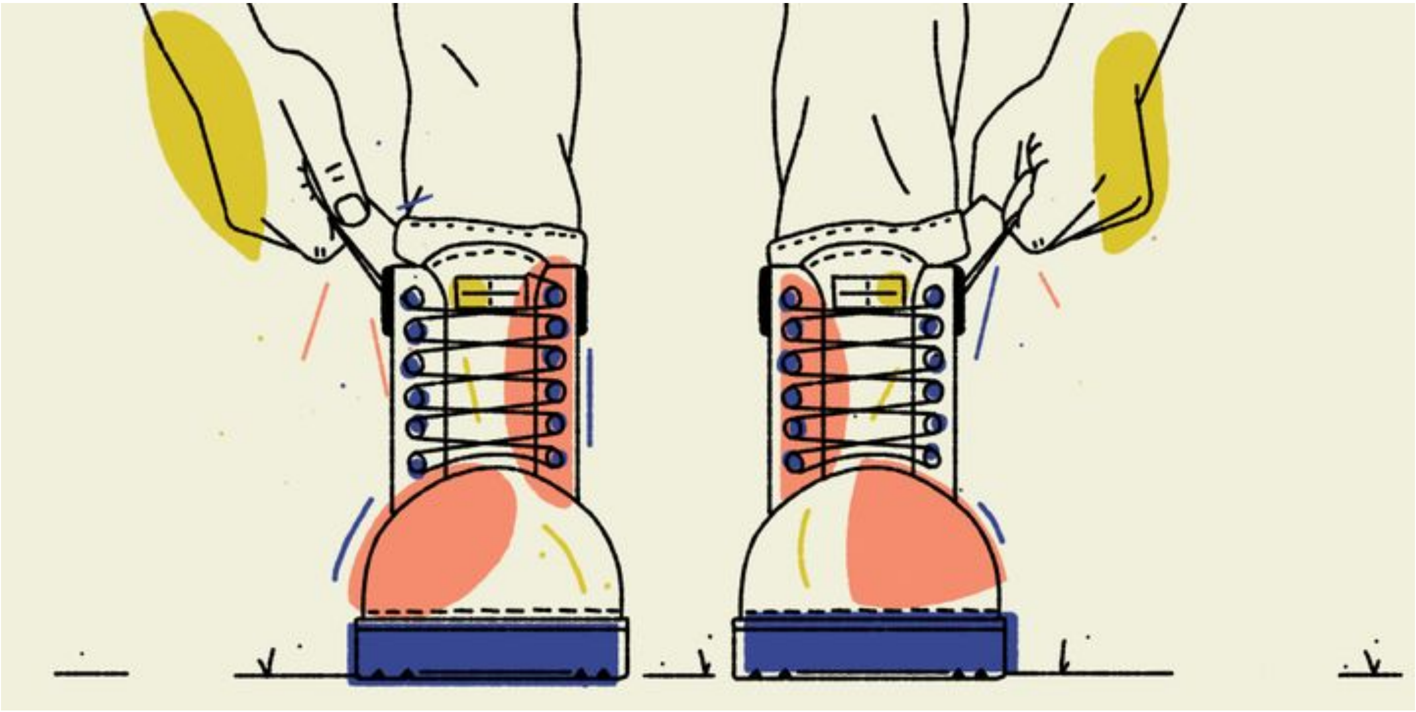
Target Network & Double DQN

Shusen Wang

http://wangshusen.github.io/

Bootstrapping



Bootstrapping: To lift oneself up by his bootstraps.

TD Learning for DQN

- In RL, bootstrapping means "*using an estimated value in the update step for the same kind of estimated value*".
- Use a transition, (s_t, a_t, r_t, s_{t+1}), to update $\mathbf{w}$.
  - TD target: $y_t = r_t + \gamma \cdot \max_{a} Q(s_{t+1}, a; \mathbf{w})$.
  - TD error: $\delta_t = Q(s_t, a_t; \mathbf{w}) - y_t$.
  - SGD: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial\, Q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$.

TD Learning for DQN

- In RL, bootstrapping means "*using an estimated value in the update step for the same kind of estimated value*".

- Use a transition, (s_t, a_t, r_t, s_{t+1}), to update $\mathbf{w}$.

  - TD target: $y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w})$.

TD target y_t is partly an estimate made by the DQN Q.

  - SGD: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial\, Q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$.

TD Learning for DQN

- In RL, bootstrapping means "*using an estimated value in the update step for the same kind of estimated value*".

- Use a transition, (s_t, a_t, r_t, s_{t+1}), to update $\mathbf{w}$.

  - TD target: $y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w})$.

TD target y_t is partly an estimate made by the DQN Q.

  - SGD: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \left(Q(s_t, a_t; \mathbf{w}) - y_t\right) \cdot \frac{\partial\, Q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$.

We in use y_t, which is partly based on Q, to update the DQN itself, $Q(s_t, a_t; \mathbf{w})$.

Problem of Overestimation

Problem of Overestimation

- TD learning makes DQN overestimate action-values. (Why?)
- Reason 1: The maximization.
  - TD target: $y_t = r_t + \gamma \cdot \max_{a} Q(s_{t+1}, a; \mathbf{w})$.
  - TD target is bigger than the real action-value.
- Reason 2: Bootstrapping propagates the maximization.

Reason 1: Maximization

- Let $x_1, x_2, \cdots, x_n$ be observed real numbers.
- Add zero-mean random noise to $x_1, \cdots, x_n$ and obtain $Q_1, \cdots, Q_n$.
- The zero-mean noise does not affect the mean:

$$\mathbb{E}[\mathrm{mean}_i(Q_i)] \ = \ \mathrm{mean}_i(x_i).$$

- The zero-mean noise increases the maximum:

$$\mathbb{E}[\max_i(Q_i)] \ \geq \ \max_i(x_i).$$

- The zero-mean noise decreases the minimum:

$$\mathbb{E}[\min_i(Q_i)] \ \leq \ \min_i(x_i).$$

Reason 1: Maximization

- Let $x(a_1), \cdots, x(a_n)$ be the true action-values.
- $Q(s, a_1; \mathbf{w}), \cdots, Q(s, a_n; \mathbf{w})$ be noisy estimates made by DQN.
- Suppose the estimate is unbiased:

$$\operatorname*{mean}_a \big(x(a)\big) = \operatorname*{mean}_a \big(Q(s, a; \mathbf{w})\big).$$

- $q = \max_a Q(s, a; \mathbf{w})$, is typically an overestimate:

$$q \geq \max_a \big(x(a)\big).$$

Reason 1: Maximization

- We conclude that $q_{t+1} = \max_a Q(s_{t+1}, a; \mathbf{w})$ is an overestimate of the true action-value at time $t+1$.
- The TD target, $y_t = r_t + \gamma \cdot q_{t+1}$, is thereby an overestimate.
- TD learning pushes $Q(s_t, a_t; \mathbf{w})$ towards the TD target which overestimates the true action-value.

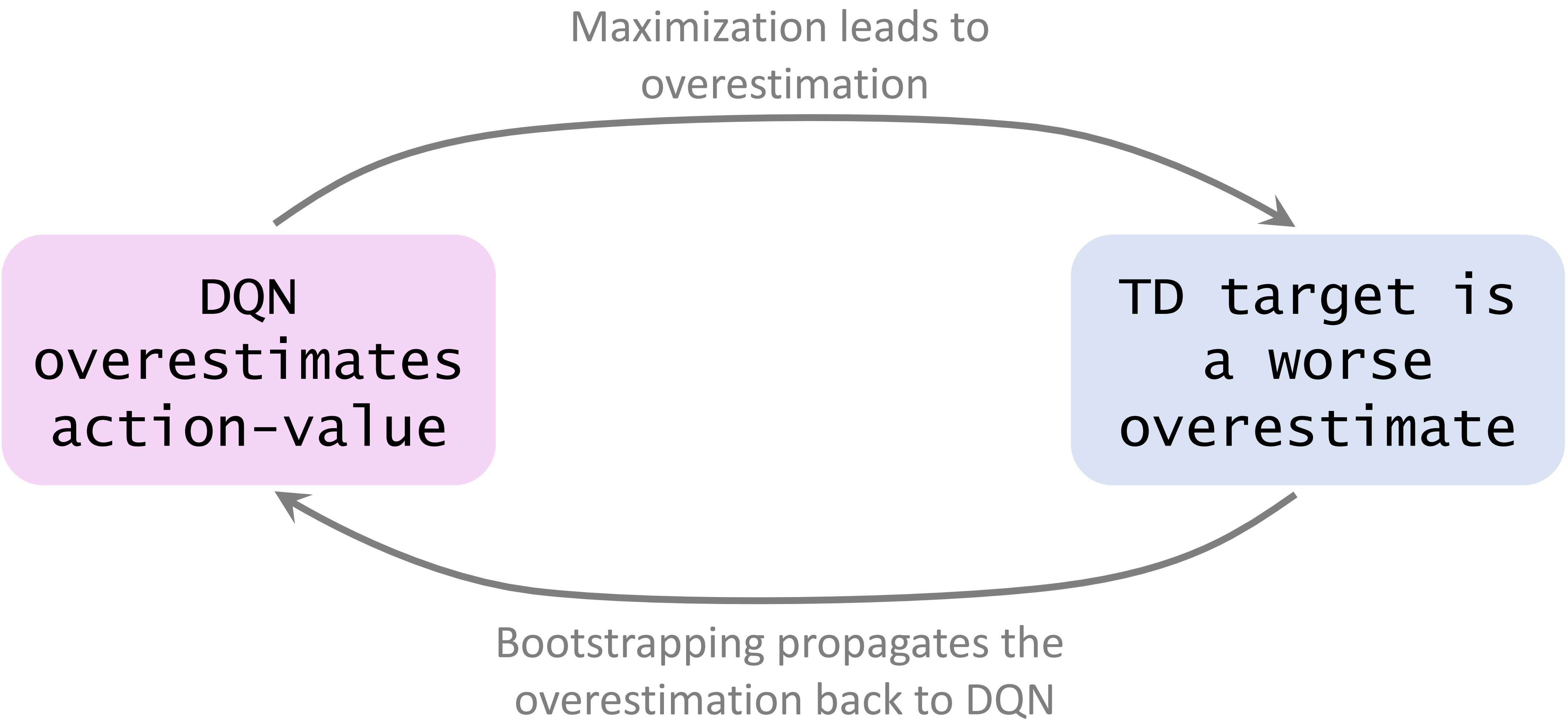
Reason 2: Bootstrapping

- TD learning is bootstrapping.
  - TD target in part uses $q_{t+1} = \max_{a} Q(s_{t+1}, a; \mathbf{w})$.
  - Use the TD target for updating $Q(s_t, a_t; \mathbf{w})$.
- Suppose DQN overestimates the action-value.
- Then $Q(s_{t+1}, a; \mathbf{w})$ is an overestimation.

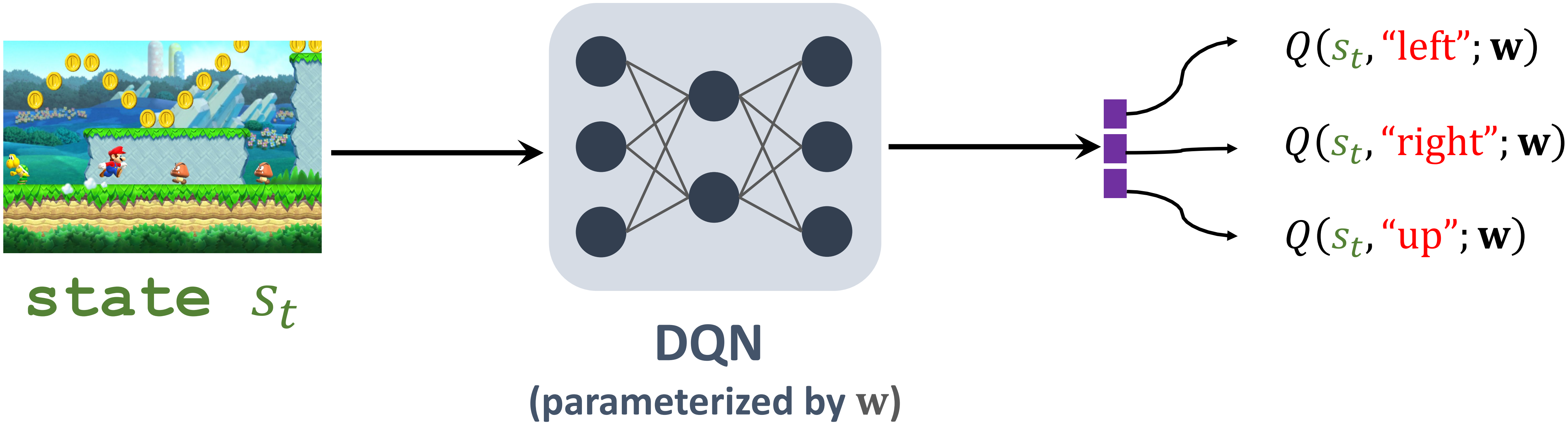
Reason 2: Bootstrapping

- TD learning is bootstrapping.
  - TD target in part uses $q_{t+1} = \max_{a} Q(s_{t+1}, a; \mathbf{w})$.
  - Use the TD target for updating $Q(s_t, a_t; \mathbf{w})$.
- Suppose DQN overestimates the action-value.
- Then $Q(s_{t+1}, a; \mathbf{w})$ is an overestimation.
- The maximization further pushes q_{t+1} up.
- When q_{t+1} is used for updating $Q(s_t, a_t; \mathbf{w})$, overestimation is propagated to DQN.

Why does overestimation happen?



Why is overestimation a shortcoming?

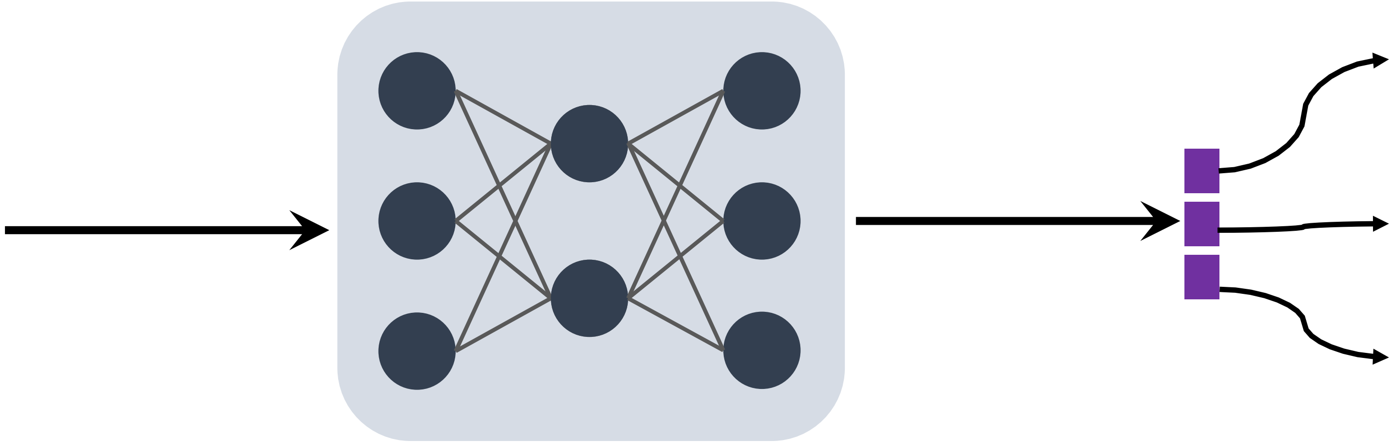


Why is overestimation a shortcoming?

The agent is controlled by the DQN: $a_t = \underset{a}{\mathrm{argmax}}\, Q(s_t, a; \mathbf{w})$.

Uniform overestimation is not a problem.

state s_t



$Q(s_t, \text{"left"}; \mathbf{w})$

$Q(s_t, \text{"right"}; \mathbf{w})$

$Q(s_t, \text{"up"}; \mathbf{w})$

DQN

(parameterized by $\mathbf{w}$)

Why is overestimation a shortcoming?

The agent is controlled by the DQN: $a_t = \underset{a}{\operatorname{argmax}}\, Q(s_t, a; \mathbf{w})$.

Uniform overestimation is not a problem.

- $Q^\star(s, a_1) = 200, \quad Q^\star(s, a_2) = 100,$ and $Q^\star(s, a_3) = 230.$
- Action a_3 will be selected.
- Suppose $Q(s, a_i; \mathbf{w}) = Q^\star(s, a_i) + 100$, for all a_i.
- Then a_3 still has the highest value and will be selected.

Why is overestimation a shortcoming?

The agent is controlled by the DQN: $a_t = \underset{a}{\text{argmax}}\, Q(s_t, a; \mathbf{w})$.

Uniform overestimation is not a problem.

Non-uniform overestimation is problematic.

- $Q^\star(s, a_1) = 200$, $Q^\star(s, a_2) = 100$, and $Q^\star(s, a_3) = 230$.
- $Q(s, a_1; \mathbf{w}) = 280$, $Q(s, a_2; \mathbf{w}) = 300$, and $Q(s, a_3; \mathbf{w}) = 240$, .
- Then a_2 (which is not good) will be selected.

Why is overestimation a shortcoming?

Unfortunately, the overestimation is non-uniform.

- Use a transition, (s_t, a_t, r_t, s_{t+1}), to update $\mathbf{w}$.
- The TD target, y_t, overestimates $Q^{\star}(s_t, a_t)$.
- TD algorithm pushes $Q(s_t, a_t; \mathbf{w})$ towards y_t.
- Thus, $Q(s_t, a_t; \mathbf{w})$ overestimates $Q^{\star}(s_t, a_t)$.

Why is overestimation a shortcoming?

Unfortunately, the overestimation is non-uniform.

- Use a transition, (s_t, a_t, r_t, s_{t+1}), to update $\mathbf{w}$.
- The TD target, y_t, overestimates $Q^\star(s_t, a_t)$.
- TD algorithm pushes $Q(s_t, a_t; \mathbf{w})$ towards y_t.
- Thus, $Q(s_t, a_t; \mathbf{w})$ overestimates $Q^\star(s_t, a_t)$.

The more frequently (s, a) appears in the replay buffer,
the more $Q(s, a; \mathbf{w})$ overestimates $Q^\star(s, a)$.

Solutions

- **Problem:** DQN trained by TD overestimates action-values.
- **Solution 1:** Use a target network [1] to compute TD targets. (Address the problem caused by bootstrapping.)
- **Solution 2:** Use double DQN [2] to alleviate the overestimation caused by maximization.

Reference:

1. Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 2015.
2. Van Hasselt, Guez, & Silver. Deep reinforcement learning with double Q-learning. In *AAAI*, 2016.

Using Target Network

Reference:

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

Target Network

- Target network: $Q(s, a; \mathbf{w}^{-})$
  - The same network structure as the DQN, $Q(s, a; \mathbf{w})$.
  - Different parameters: $\mathbf{w}^{-} \neq \mathbf{w}$.
- Use $Q(s, a; \mathbf{w})$ to control the agent and collect experience:

$$\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}.$$

- Use $Q(s, a; \mathbf{w}^{-})$ to compute TD target:

$$y_t = r_t + \gamma \cdot \max_{a} Q(s_{t+1}, a; \mathbf{w}^{-}).$$

TD Learning with Target Network

- Use a transition, (s_t, a_t, r_t, s_{t+1}), to update $\mathbf{w}$.
  - TD target: $y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w}^-)$.
  - TD error: $\delta_t = Q(s_t, a_t; \mathbf{w}) - y_t$.
  - SGD: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial\, Q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$.

Update Target Network

- Periodically update $\mathbf{w}^{-}$:
- Option 1: $\mathbf{w}^{-} \leftarrow \mathbf{w}$.
- Option 2: $\mathbf{w}^{-} \leftarrow \tau \cdot \mathbf{w} + (1 - \tau) \cdot \mathbf{w}^{-}$

Comparisons

- TD learning with naïve update:

$$\text{TD Target: } y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w}).$$

- TD learning with target network:

$$\text{TD Target: } y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w}^-).$$

- Though better than the naïve update, TD learning with target network nevertheless overestimate action-values.

Double DQN

Reference:

1. Van Hasselt, Guez, & Silver. Deep reinforcement learning with double Q-learning. In *AAAI*, 2016.

Naïve Update

TD Target: $y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w})$.

Reference:

1. Mnih et al. Playing Atari with deep reinforcement learning. In *NIPS Workshop*, 2013.

Naïve Update

- Selection using DQN:

$$a^\star = \underset{a}{\mathrm{argmax}}\, Q(s_{t+1}, a; \mathbf{w}).$$

- Evaluation using DQN:

$$y_t = r_t + \gamma \cdot Q(s_{t+1}, a^\star; \mathbf{w}).$$

- Serious overestimation.

Reference:

1. Mnih et al. Playing Atari with deep reinforcement learning. In *NIPS Workshop*, 2013.

Using Target Network

- Selection using target network:

$$a^\star = \underset{a}{\operatorname{argmax}}\, Q(s_{t+1}, a; \mathbf{w}^-).$$

- Evaluation using target network:

$$y_t = r_t + \gamma \cdot Q(s_{t+1}, a^\star; \mathbf{w}^-).$$

- Works better, but overestimation is still serious.

Reference:

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

Double DQN

- Selection using DQN:

$$a^\star = \underset{a}{\operatorname{argmax}}\, Q(s_{t+1}, a; \mathbf{w}).$$

- Evaluation using target network:

$$y_t = r_t + \gamma \cdot Q(s_{t+1}, a^\star; \mathbf{w}^-).$$

- Works even better, but overestimation still happens.

Reference:

1. Van Hasselt, Guez, & Silver. Deep reinforcement learning with double Q-learning. In *AAAI*, 2016.

Why does double DQN work better?

- Double DQN decouples the selection from the evaluation.
- Selection using DQN: $a^\star = \underset{a}{\operatorname{argmax}}\, Q(s_{t+1}, a; \mathbf{w})$.
- Evaluation using target network: $y_t = r_t + \gamma \cdot Q(s_{t+1}, a^\star; \mathbf{w}^-)$.

Why does double DQN work better?

- Double DQN decouples the selection from the evaluation.
- Selection using DQN: $a^\star = \underset{a}{\mathrm{argmax}}\, Q(s_{t+1}, a; \mathbf{w})$.
- Evaluation using target network: $y_t = r_t + \gamma \cdot Q(s_{t+1}, a^\star; \mathbf{w}^-)$.
- Double DQN alleviates overestimation:

$$\boxed{Q(s_{t+1}, a^\star; \mathbf{w}^-)} \leq \boxed{\max_{a} Q(s_{t+1}, a; \mathbf{w}^-)}.$$

Estimate by Double DQN

Estimate by target network

Summary

Problem of Overestimation

- Because of the maximization, the TD target overestimates the true action-value.
- By creating a "positive feedback loop", bootstrapping further exacerbates the overestimate.
- Target network can partly avoid bootstrapping. (Not completely, because $\mathbf{w}^{-}$ depends on $\mathbf{w}$.)
- Double DQN alleviate the overestimate caused by the maximization.

Computing TD Targets

	Selection	Evaluation
Naïve Update	DQN	DQN
Using Target Network	Target Network	Target Network
Double DQN	DQN	Target Network

Thank you!