Approximate Value Iteration and Variants

Chen-Yu Wei

Value Iteration

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

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

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

Value Iteration with Samples

$s_1, a_1, r_1, s_2, a_2, r_2, \ldots\ s_H, a_H, r_H, s_{H+1}$

For $k = 1, 2, \ldots$

$(s_1, a_1, r_1, s_2)\ (s_2, a_2, r_2, s_3), \ldots\ (s_H, a_H, r_H, s_{H+1})\ (s, \overset{\sim P}{} \ldots)$

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

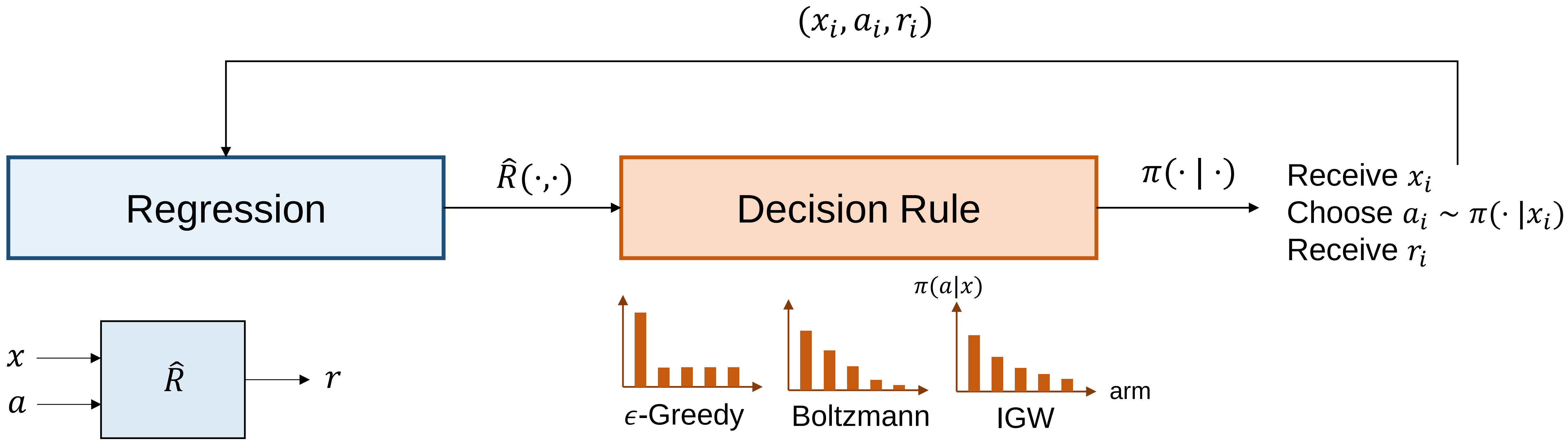
$Q_\theta(s,a)$
↑
parameter

Find θ_k that minimize

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

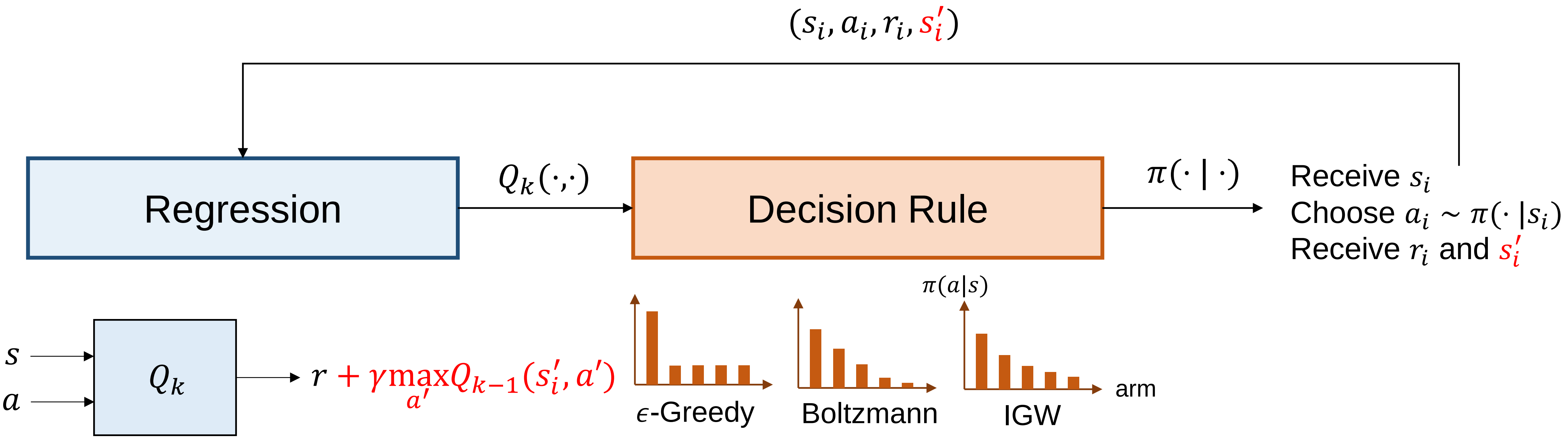
$$\mathbb{E}\left[\sim \middle| s_i, a_i \right] = R(s_i, a_i) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q_{\theta_{k-1}}(s', a')$$

Recall: Contextual Bandits with Regression



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

Value Iteration with Regression



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

This is just one iteration of Value Iteration

Value Iteration with Samples

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

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

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

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

$\theta \leftarrow \theta_k$

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

Randomly pick an i (or a mini-batch) from $\{1, 2, \ldots, N\}$

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

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

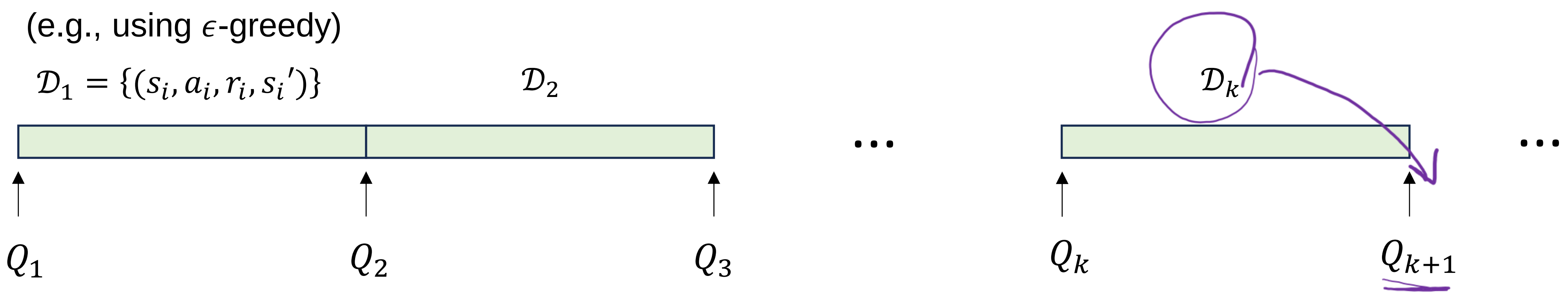
↑
Target network

Data collection

Perform one iteration of Value Iteration

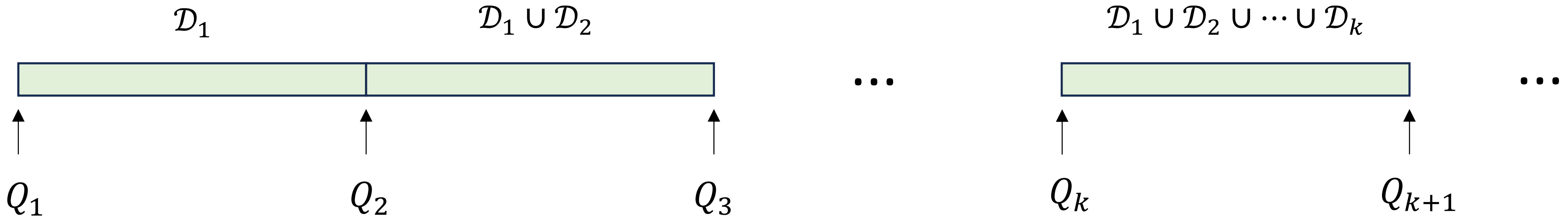
2nd for-loop: trying to find $\theta_{k+1} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(Q_\theta(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\theta_k}(s_i', a') \right)^2$

It is Valid to Reuse 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)

HW4 task

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

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

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

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

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

Insert (s_i, a_i, r_i, s_i') to $\mathcal{B}$

Data collection

$\theta \leftarrow \theta_k$

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

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

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

↑ Target network

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

Perform one iteration of Value Iteration

Another Popular Implementation

HW4 task

Initialize $\mathcal{B} = \{\}$ $\leftarrow$ Replay buffer

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

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

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

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

Insert (s_i, a_i, r_i, s_i') to $\mathcal{B}$

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

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

$$\theta \leftarrow \theta - \nabla_\theta \left(Q_\theta(s_i, a_i) - r_i - \gamma \max_{a'} \underset{\substack{\uparrow \\ \text{Target network}}}{Q_{\overline{\theta}}(s_i', a')} \right)^2$$

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

When Does DQN Succeed?

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

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

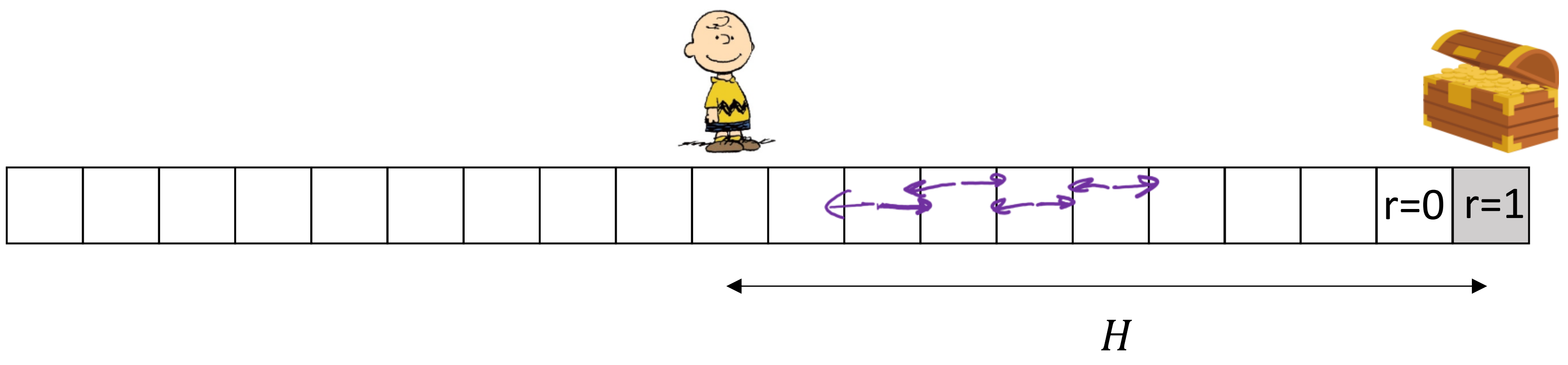
The true Value Iteration:

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

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

- $\mathcal{B}$ should contain a wide range of state-action pairs (a challenge of **exploration**)
- $Q_{\theta_{k+1}}(s,a)$ should recover $R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q_{\theta_k}(s', a')$ well for all state-actions (a challenge of **function approximation,** or **generalization**)

1. Exploration in MDPs (Not Easy)

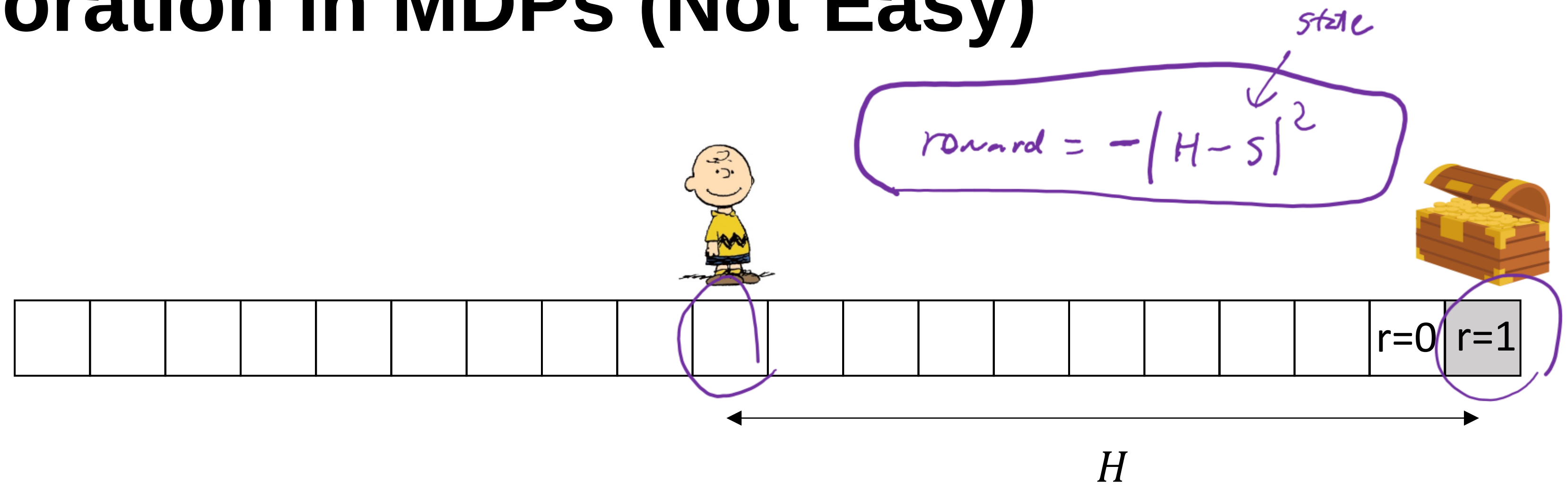


Environment:

- 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)

1. Exploration in MDPs (Not Easy)



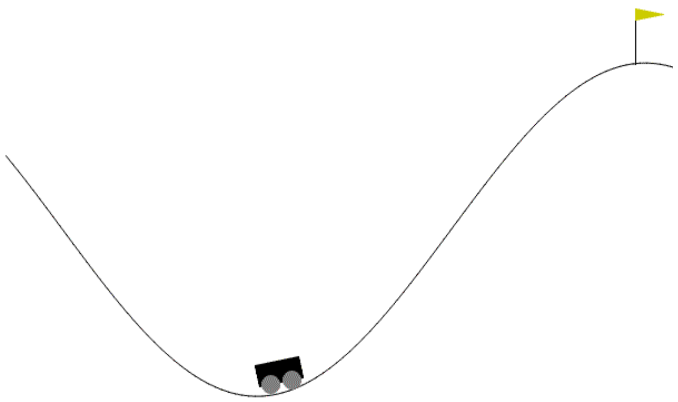
Key issue:

- The ϵ-greedy strategy (or BE, IGW) performs **action-space** exploration but not **state-space** exploration.
- This problem becomes more severe when the reward signal is **sparse**.
- To solve this, we usually require the exploration bonus (a form of reward shaping) technique – will be covered much later.

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

1. Exploration in MDPs (Not Easy)

Classic sparse-reward environments:



Mountain Car

Montezuma's Revenge

2. Function Approximation

To make DQN well approximate VI, we need

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

(ϵ-approximate) Bellman Completeness
an assumption both on the MDP and the function expressiveness

$$\forall \theta', \exists \theta \qquad \forall s, a, \qquad \left| Q_{\theta}(s,a) - \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q_{\theta'}(s',a') \right) \right| \leq \epsilon$$

This allows us to quantify the regression error in each iteration.

2. Function Approximation

In HW1 you have shown

ϵ-Greedy ensures

$$\text{Regret} \lesssim \epsilon T + \sqrt{\frac{AT \cdot \text{Err}}{\epsilon}}$$

Regression error

$$\text{Err} = \sum_{t=1}^{T} \left(\hat{R}_t(x_t, a_t) - R(x_t, a_t) \right)^2$$

In value-based contextual bandits, the requirement / assumption for function approximation is

$$\exists \theta \quad \forall x, a \qquad R_\theta(x, a) \approx R(x, a)$$

In value-based MDPs, the requirement / assumption for function approximation is

$$\forall \theta', \exists \theta \quad \forall s, a \qquad Q_\theta(s, a) \approx R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q_{\theta'}(s', a')$$

BC assumption

Analysis of DQN assuming sufficient exploration and Bellman Completeness

Recall the analysis for the exact Value Iteration:

1. Value Iteration will terminate.

$$|Q_k(s,a) - Q_{k-1}(s,a)| \leq \epsilon \quad \forall s, a$$

$$\max_{s,a} |Q_k(s,a) - Q_{k-1}(s,a)| \leq \gamma \max_{s,a} |Q_{k-1}(s,a) - Q_{k-2}(s,a)|$$

2. When it terminates, it holds that

$$|Q_k(s,a) - Q^\star(s,a)| \leq \frac{\epsilon}{1-\gamma} \quad \forall s, a$$

ValueError $\leq \frac{1}{1-\gamma}$ BellmanError

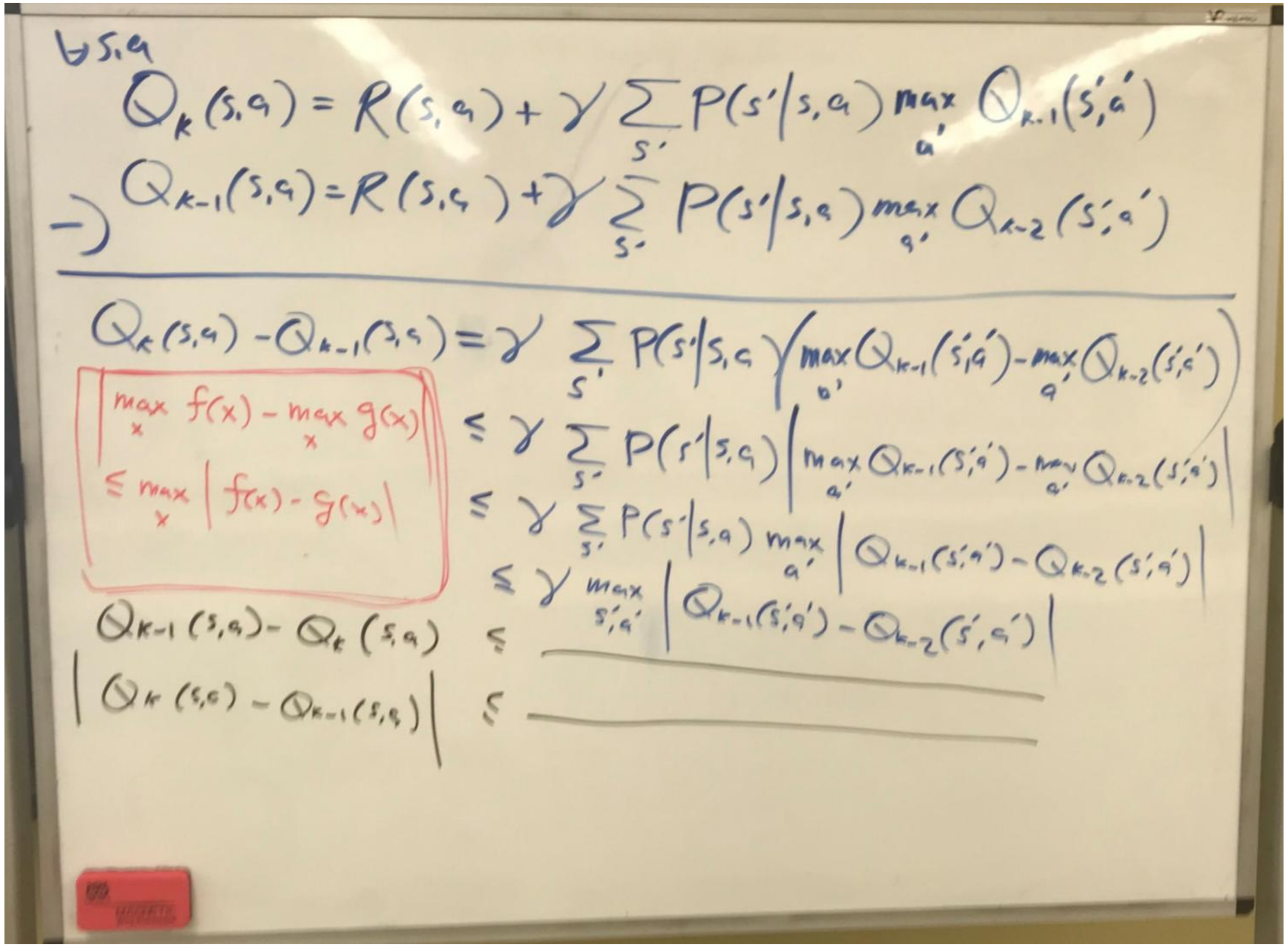
3. When it terminates, it holds that

$$V^\star(s) - V^{\hat{\pi}}(s) \leq \frac{2\epsilon}{(1-\gamma)^2} \quad \forall s$$

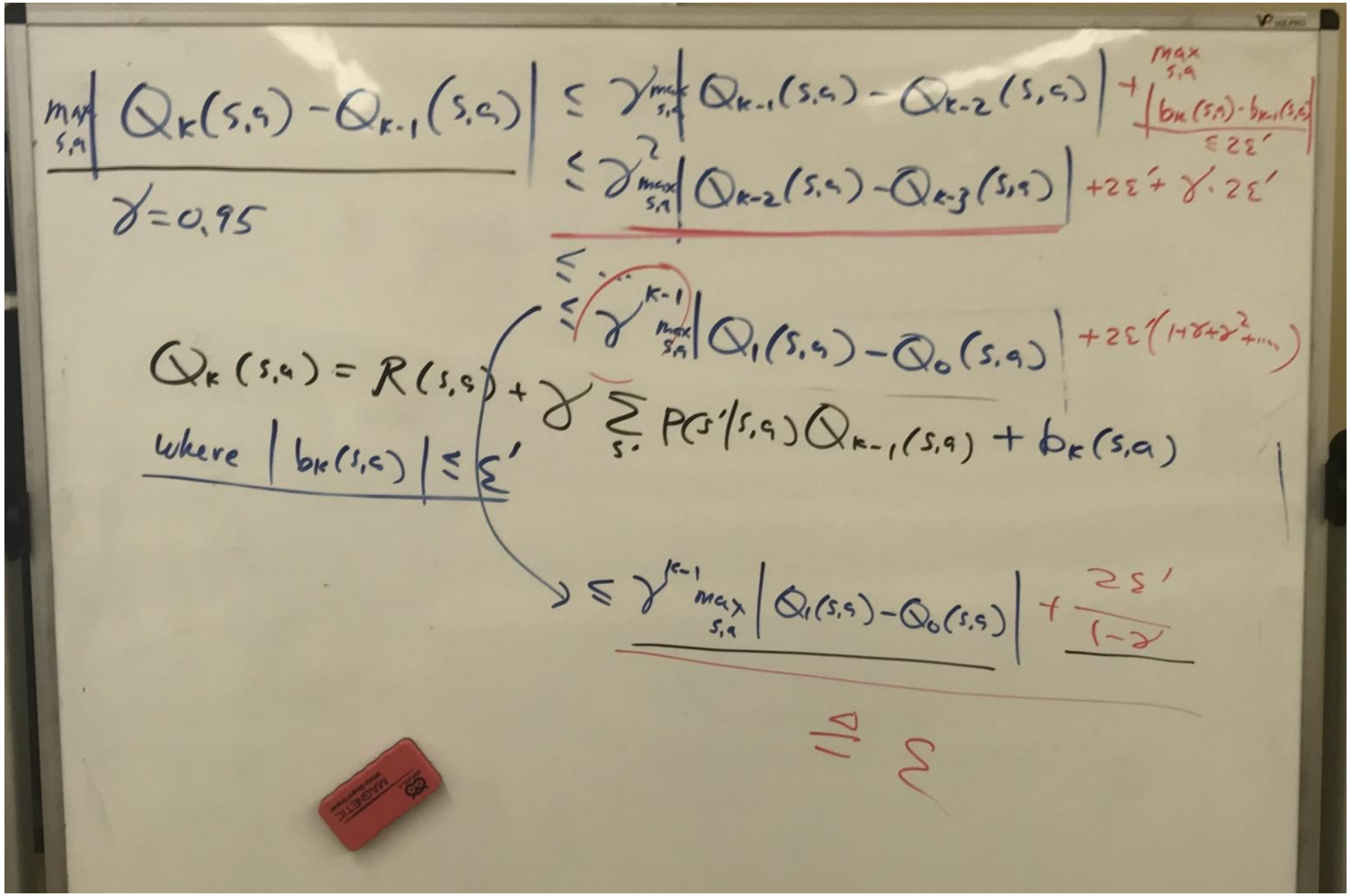
where $\hat{\pi}(s) = \underset{a}{\operatorname{argmax}}\, Q_k(s,a)$

Suboptimality $\leq \frac{1}{1-\gamma}$ ValueError

Completing the Analysis of VI (1st Step)



Analysis of DQN assuming sufficient exploration and Bellman Completeness



DQN can be offline

Let $\mathcal{B}$ consists of (s, a, r, s') tuples collected by a mixture of **arbitrary policies.**

Data collection

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

$\theta \leftarrow \theta_k$

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

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

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

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

Perform Value Iteration

Again, its success relies on 1) $\mathcal{B}$ contains data with sufficiently wide range of state-actions, 2) Bellman completeness.

The same theoretical analysis applies.

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
- Bellman completeness may not hold

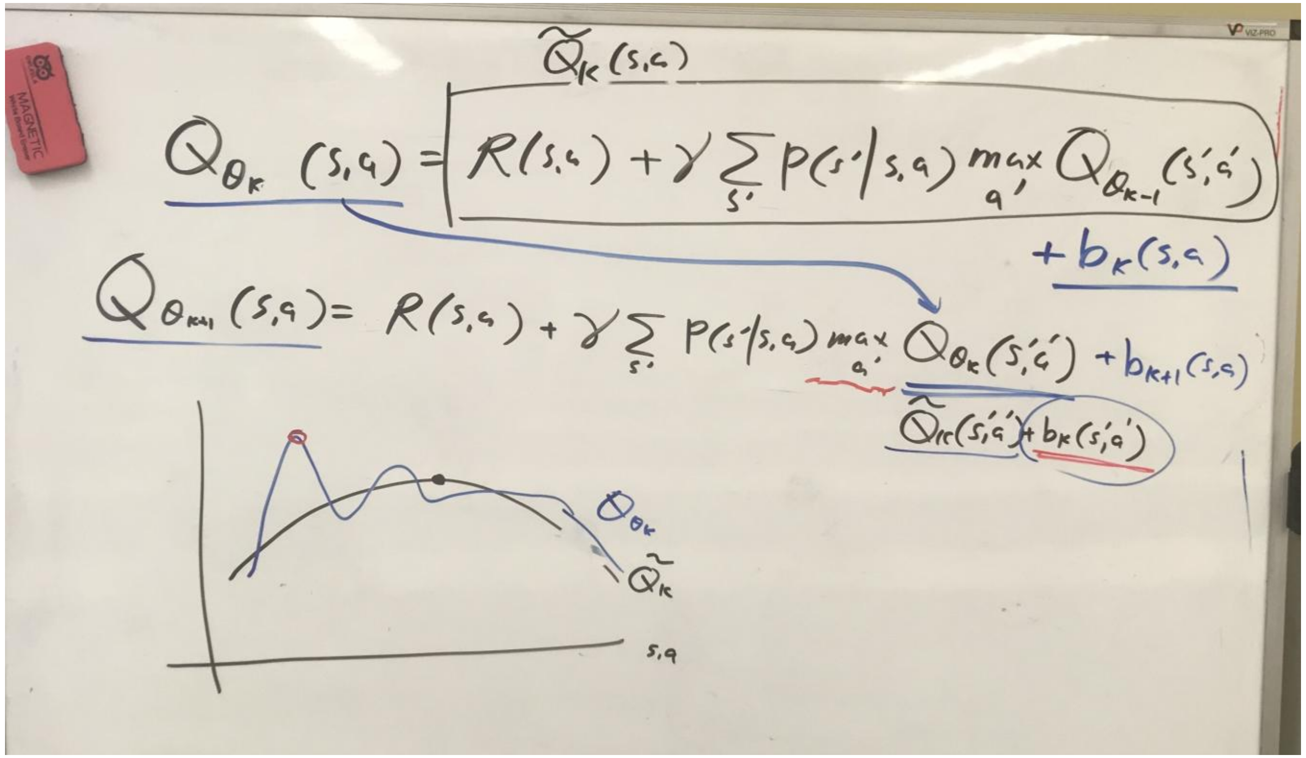
In either case, we may not have

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

This makes our previous analysis based on VI fails.

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 $\hat{\pi}(s) = \underset{a}{\operatorname{argmax}}\, Q_{\theta_k}(s,a)$ could be very bad.



When DQN cannot well-approximate VI

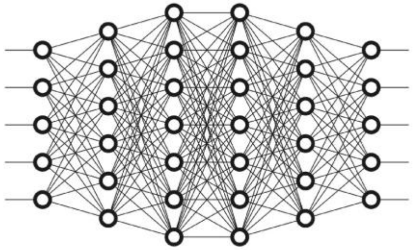
Such “seeking the error” behavior is due to “**bootstrapping**”

- An issue only in MDP but not in bandits

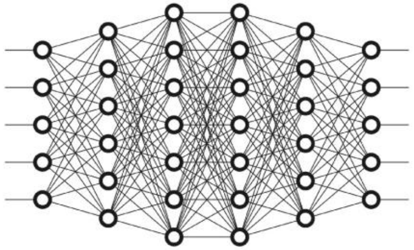
To prevent overestimation, two strategies are

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

Double DQN (v1)

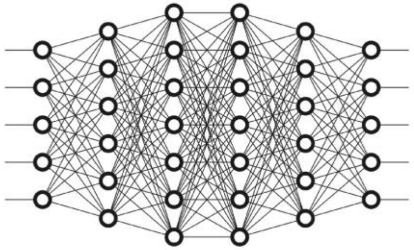


θ_1

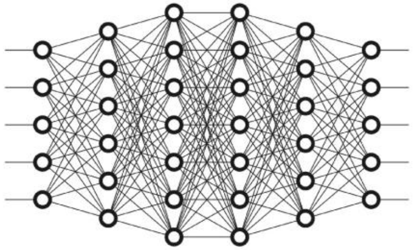


$\bar{\theta}_1$

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



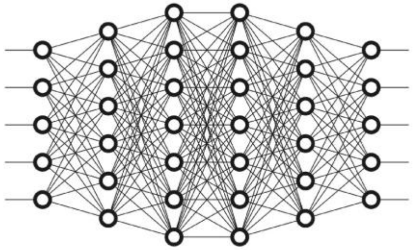
θ_2

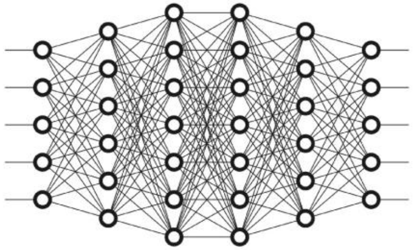


$\bar{\theta}_2$

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

Double DQN (v1)

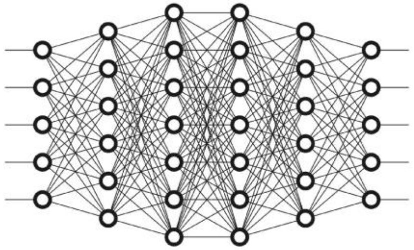


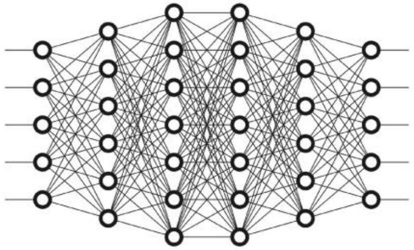


θ_1 $\bar{\theta}_1$

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

$$\underset{a'}{\text{argmax}}\, Q_{\bar{\theta}_2}(s',a')$$



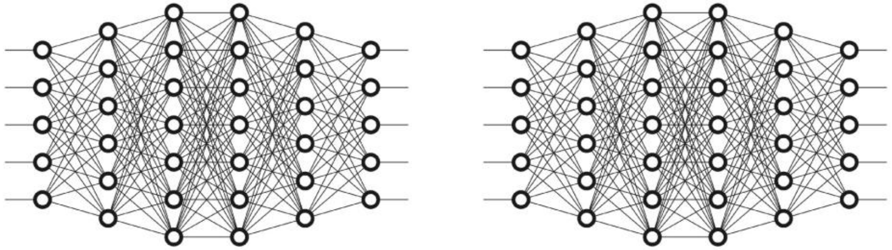


θ_2 $\bar{\theta}_2$

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

$$\underset{a'}{\text{argmax}}\, Q_{\bar{\theta}_1}(s',a')$$

Double DQN (v2)



θ $\qquad\qquad$ $\bar{\theta}$

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

$$\underset{a'}{\operatorname{argmax}}\, Q_{\theta}(s',a')$$

Hado van Hasselt, Arthur Guez, David Silver. Deep Reinforcement Learning with Double Q-learning. 2015.

Conservative Q-learning (CQL)

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

$$+ \alpha \sum_{i \in \mathcal{B}} \left(\log \left(\sum_{a} \exp(Q_\theta(s_i, a)) \right) - Q_\theta(s_i, a_i) \right)$$

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

$$+ \alpha \sum_{i \in \mathcal{B}} \left(\max_{\mu} \sum_{a} \mu(a|s_i) Q_\theta(s_i, a) - Q_\theta(s_i, a_i) - \mathrm{KL}(\mu(\cdot\,|s_i), \mathrm{Unif}) \right)$$

Aviral Kumar, Aurick Zhou, George Tucker, Sergey Levine Conservative Q-Learning for Offline Reinforcement Learning. 2020.

Comparison

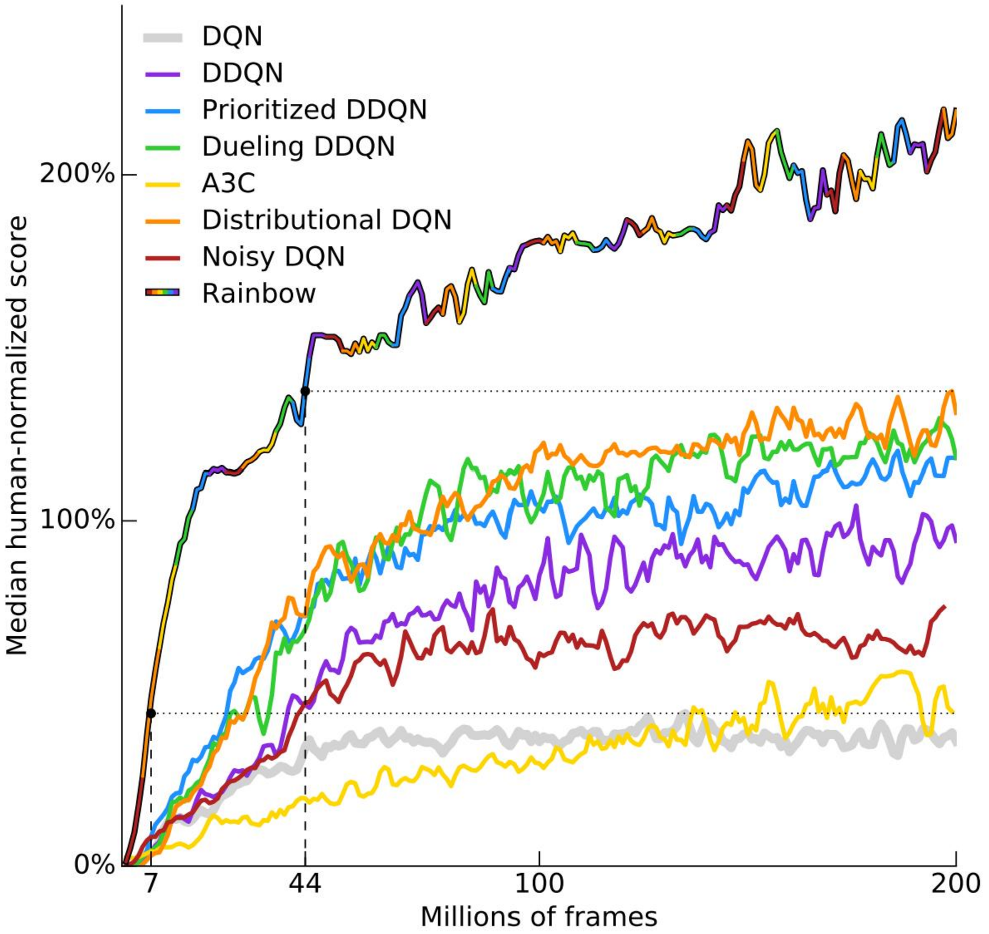
- Double-Q: make the $\arg\max_a Q_\theta(s, a)$ choice decoupled from θ
- Conservative-Q: mitigate the overestimation of $\max_a Q_\theta(s_i, a)$ over $Q_\theta(s_i, a_i)$

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
  - Conservative Q-learning

Improvements on DQN

- Dueling DDQN
- Prioritized replay
- Distributional DQN
- …



Rainbow: Combining Improvements in Deep Reinforcement Learning. 2018.

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 $\mathcal{B}$

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

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

cf. DQN with target network

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

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

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

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

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

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

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

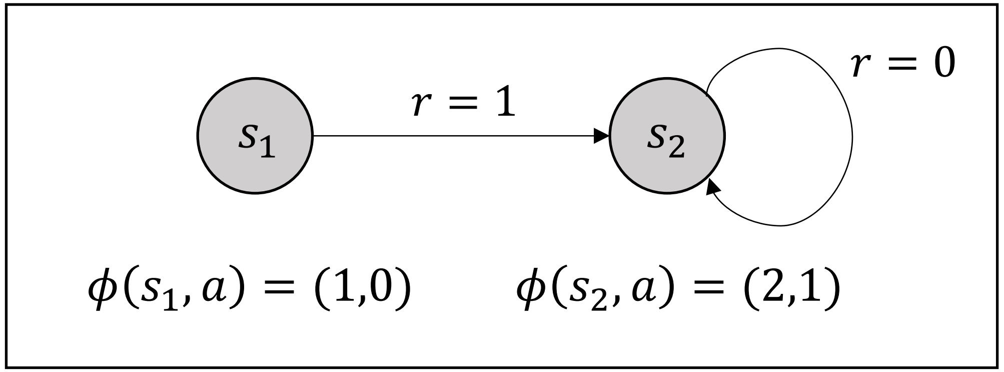
Randomly pick an i (or a mini-batch) from $\mathcal{B}$

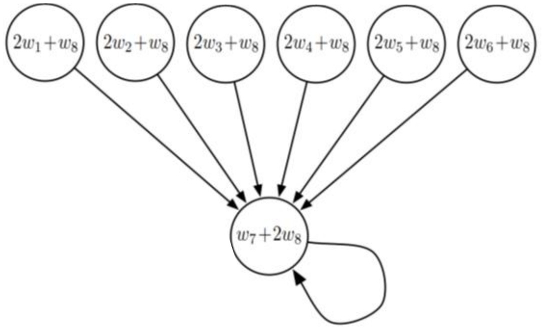
$$\theta \leftarrow \theta - \alpha \nabla_\theta \left(Q_\theta(s_i, a_i) - r_i - \gamma \max_{a'} Q_{\bar{\theta}}(s_i', a') \right)^2$$

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

An Unstable Variant

Diverges even when exploration assumption and Bellman completeness hold





Simplified from the "Baird's counterexample"
(see Sutton and Barto Section 11.2)

The Effect of Target Network

Let $KN = 100000$

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

$\theta_k \leftarrow \theta$

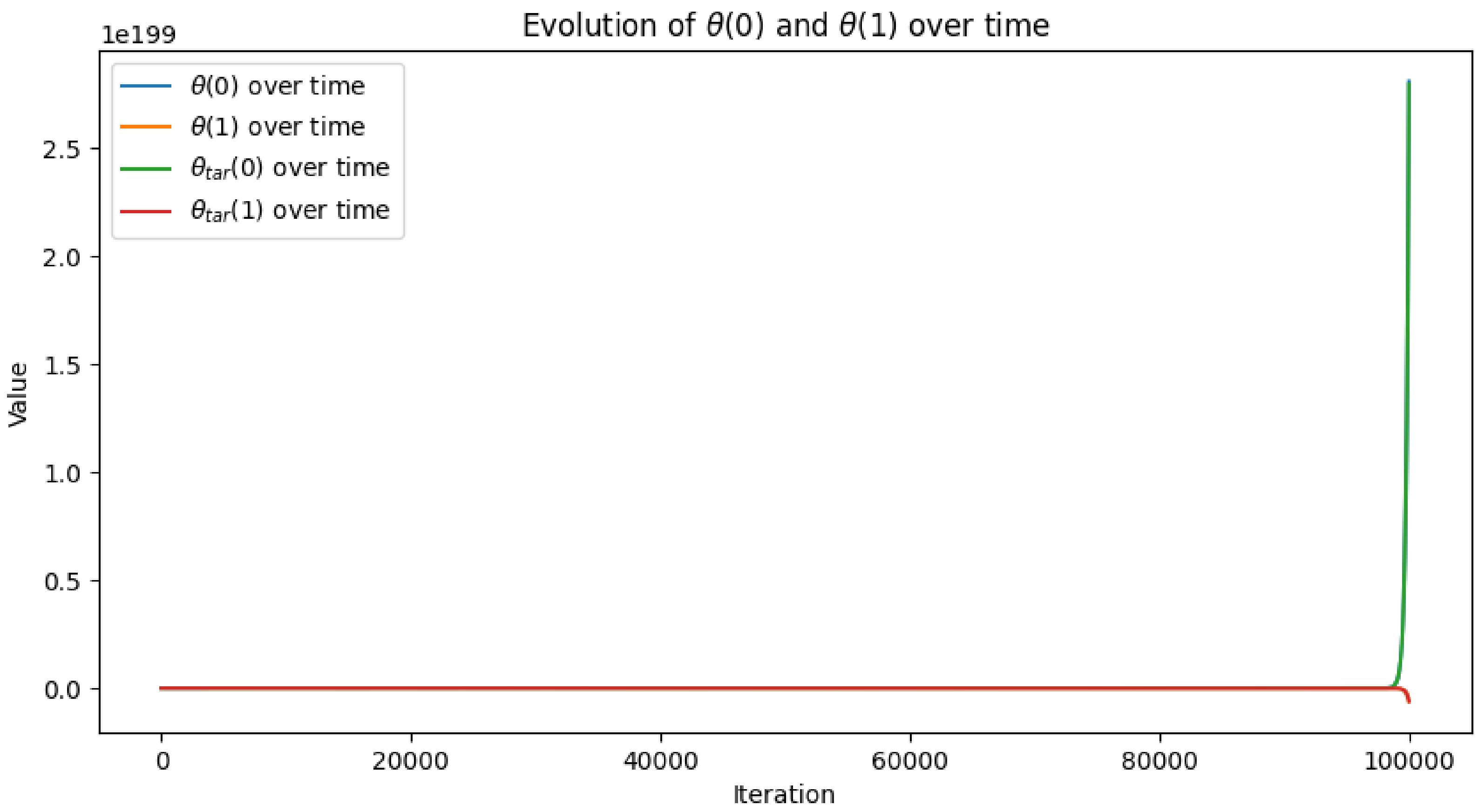
For $i = 1, \dots, N$:

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

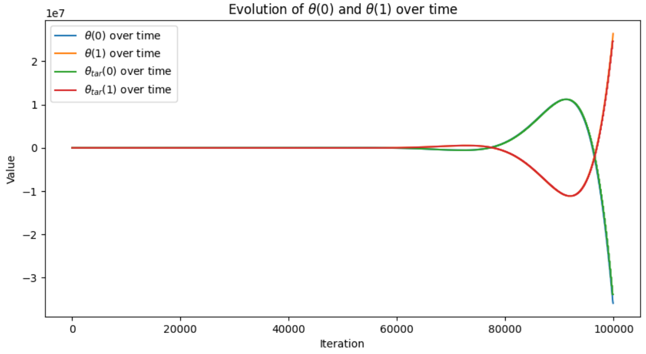
$$\theta \leftarrow \theta - \alpha\left(\phi(s,a)^\top\theta - r - \gamma\phi\left(s', a\right)^\top\theta_k\right)\phi(s,a)$$

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

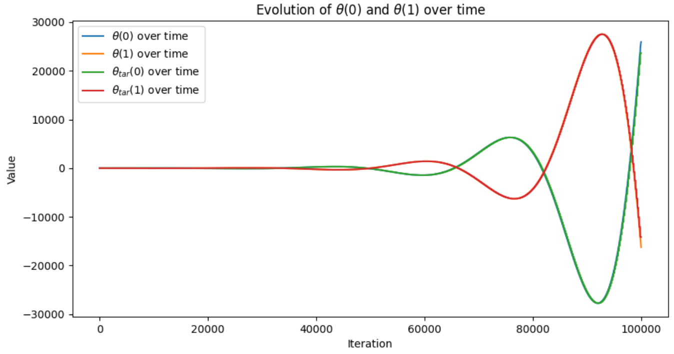
The Effect of Target Network



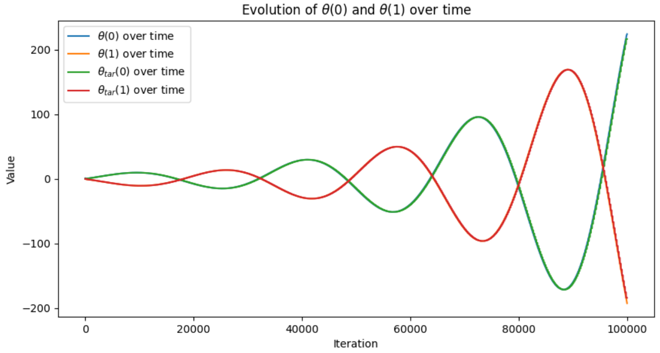
N=1



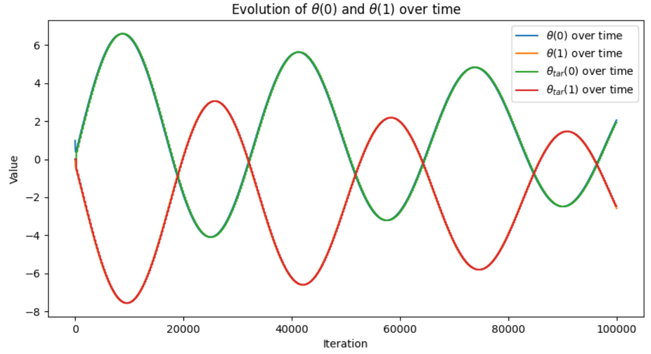
N=150



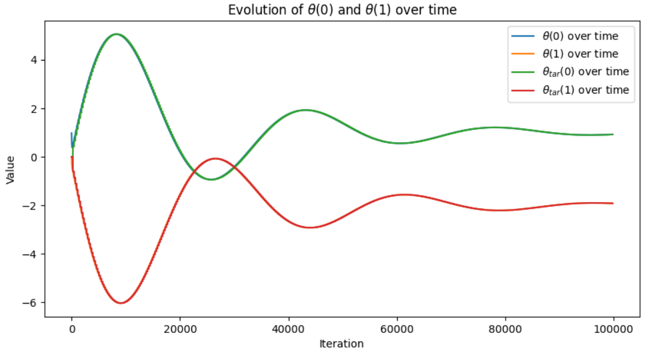
N=170



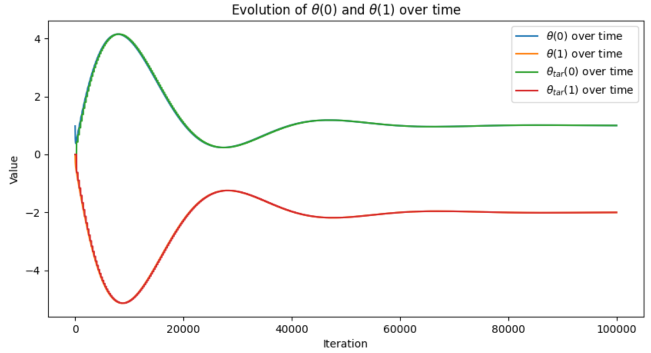
N=190



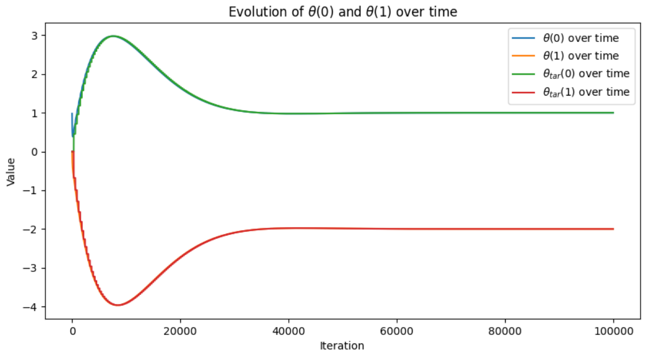
N=210



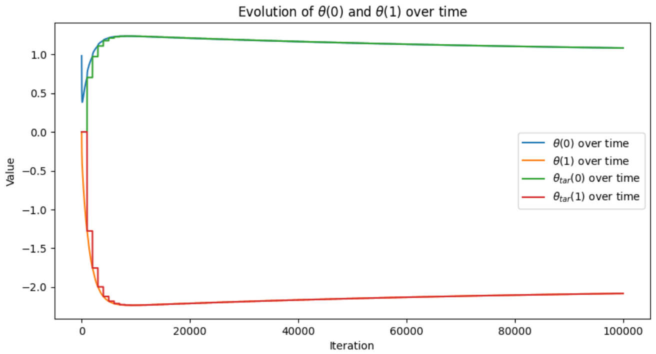
N=230



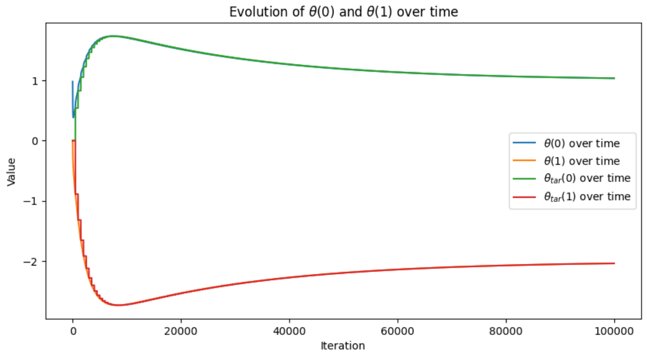
N=250



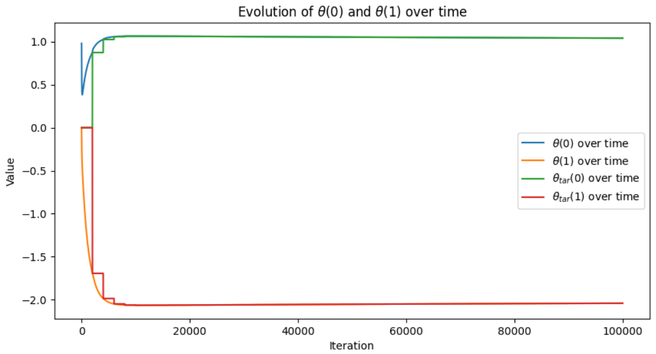
N=300



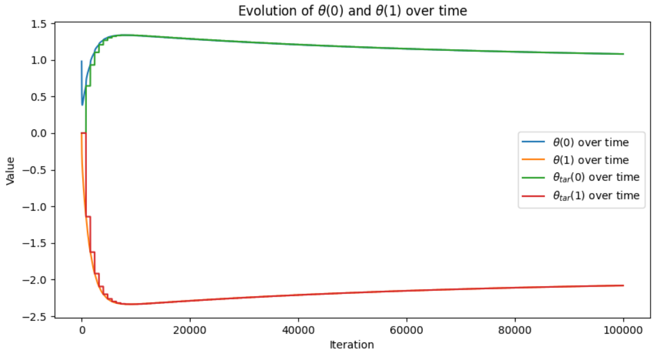
N=1000



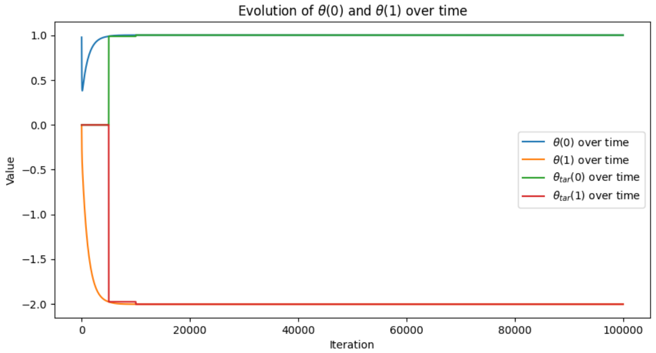
N=500



N=2000



N=800



N=5000

A Biased Variant

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 could be undesirable.

The underlying loss function of this algorithm is

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

Variants that Fail

- Both variants, while look somewhat reasonable, deviate from the idea of Value Iteration.