N=230



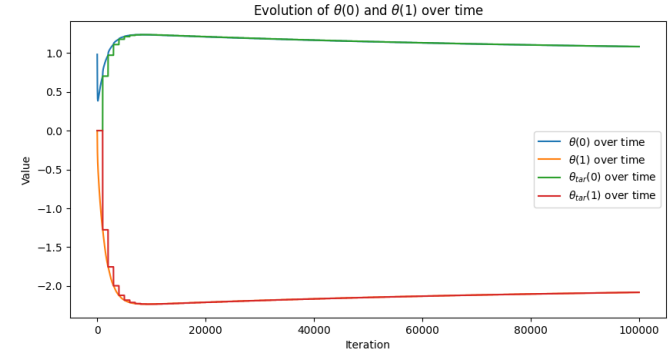
N=190



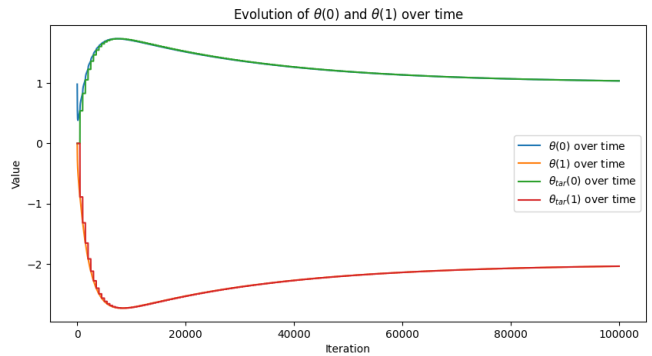
N=250



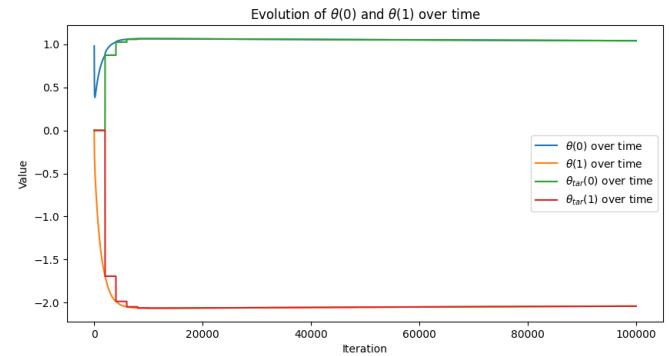
N=300



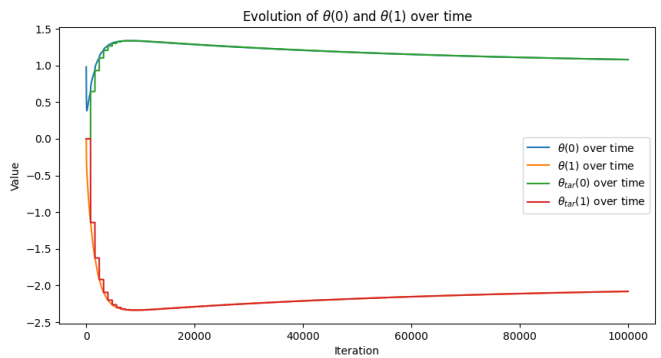
N=1000



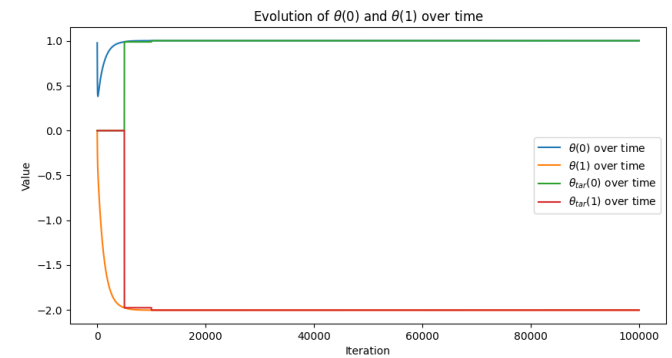
N=500



N=2000



N=800



N=5000

A Biased Variant

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

For $k = 1,\ 2, \ldots$

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 $k = 1,\ 2, \ldots$

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

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

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

For $k = 1,\ 2, \ldots$

$\theta \leftarrow \overline{\theta}$

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

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

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

$\overline{\theta} \leftarrow \theta$

A Biased Variant

This variant will converge (as it is similar to standard SGD), but the solution it converges to could be undesirable.

The underlying loss function of the Residual Gradient algorithm is

$$\sum_{i \in \mathcal{B}} \left(Q_\theta(s_i, a_i) - r_i - \gamma \max_{a'} Q_\theta(s_i', a') \right)^2$$