Deep Reinforcement Learning
&
Applications

Chris Reinke

Reinforcement Learning (RL)

Learning multi-step decision tasks from interactions.

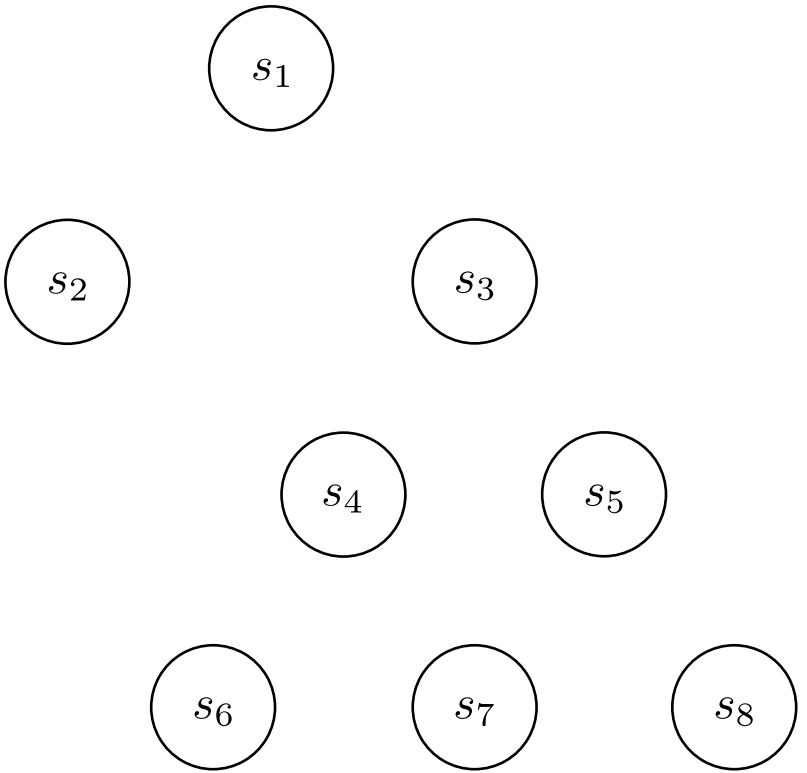
Reinforcement Learning

- Tasks are Markov Decision Problems (MDPs)

Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

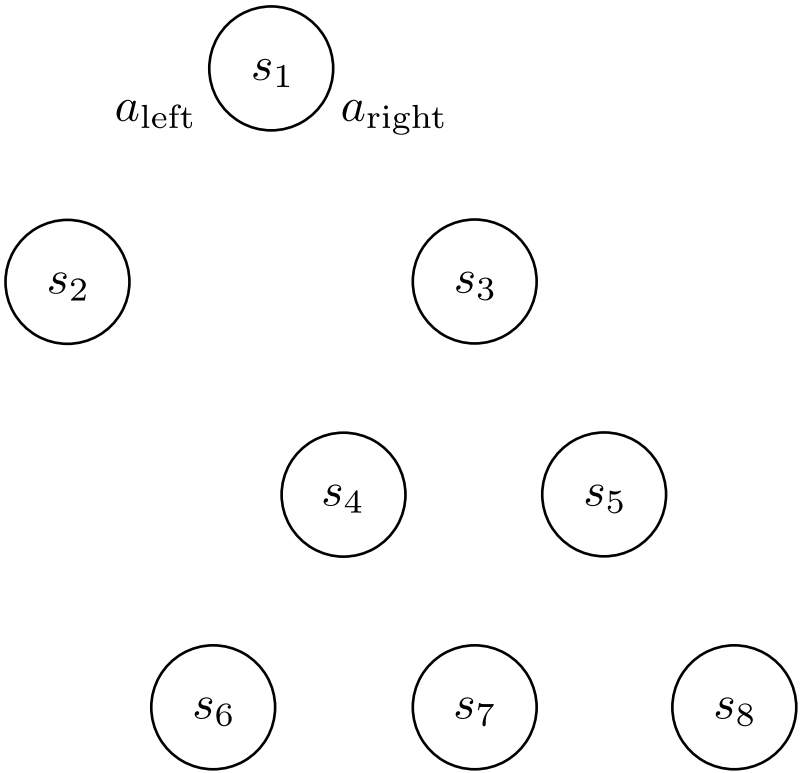
- Tasks are Markov Decision Problems (MDPs):
  - Set of states: $\mathcal{S} = \{s_1, \ldots, s_8\}$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

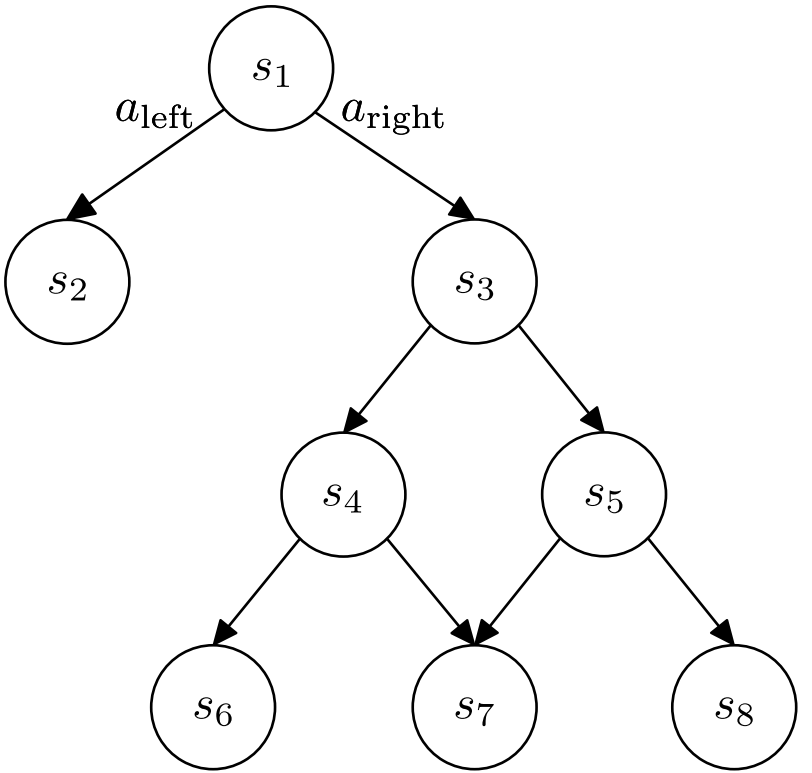
- Tasks are Markov Decision Problems (MDPs):
  - Set of states: $\mathcal{S} = \{s_1, \ldots, s_8\}$
  - Set of Actions: $\mathcal{A} = \{a_{\text{left}}, a_{\text{right}}\}$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

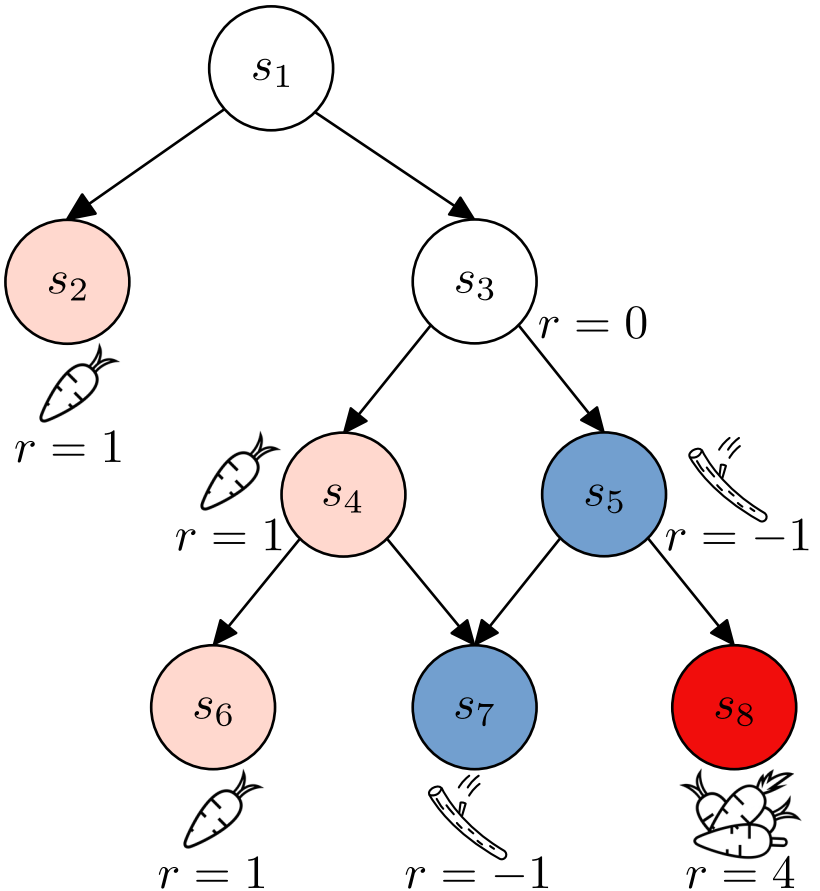
- Tasks are Markov Decision Problems (MDPs):
  - Set of states: $\mathcal{S} = \{s_1, \ldots, s_8\}$
  - Set of Actions: $\mathcal{A} = \{a_{\text{left}}, a_{\text{right}}\}$
  - Transition function: $\Pr(s_{t+1}|s_t, a_t)$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

- Tasks are Markov Decision Problems (MDPs):
  - Set of states: $\mathcal{S} = \{s_1, \ldots, s_8\}$
  - Set of Actions: $\mathcal{A} = \{a_{\text{left}}, a_{\text{right}}\}$
  - Transition function: $\Pr(s_{t+1}|s_t, a_t)$
  - Reward function: $r_t = R(s_{t+1})$



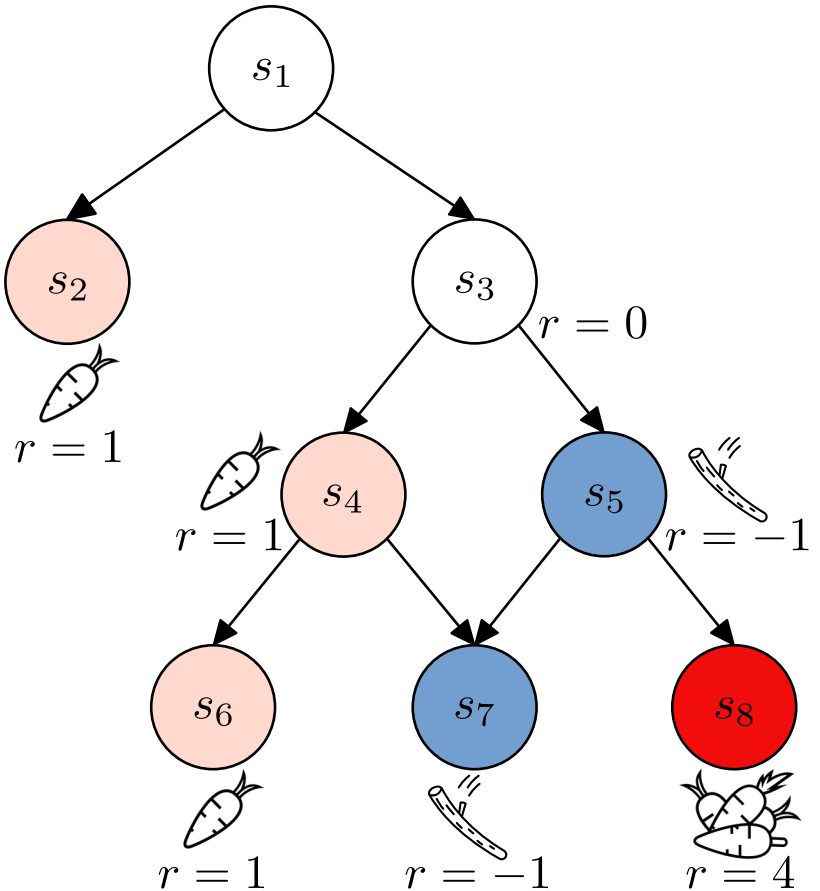
Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

- Tasks are Markov Decision Problems (MDPs):
  - Set of states: $\mathcal{S} = \{s_1, \ldots, s_8\}$
  - Set of Actions: $\mathcal{A} = \{a_{\text{left}}, a_{\text{right}}\}$
  - Transition function: $\Pr(s_{t+1}|s_t, a_t)$
  - Reward function: $r_t = R(s_{t+1})$

- Goal: Maximization of return

$$E\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1})\right] \quad = \quad E\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots\right]$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\gamma \in [0, 1)$ - Discount factor

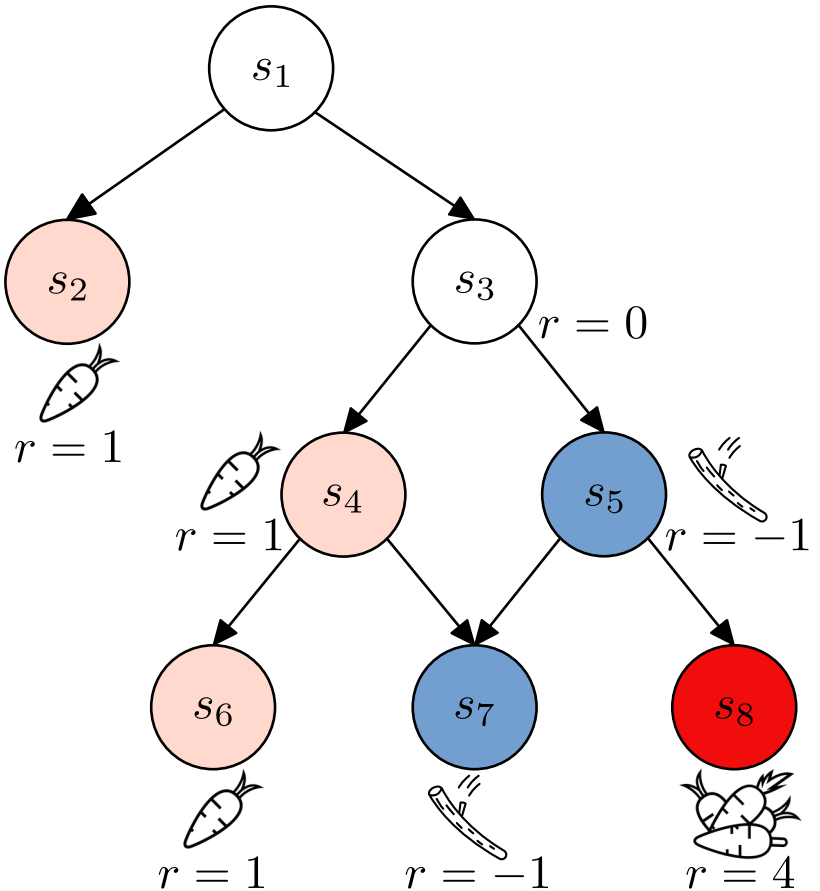
Reinforcement Learning

- Tasks are Markov Decision Problems (MDPs):
  - Set of states: $\mathcal{S} = \{s_1, \ldots, s_8\}$
  - Set of Actions: $\mathcal{A} = \{a_{\text{left}}, a_{\text{right}}\}$
  - Transition function: $\Pr(s_{t+1}|s_t, a_t)$
  - Reward function: $r_t = R(s_{t+1})$

- Goal: Maximization of return

$$E\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1})\right] \quad = \quad E\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots\right]$$

- Policy: $a_t = \pi(s_t) \quad | \quad a_t \sim \pi(s_t)$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\gamma \in [0, 1)$ - Discount factor

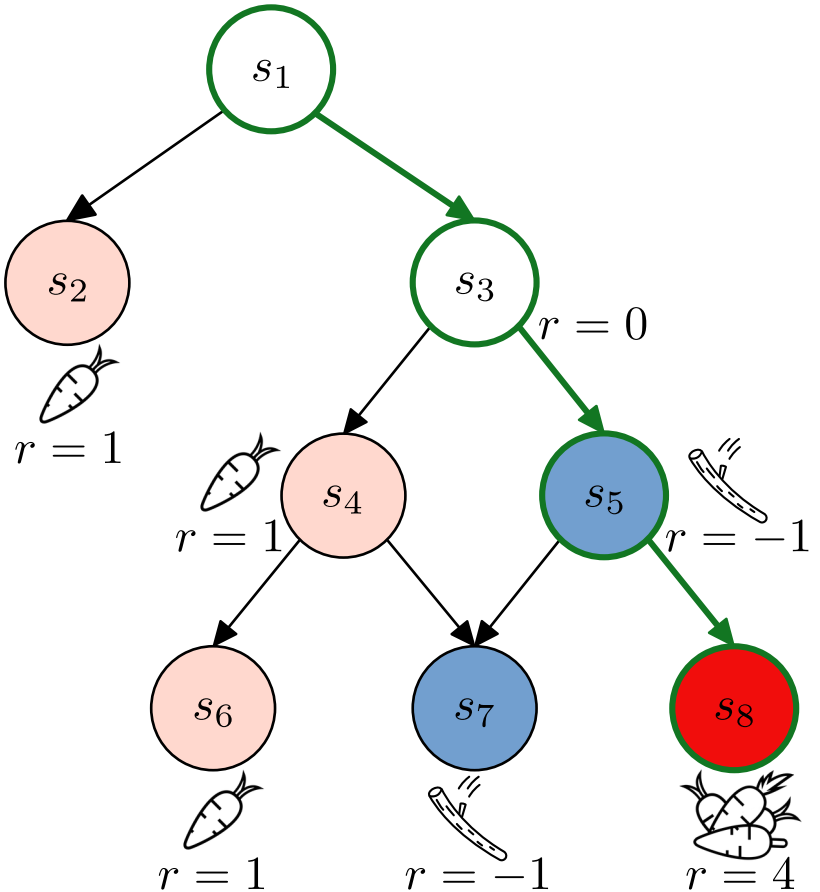
Reinforcement Learning

- Tasks are Markov Decision Problems (MDPs):
  - Set of states: $\mathcal{S} = \{s_1, \ldots, s_8\}$
  - Set of Actions: $\mathcal{A} = \{a_{\text{left}}, a_{\text{right}}\}$
  - Transition function: $\Pr(s_{t+1}|s_t, a_t)$
  - Reward function: $r_t = R(s_{t+1})$

- Goal: Maximization of return

$$E\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1})\right] \quad = \quad E\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots\right]$$

- Policy: $a_t = \pi(s_t) \quad | \quad a_t \sim \pi(s_t)$



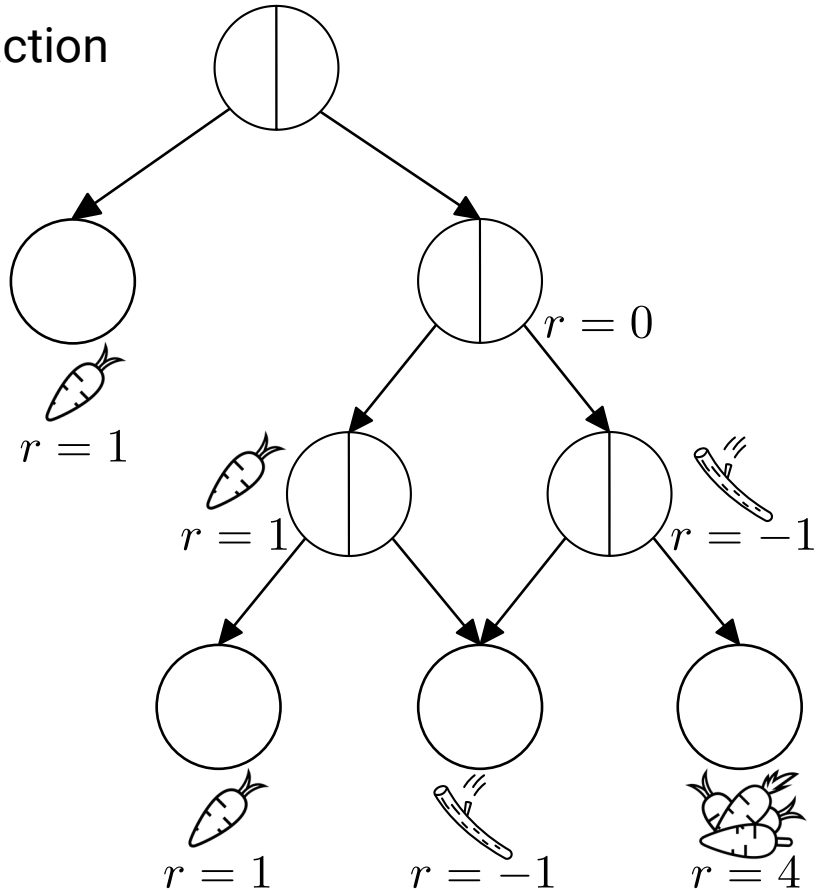
Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\gamma \in [0, 1)$ - Discount factor

Reinforcement Learning

- Q-function: Expected future reward sum per state-action

$$Q^{\pi}(s_t, a_t) = E_{\pi}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots\right]$$



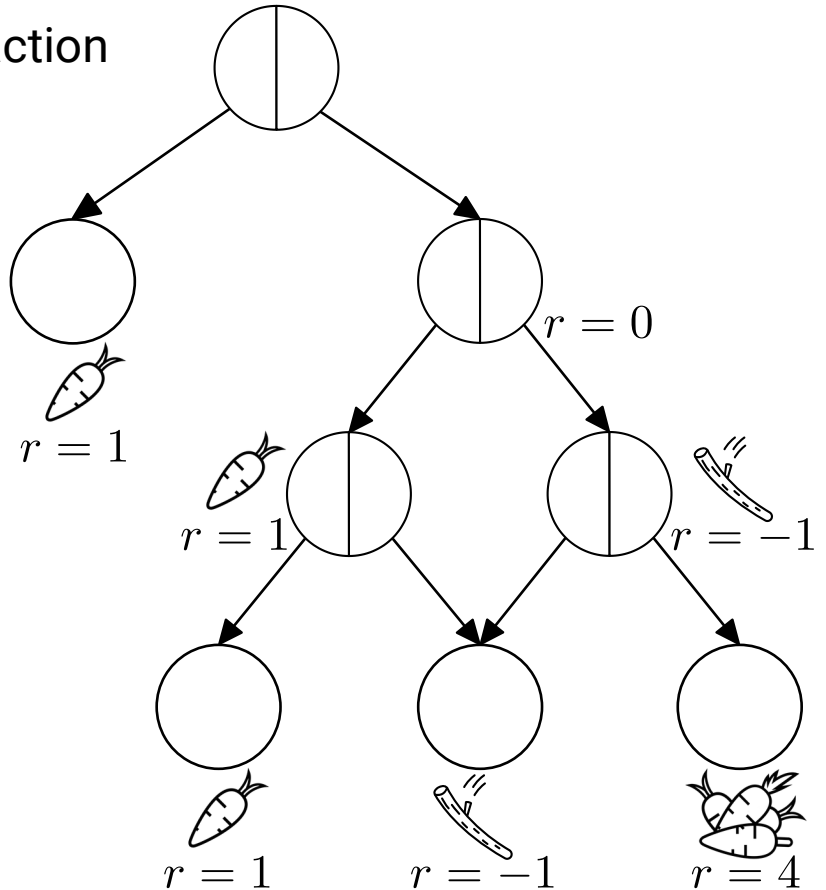
Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\gamma \in [0, 1)$ - Discount factor

Reinforcement Learning

- Q-function: Expected future reward sum per state-action

$$Q^{\pi}(s_t, a_t) = E_{\pi}\left[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots\right]$$

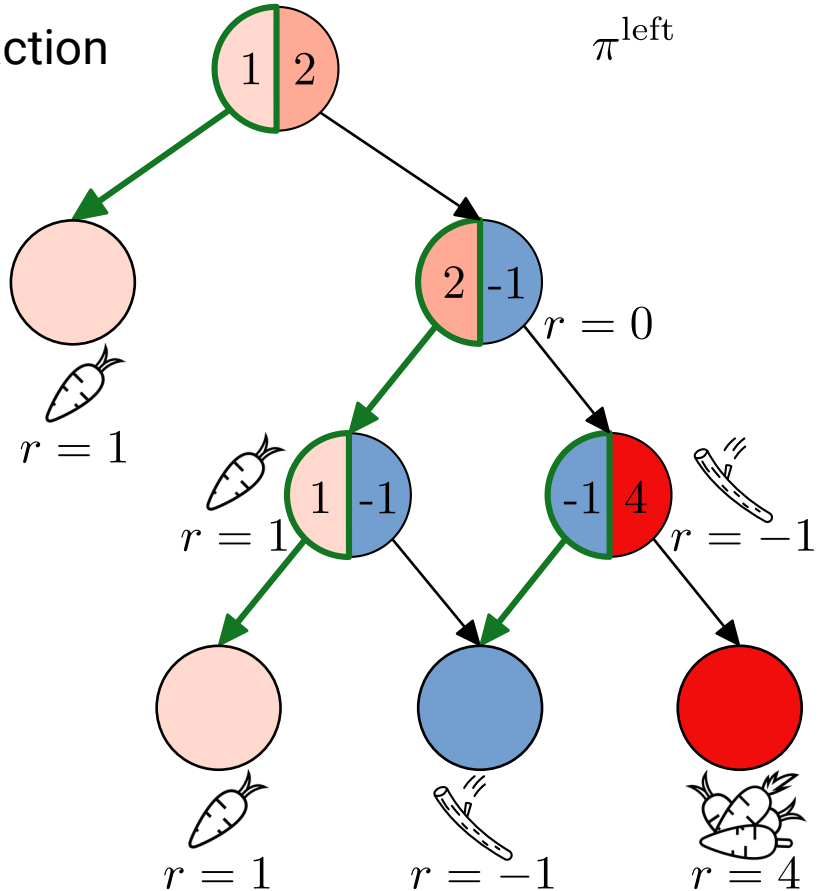


Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

- Q-function: Expected future reward sum per state-action

$$Q^{\pi}(s_t, a_t) = E_{\pi}\left[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots\right]$$



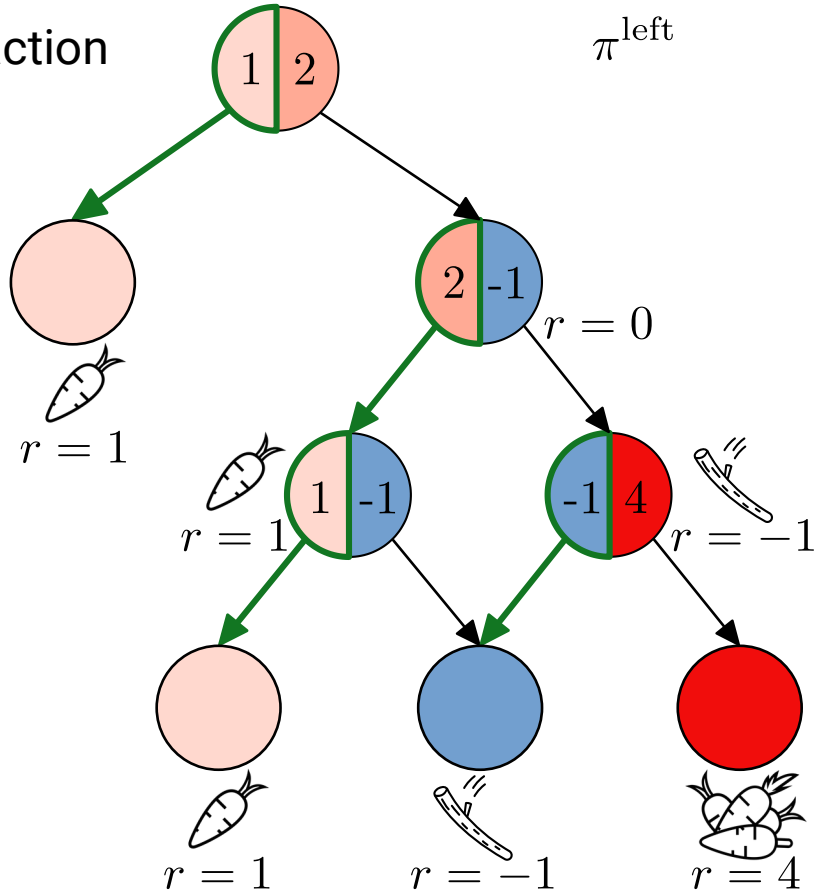
Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

- Q-function: Expected future reward sum per state-action

$$Q^{\pi}(s_t, a_t) = E_{\pi}\left[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots\right]$$

$$= E_{\pi}\left[r_t + Q^{\pi}(s_{t+1}, \cdot)\right]$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

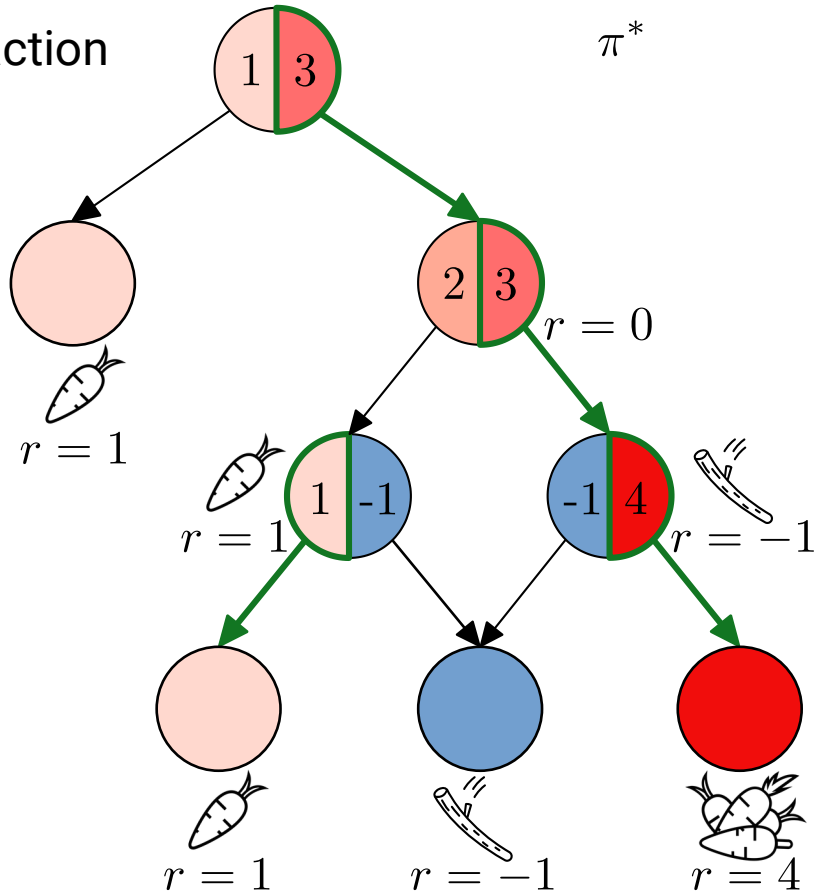
- Q-function: Expected future reward sum per state-action

$$Q^{\pi}(s_t, a_t) = E_{\pi}\left[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots\right]$$

$$= E_{\pi}\left[r_t + Q^{\pi}(s_{t+1}, \cdot)\right]$$

- Optimal Q-function:

$$Q^{*}(s_t, a_t) = E\left[r_t + \max_{a_{t+1}} Q^{\pi}(s_{t+1}, a_{t+1})\right]$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

- Q-function: Expected future reward sum per state-action

$$Q^{\pi}(s_t, a_t) = E_{\pi}\left[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots\right]$$

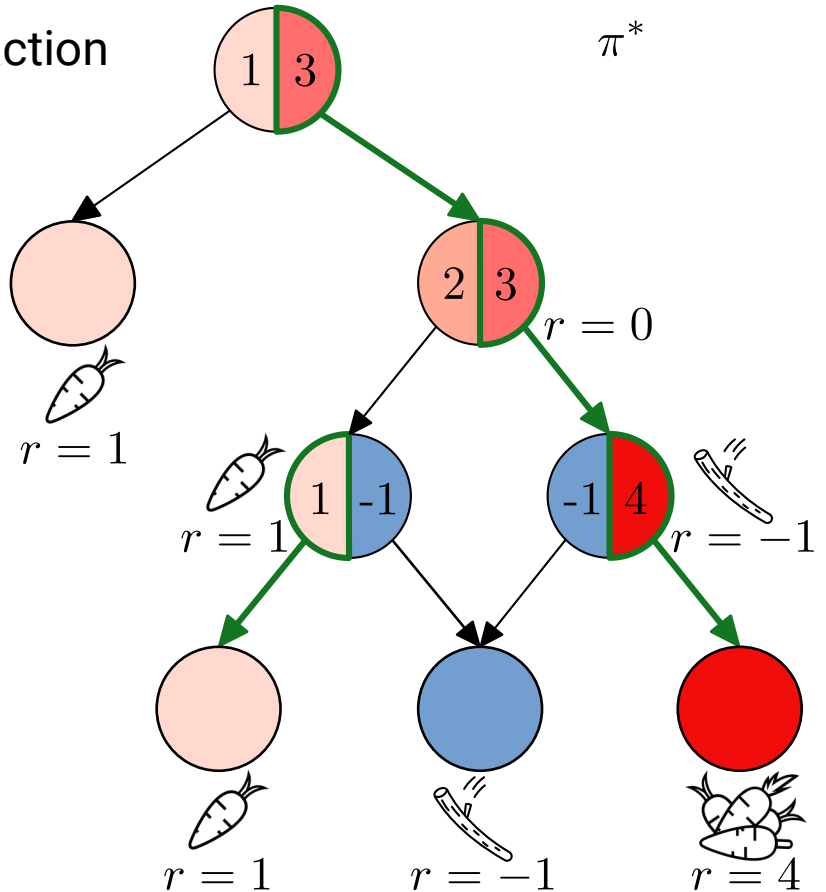
$$= E_{\pi}\left[r_t + Q^{\pi}(s_{t+1}, \cdot)\right]$$

- Optimal Q-function:

$$Q^{*}(s_t, a_t) = E\left[r_t + \max_{a_{t+1}} Q^{\pi}(s_{t+1}, a_{t+1})\right]$$

- Target policy based on Q-function:

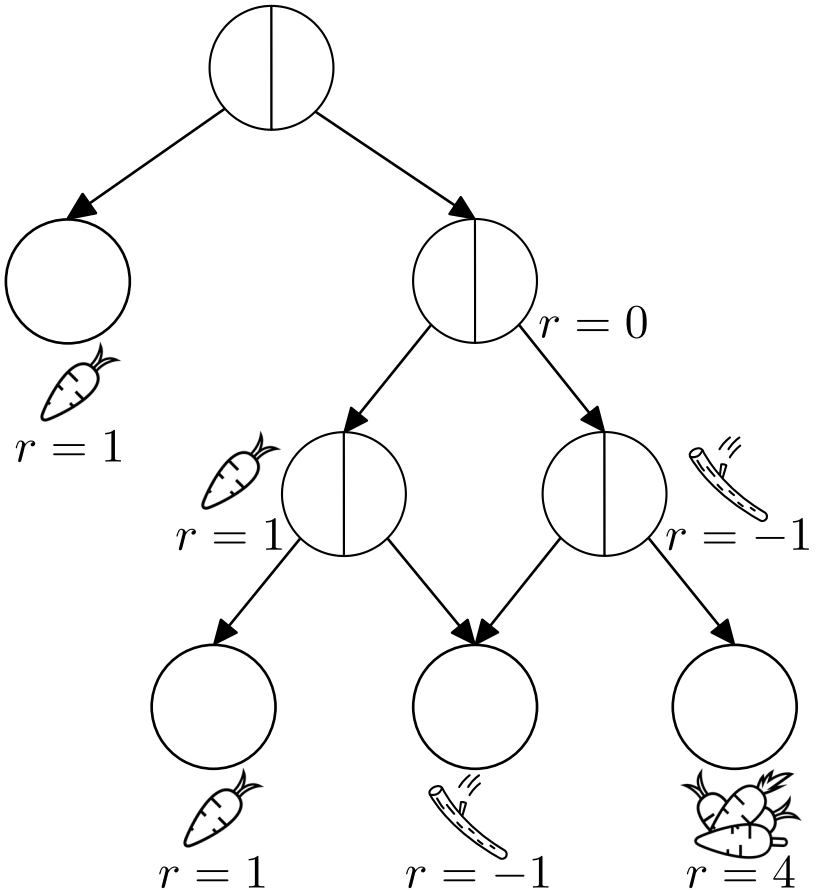
$$\pi^{*}(s_t) = \operatorname*{argmax}_{a} Q^{*}(s_t, a)$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

- Q-Learning

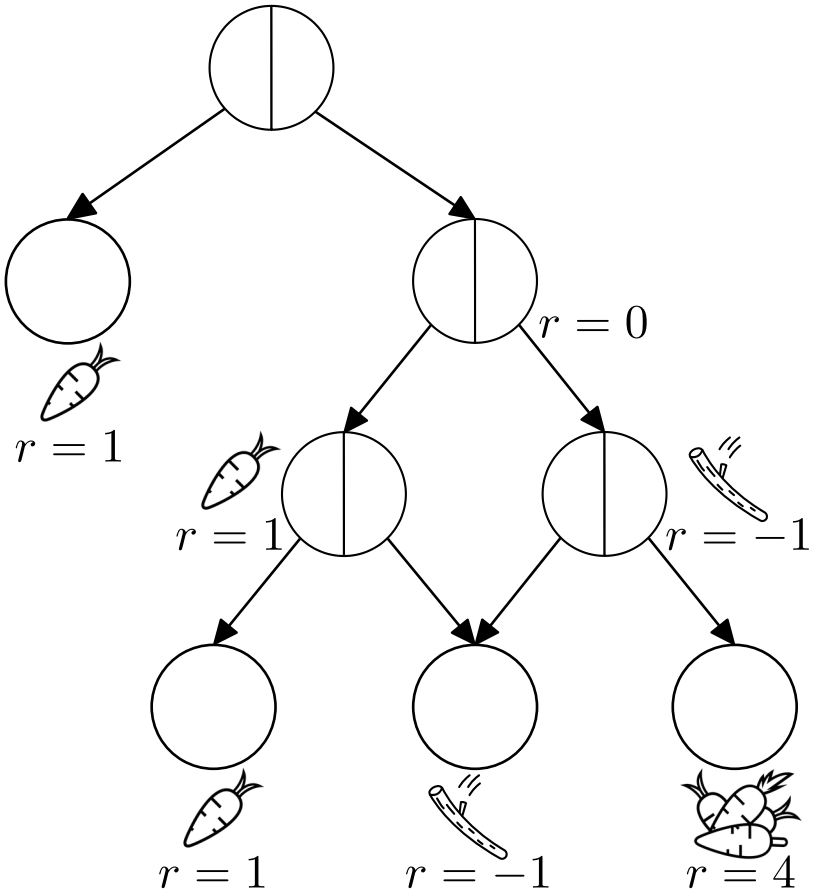


Barto & Sutton (2018) “Reinforcement learning: An introduction”

Reinforcement Learning

- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

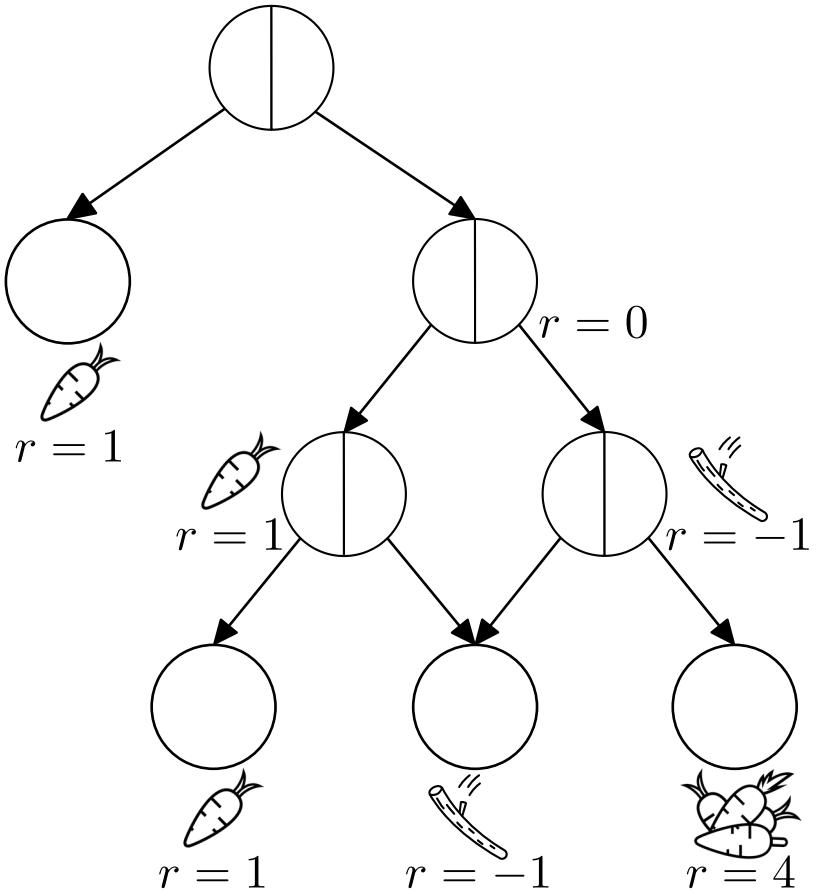
$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

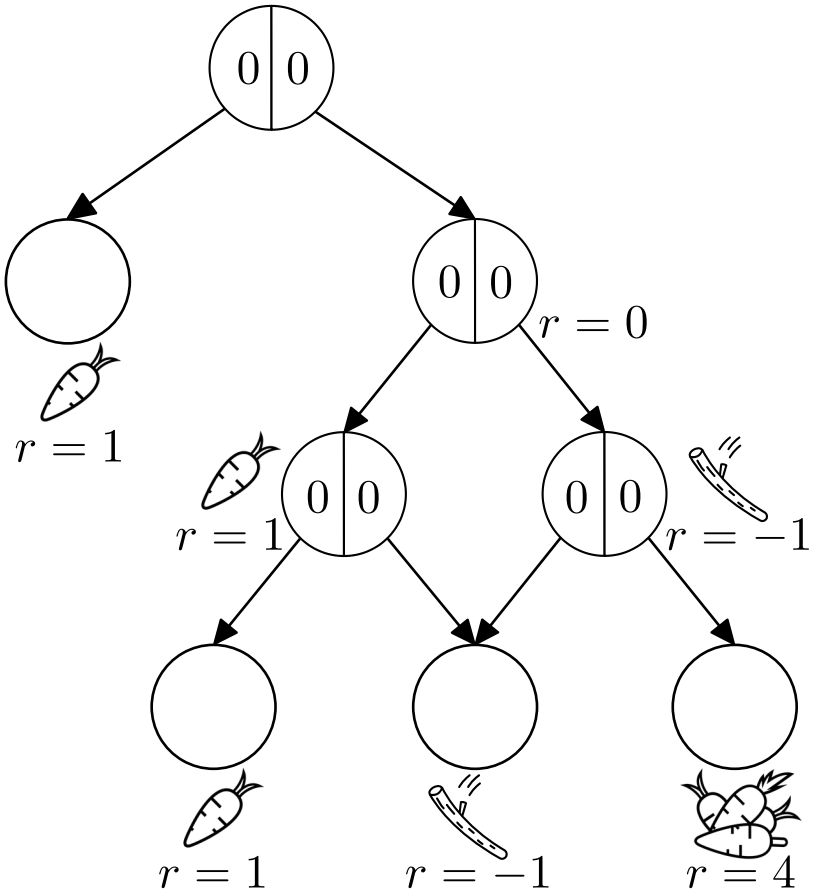
$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

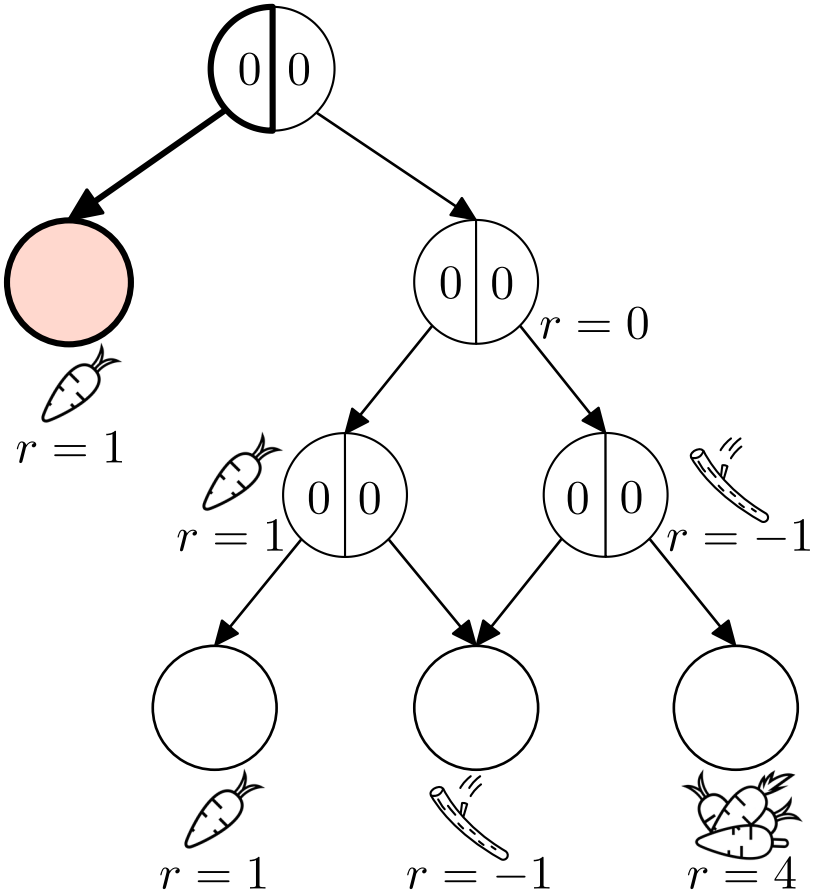
$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

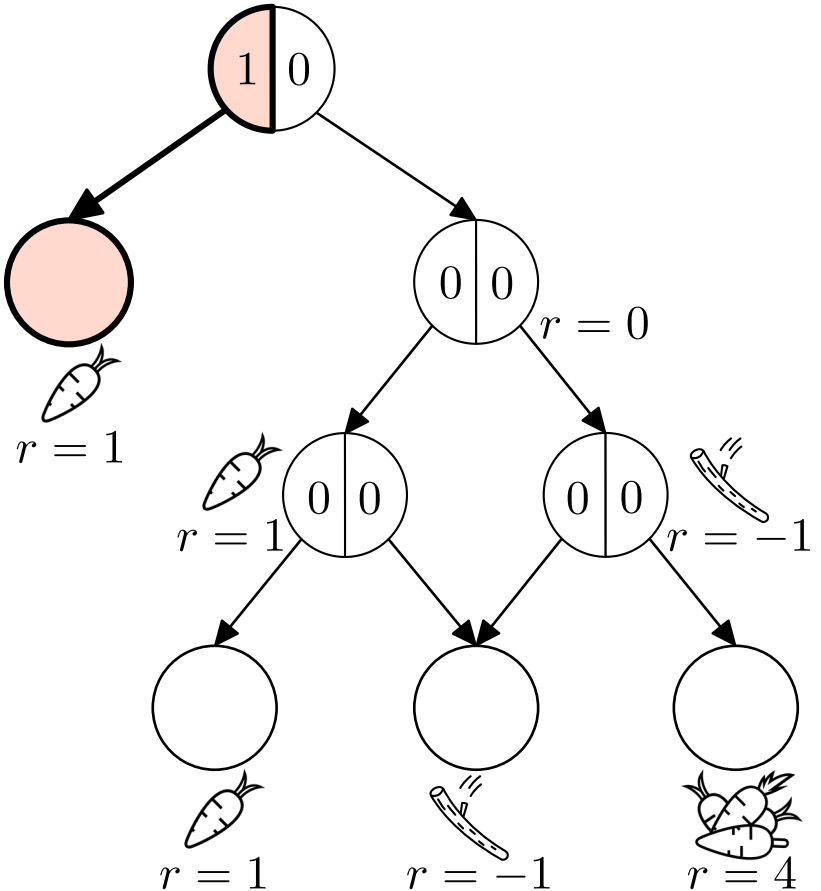
$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

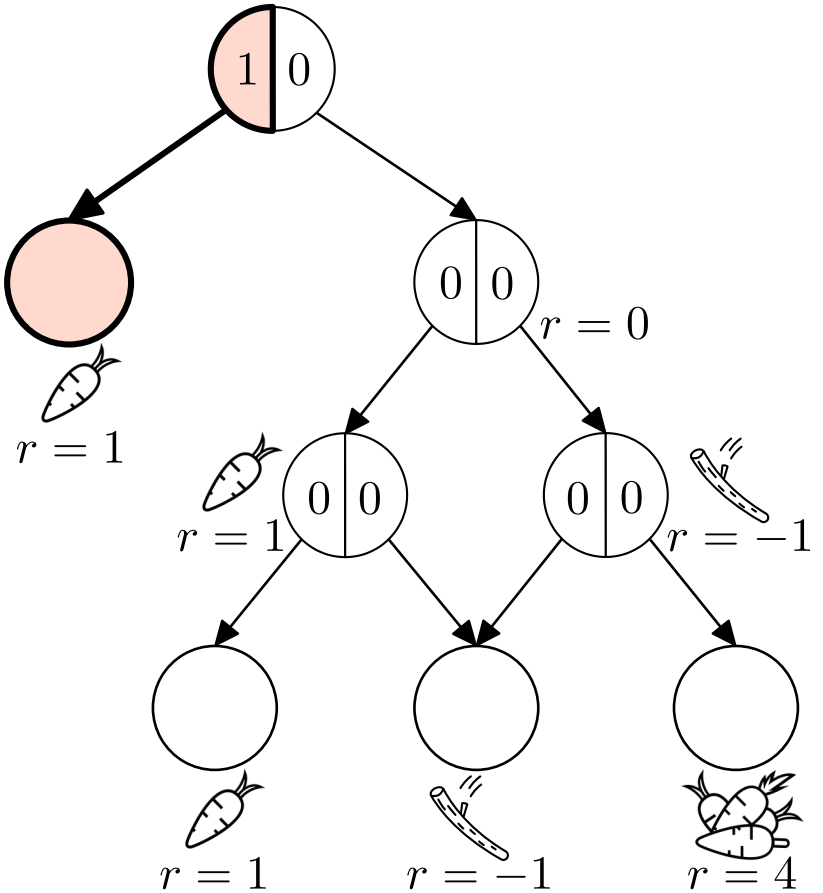
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & | \text{ with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & | \text{ with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

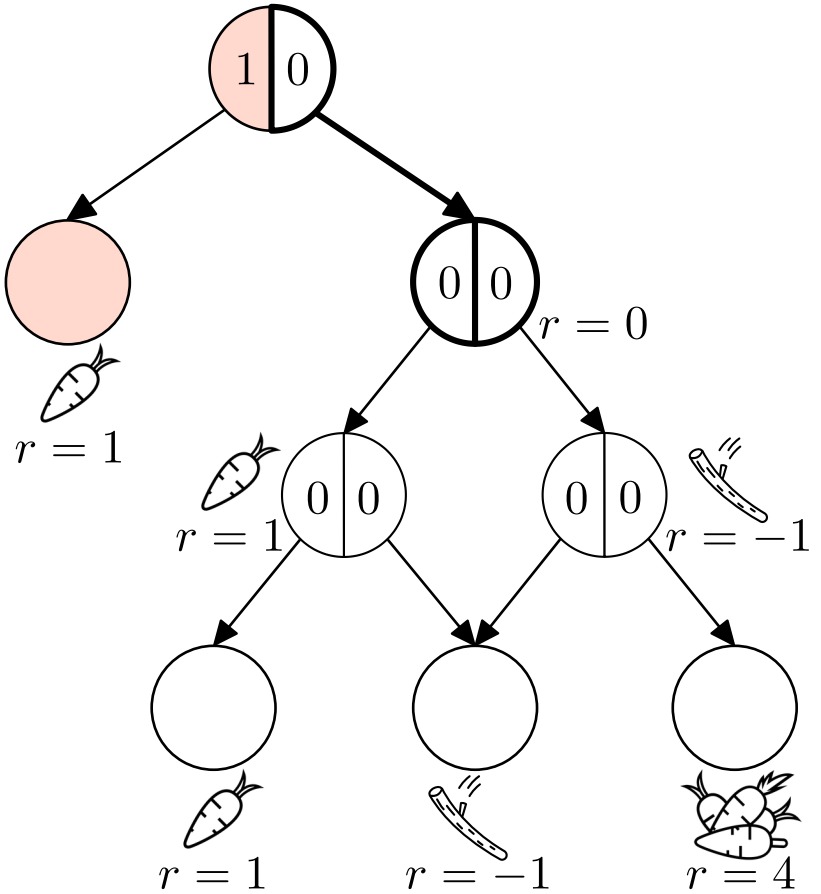
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & | \text{ with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & | \text{ with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

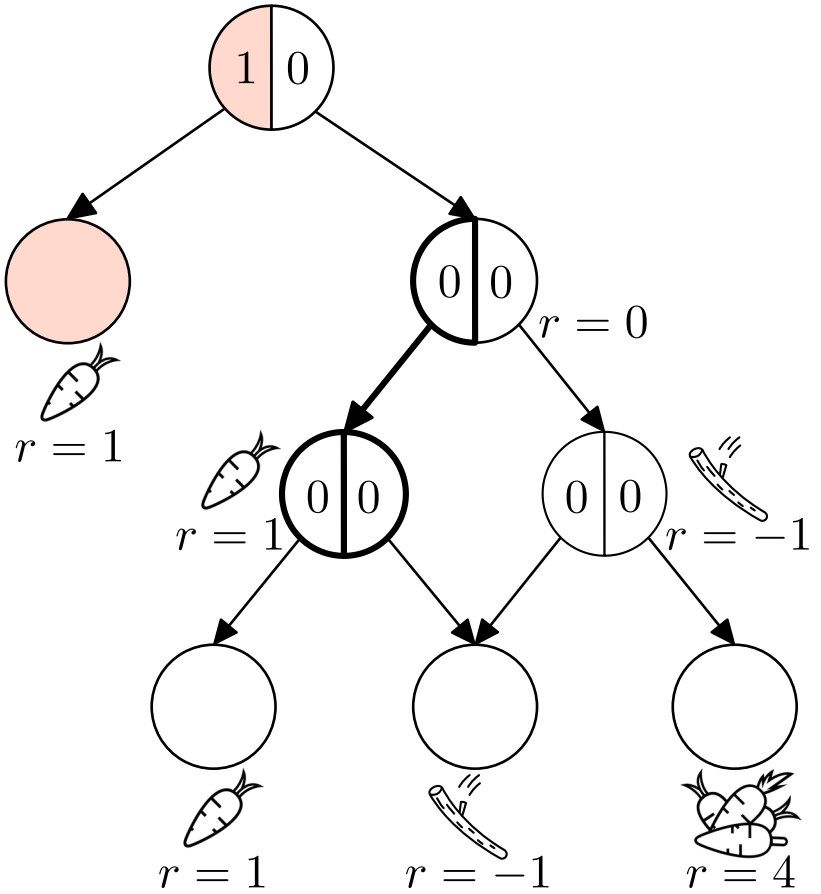
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & \mid \text{with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & \mid \text{with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

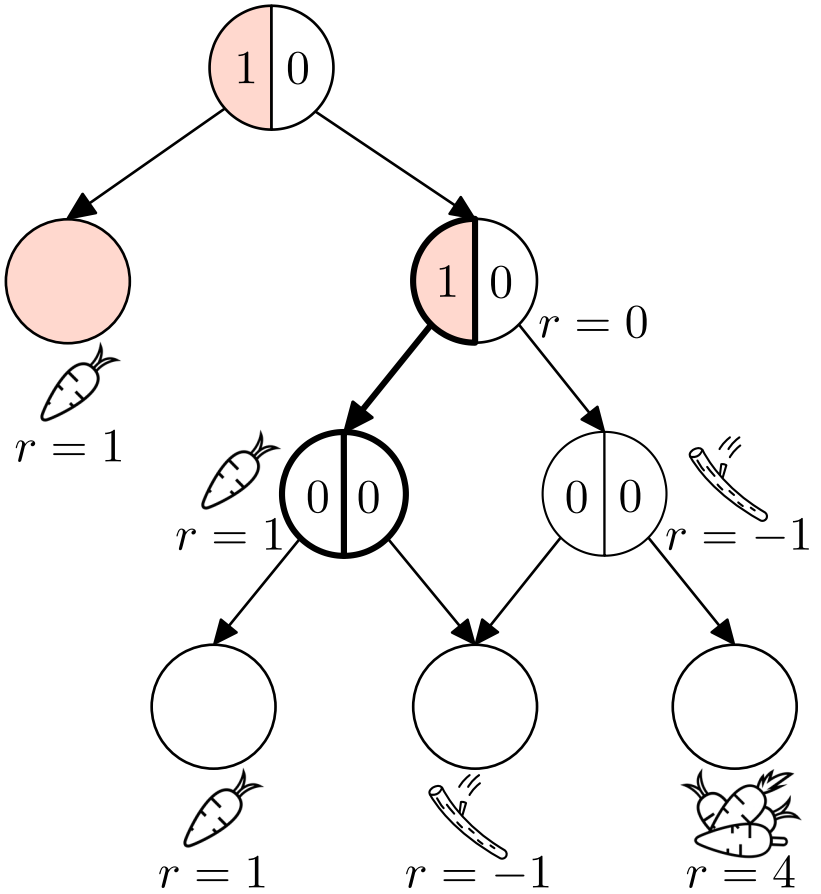
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & | \text{ with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & | \text{ with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

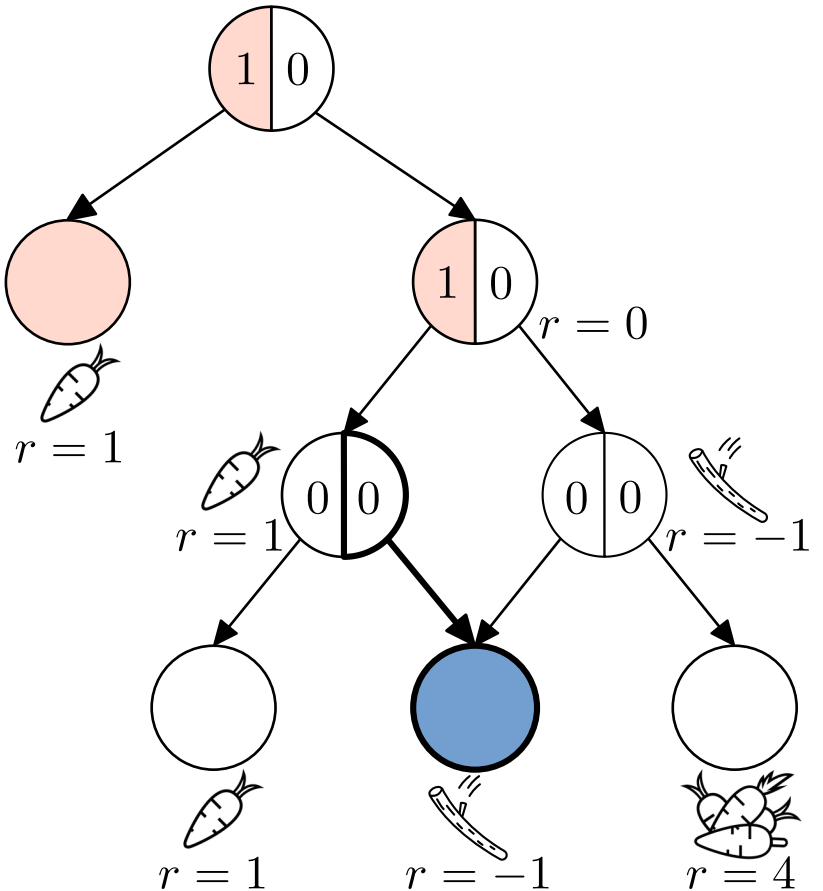
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & \mid \text{with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & \mid \text{with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

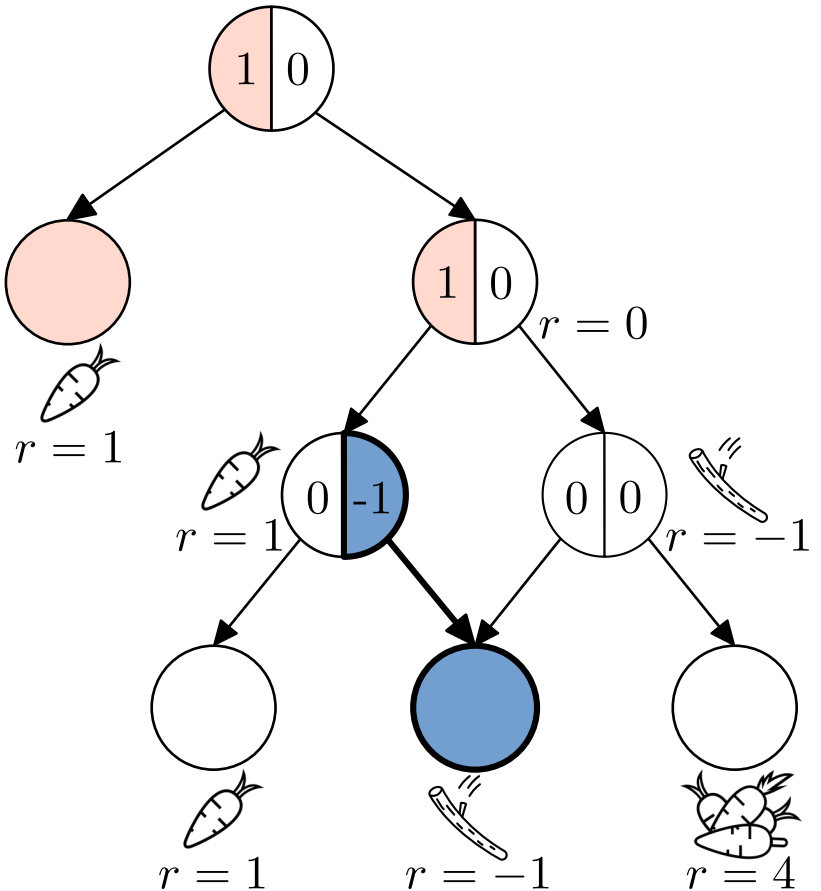
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & \mid \text{with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & \mid \text{with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

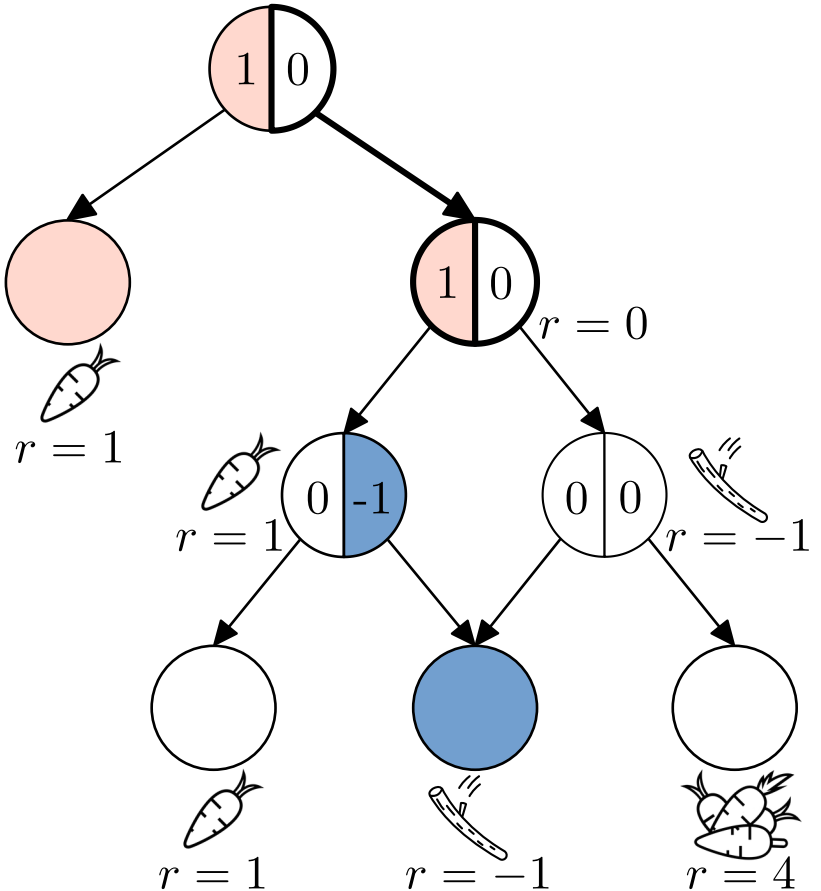
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & | \text{ with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & | \text{ with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

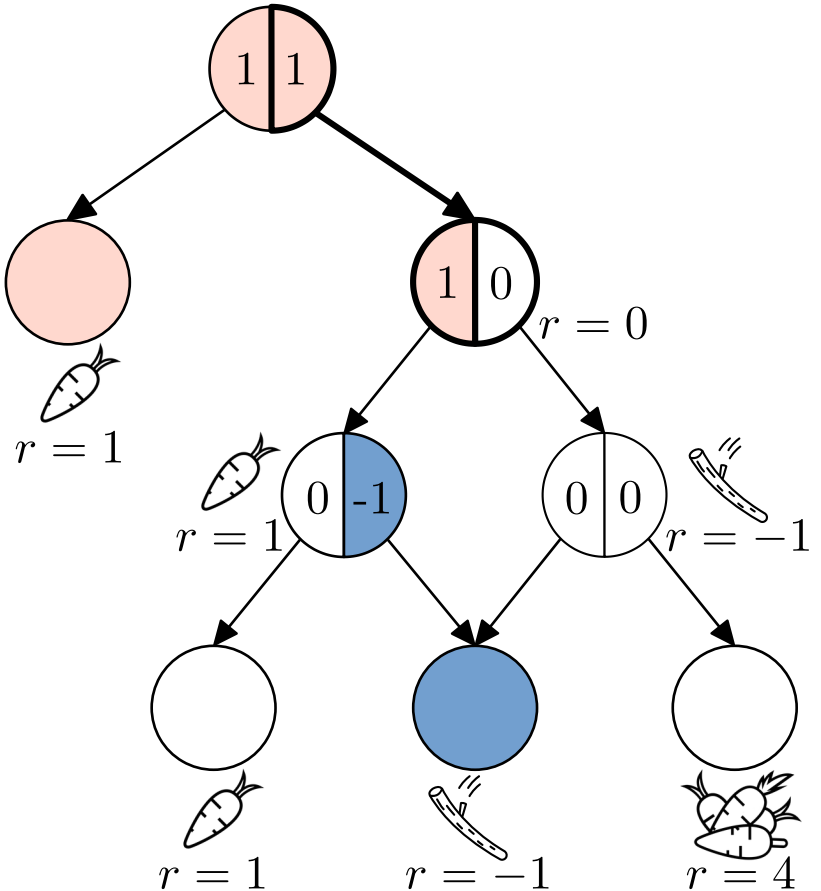
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & | \text{ with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & | \text{ with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

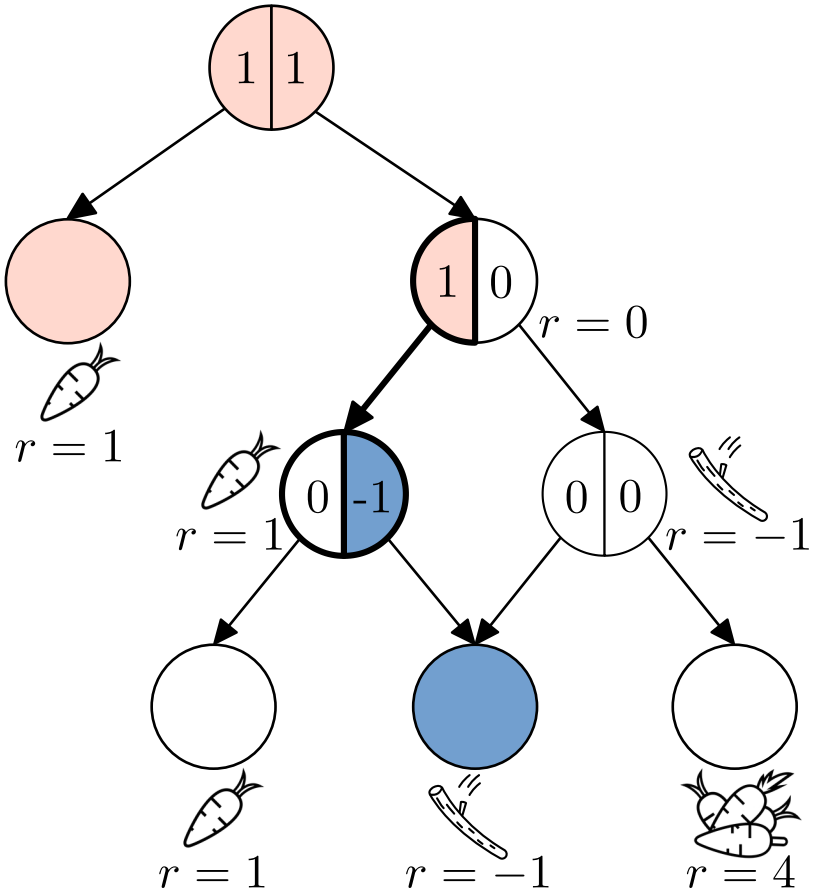
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & \mid \text{with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & \mid \text{with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

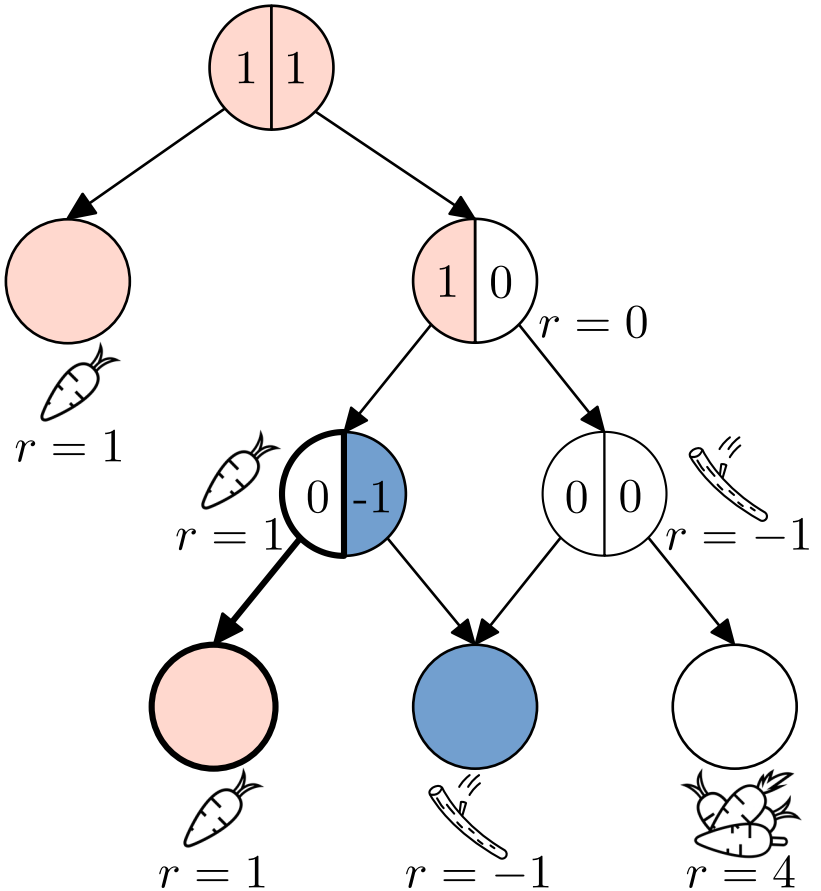
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & | \text{ with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & | \text{ with probability } 1-\epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

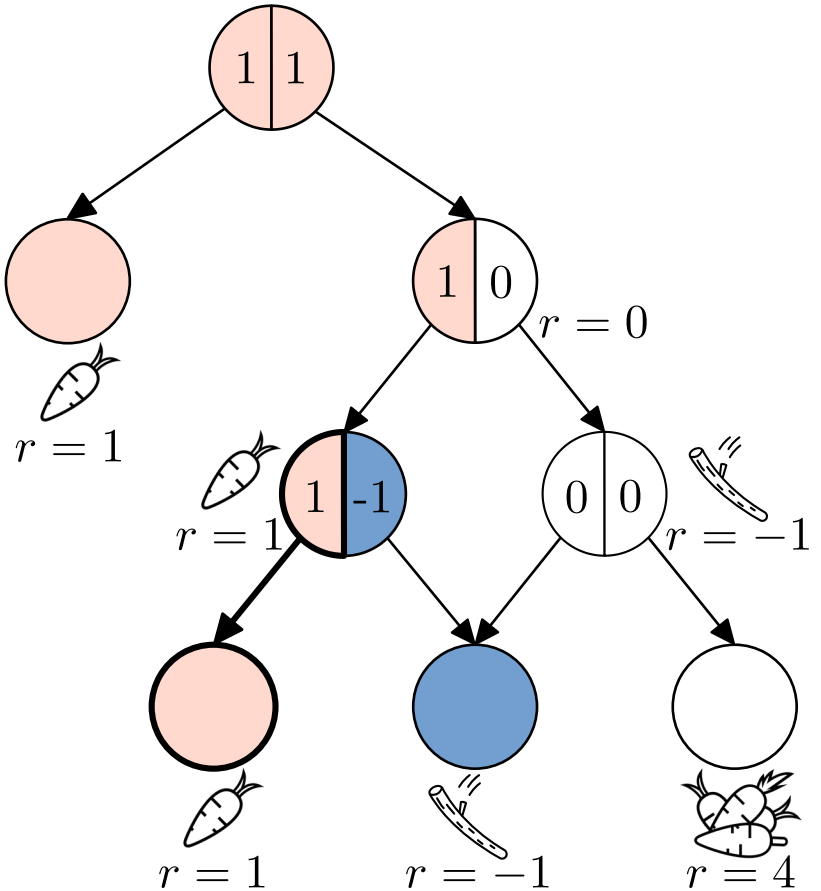
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & | \text{ with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & | \text{ with probability } 1 - \epsilon \end{cases}$$



Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Reinforcement Learning

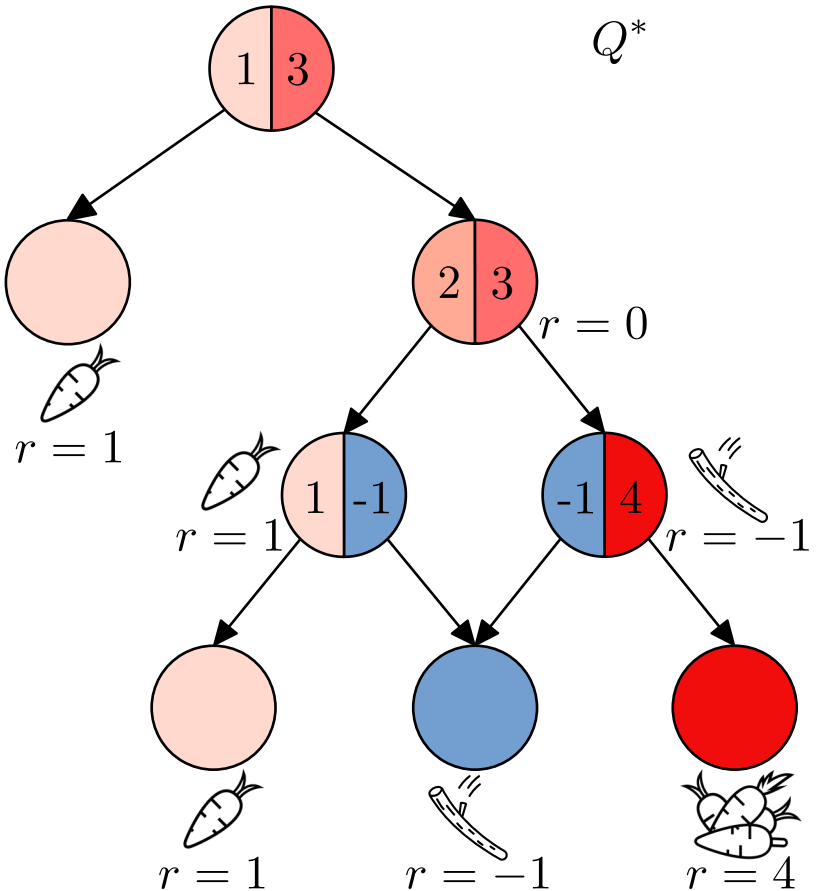
- Q-Learning: Based on observation (s_t, a_t, r_t, s_{t+1})

$$Q_{k+1}(s_t, a_t) \leftarrow Q_k(s_t, a_t) + \alpha\delta_Q$$

$$\delta_Q = \underbrace{r_t + \max_{a_{t+1}} Q_k(s_{t+1}, a_{t+1}))}_{\text{observation}} - \underbrace{Q_k(s_t, a_t)}_{\text{prediction}}$$

- Behavior policy allows exploration:

$$\pi^\epsilon(s_t) = \begin{cases} \text{random action} & | \text{ with probability } \epsilon \\ \underset{a}{\text{argmax}}\ Q(s_t, a) & | \text{ with probability } 1-\epsilon \end{cases}$$



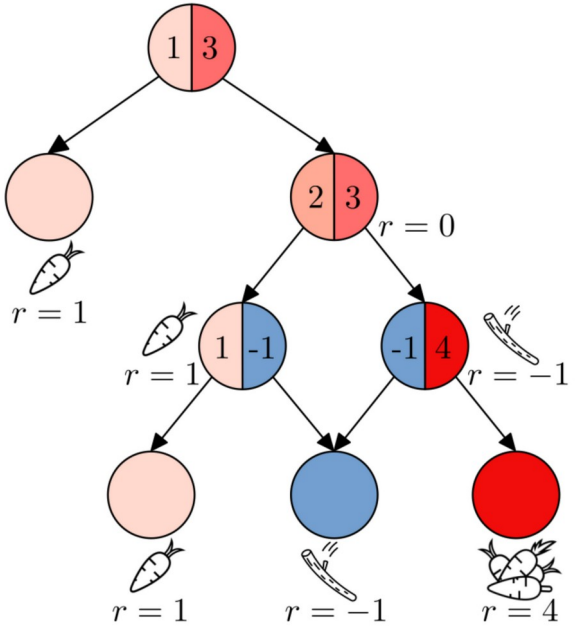
Barto & Sutton (2018) “Reinforcement learning: An introduction”

$\alpha \in [0, 1]$ - Learning rate

Deep Reinforcement Learning (DRL)

Reinforcement Learning with Deep Neural Networks.

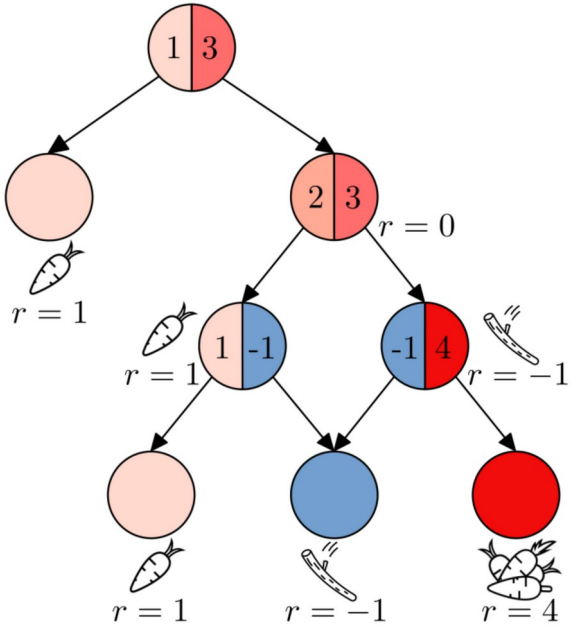
Deep Reinforcement Learning



Q-Table:

Q	s_1	s_3	s_4	s_5
a_{left}	1	2	1	-1
a_{right}	3	3	-1	4

Deep Reinforcement Learning



$\approx 10^{46}$ states

High-dimensional &
Continuous states

Q-Table:

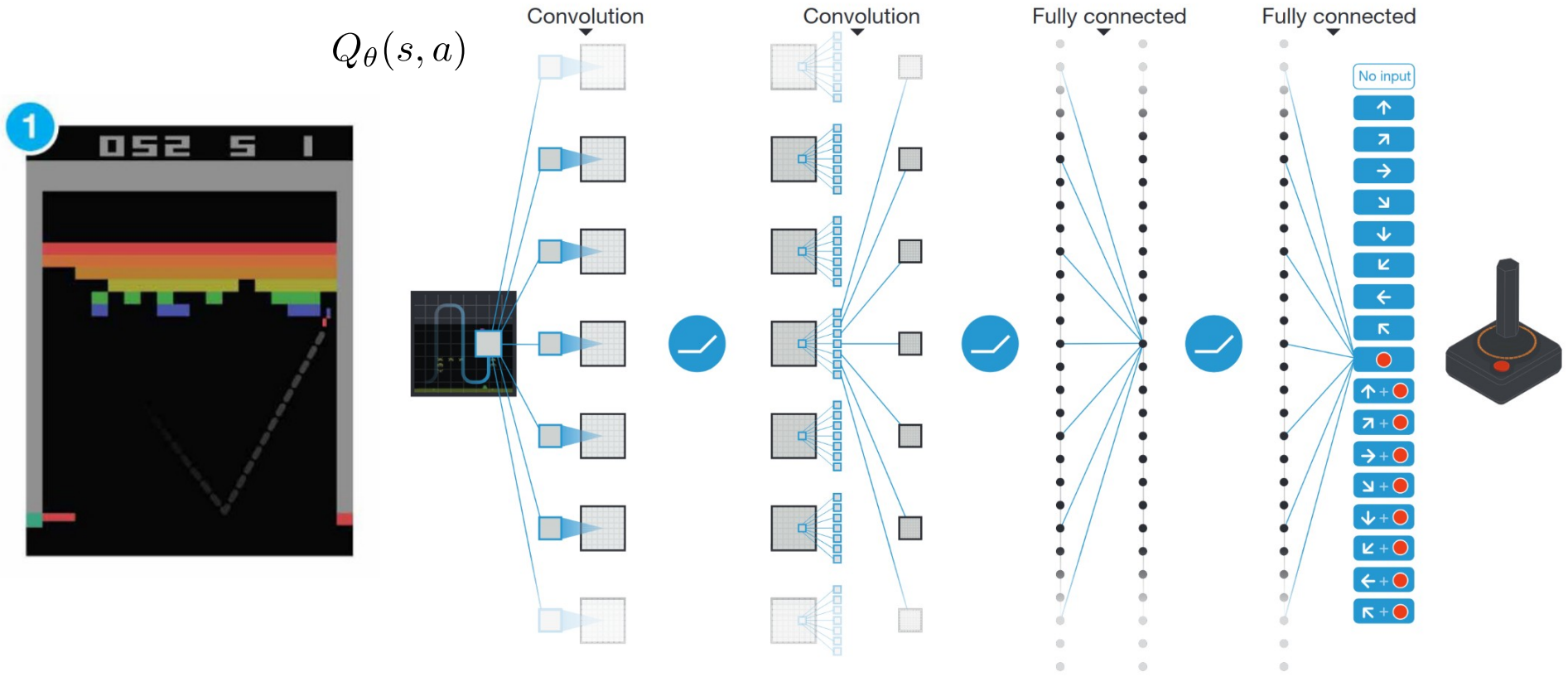
Q	s_1	s_3	s_4	s_5
a_{left}	1	2	1	-1
a_{right}	3	3	-1	4

→ How to represent the Q-function?

Deep Q-Learning

Use Deep Neural Networks (DNNs) for Q-function

$Q_\theta(s, a)$



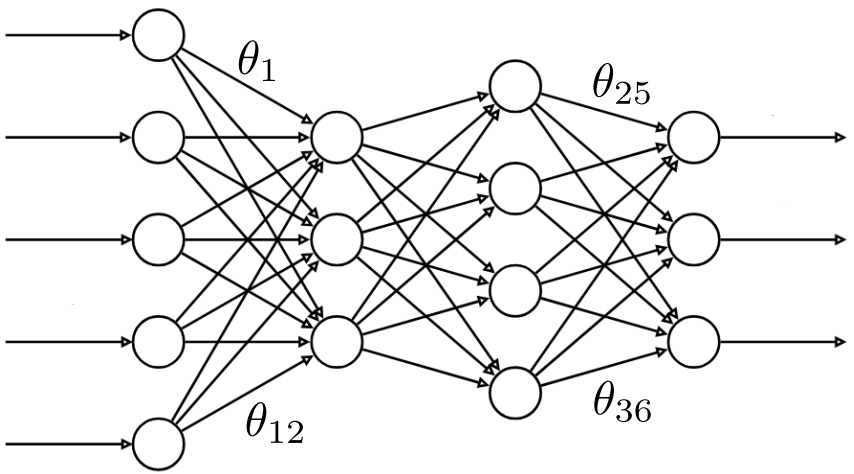
Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Deep Q-Learning

How to train the network ?

Given a transition $b = (s_t, a_t, r_t, s_{t+1})$ use Mean Squared Error (MSE) loss:

$$Q_\theta(s, a) \quad | \quad \theta = \{\theta_1, \ldots, \theta_{36}\}$$



$$\mathcal{L}_\theta = \frac{1}{2}\left(\underbrace{r_t + \max_{a_{t+1}} Q_{\bar{\theta}}(s_{t+1}, a_{t+1})}_{\text{observation / target}} - \underbrace{Q_\theta(s_t, a_t)}_{\text{prediction}}\right)^2$$

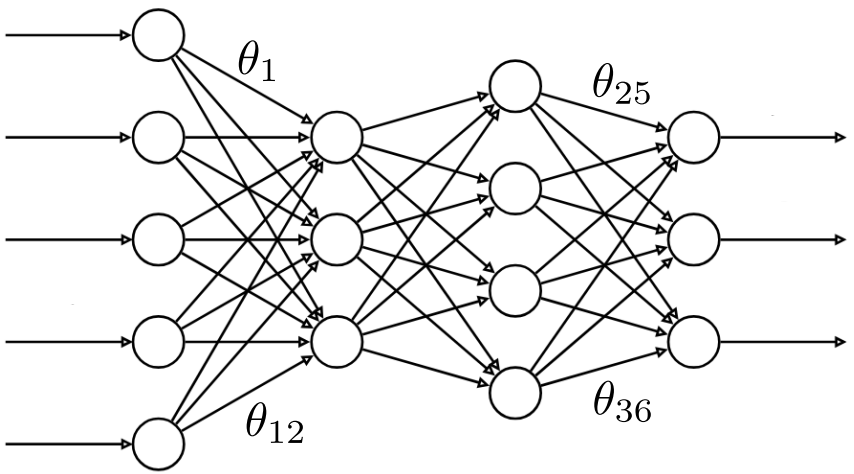
Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Deep Q-Learning

How to train the network ?

Given a transition $b = (s_t, a_t, r_t, s_{t+1})$ use Mean Squared Error (MSE) loss:

$$Q_\theta(s, a) \quad | \quad \theta = \{\theta_1, \ldots, \theta_{36}\}$$

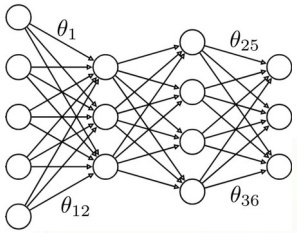


$$\mathcal{L}_\theta = \frac{1}{2}\left(\underbrace{r_t + \max_{a_{t+1}} Q_{\bar{\theta}}(s_{t+1}, a_{t+1})}_{\text{observation / target}} - \underbrace{Q_\theta(s_t, a_t)}_{\text{prediction}}\right)^2$$

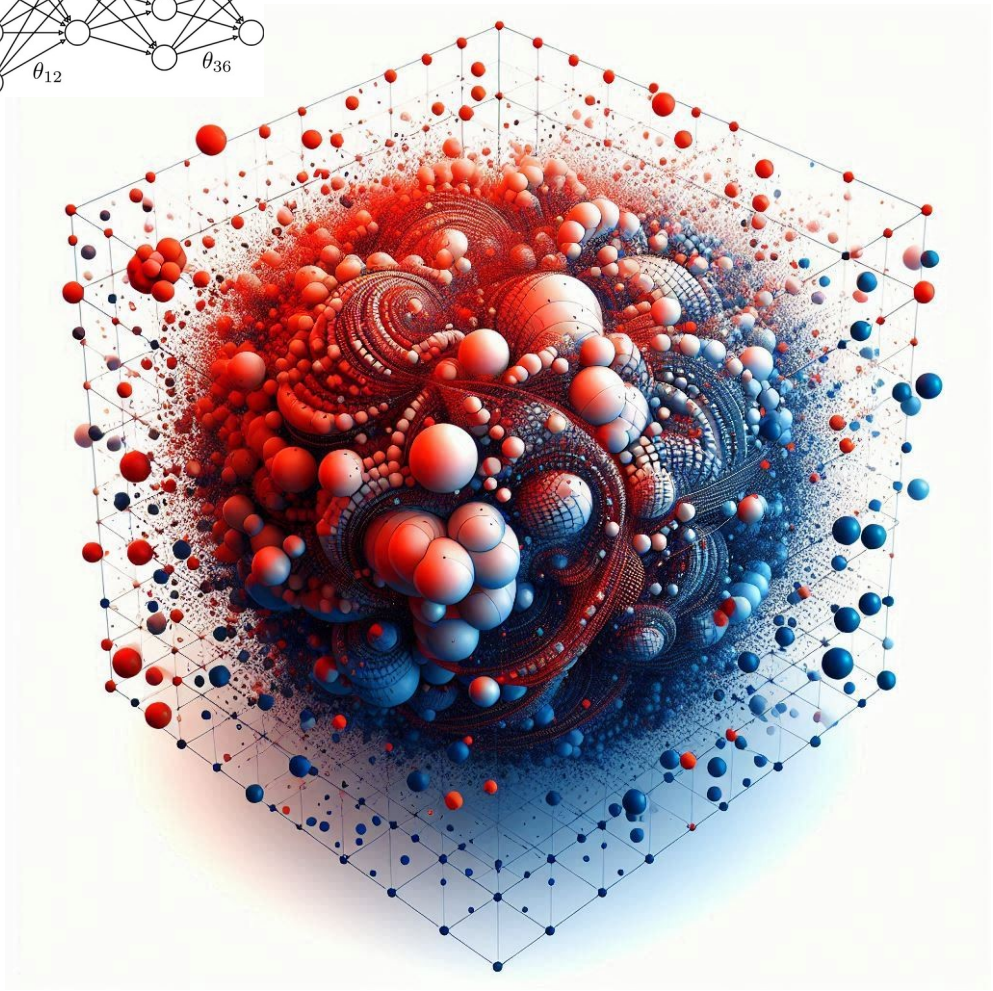
Important techniques

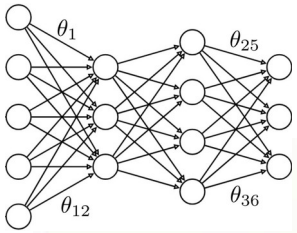
- Experience Replay Buffer
- Target Network

Mnih et al. (2015) “Human-level control through deep reinforcement learning”

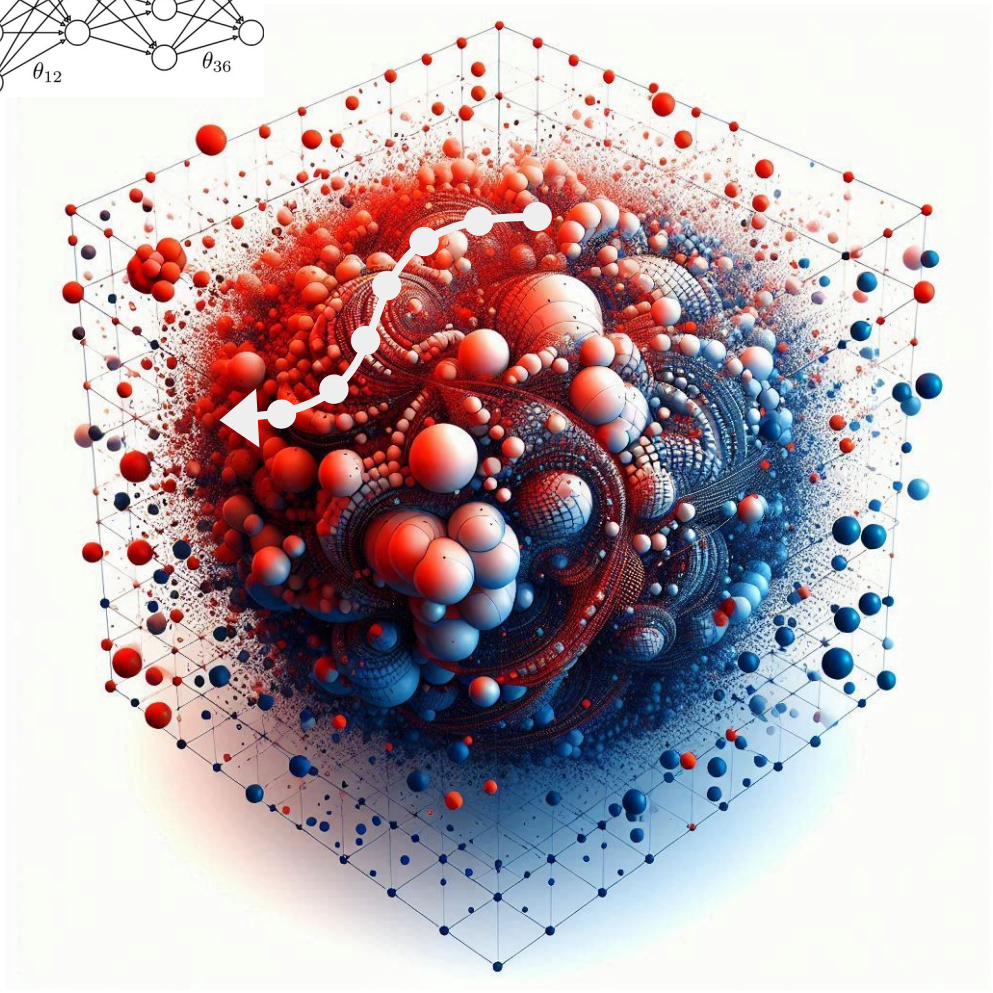


Experience Replay Buffer

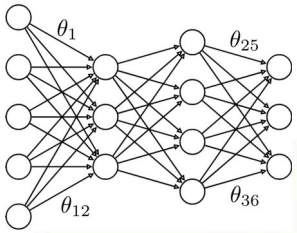


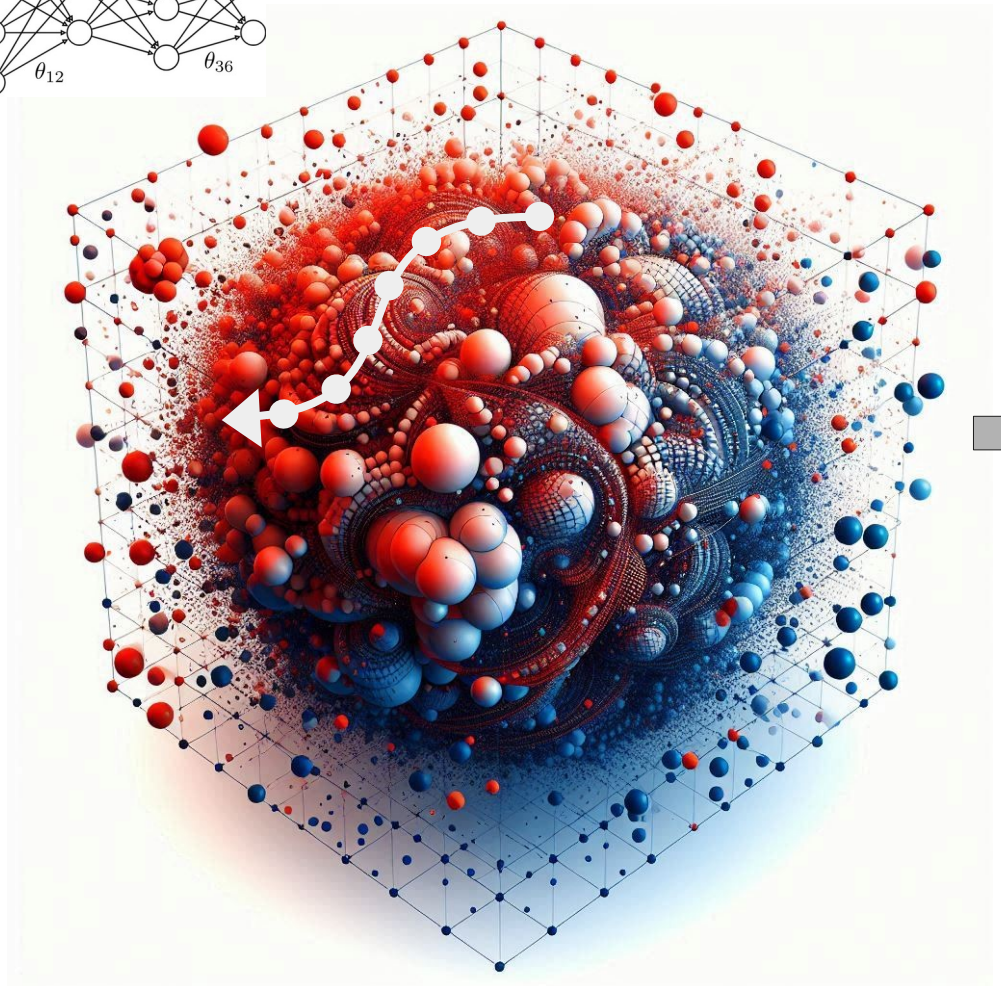


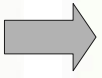
Experience Replay Buffer

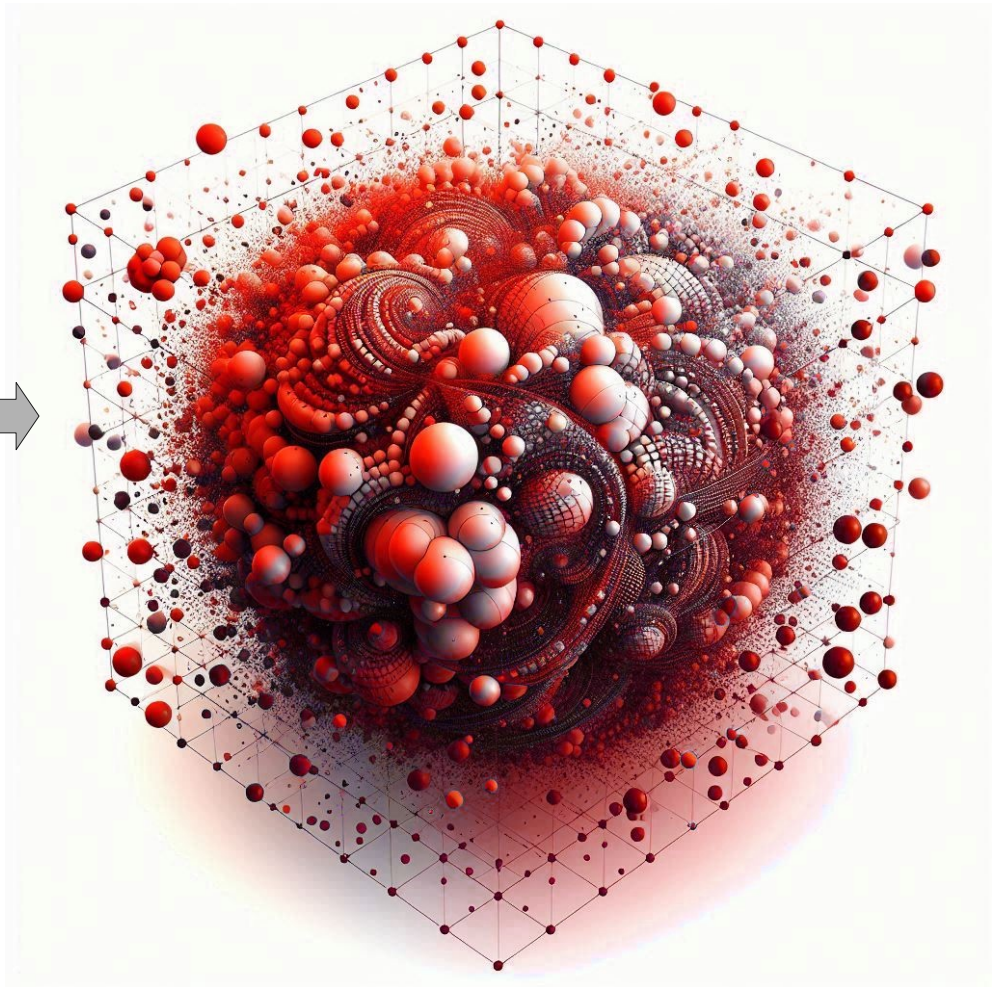


Experience Replay Buffer

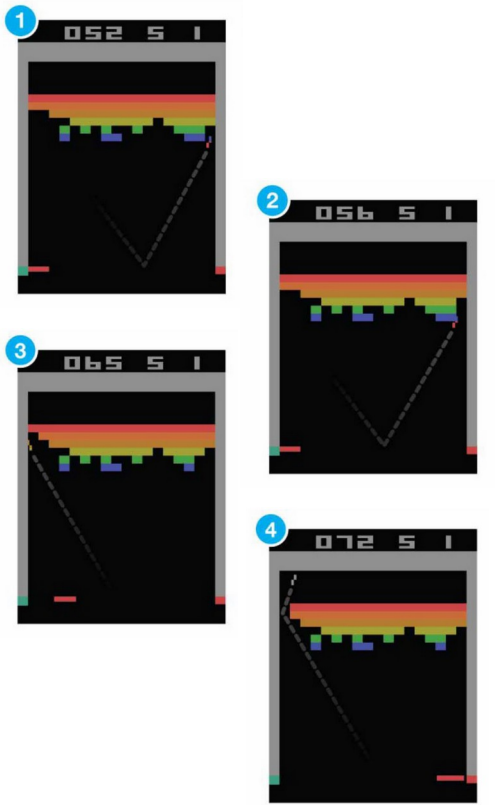








Experience Replay Buffer



collect

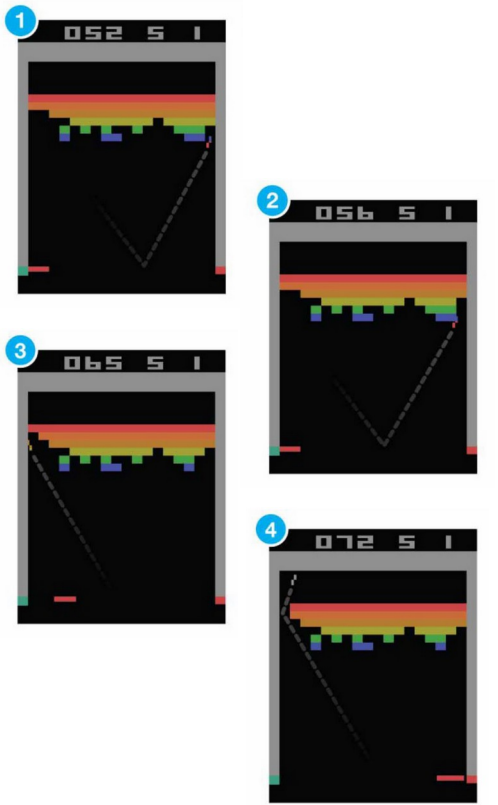
Buffer

b_1
b_2
b_3
b_4
b_5
b_6
$\ldots$

$$b_i = (s_t, a_t, r_t, s_{t+1})$$

Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Experience Replay Buffer



collect

Buffer

b_1
b_2
b_3
b_4
b_5
b_6
$\dots$

sample

Batch update

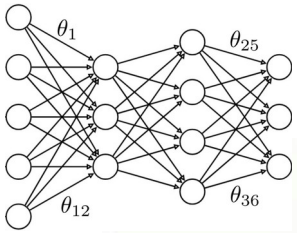
$$B = \{b_2, b_{12}, b_{133}, \dots\}$$

$$b_i = (s_t, a_t, r_t, s_{t+1})$$

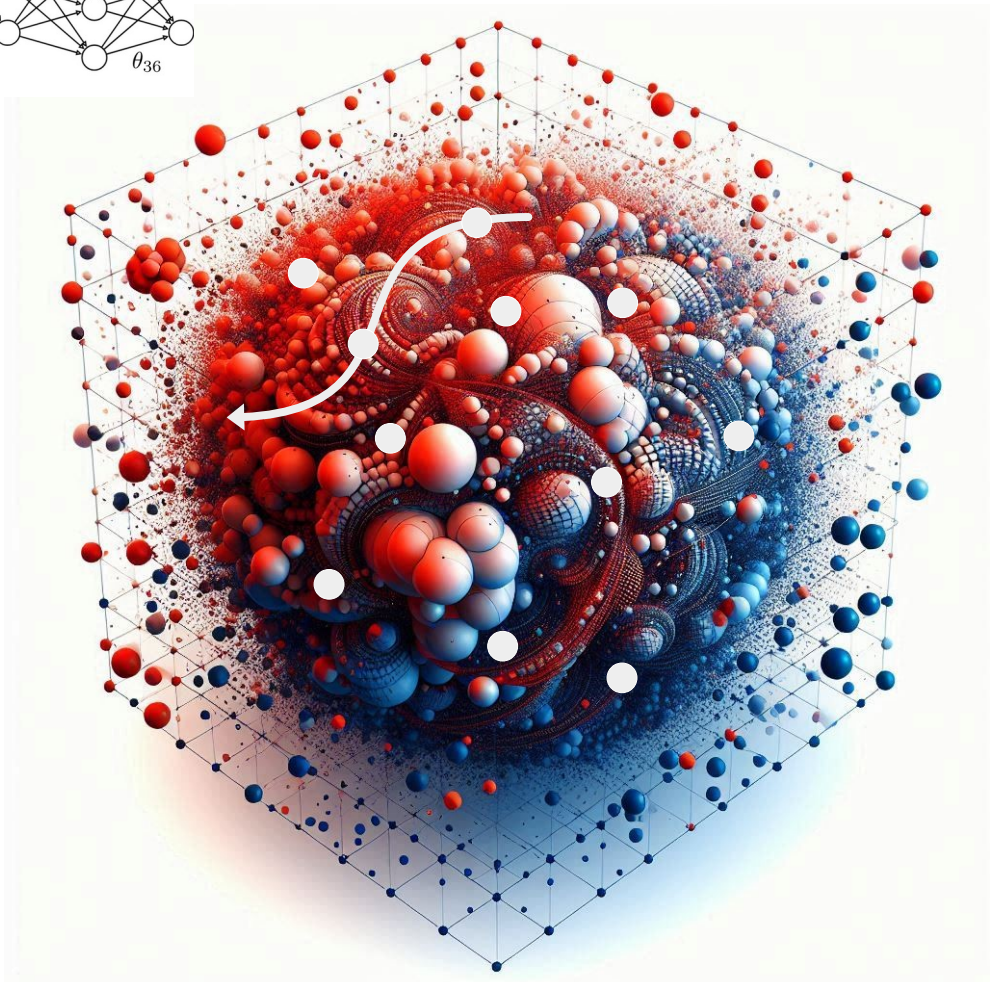
$$\mathcal{L} = \frac{1}{|B|} \sum_i \mathcal{L}_\theta(b_i)$$

$$\mathcal{L}_\theta = \frac{1}{2} \left(r_t + \max_{a_{t+1}} Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t) \right)^2$$

Mnih et al. (2015) “Human-level control through deep reinforcement learning”



Experience Replay Buffer



Batch update

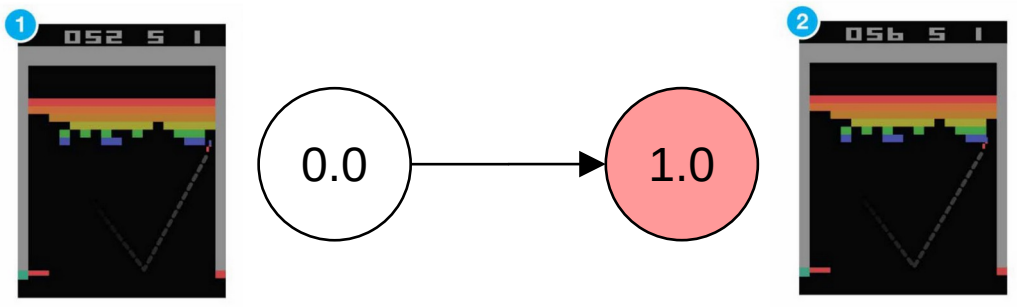
$$B = \{b_2, b_{12}, b_{133}, \ldots\}$$

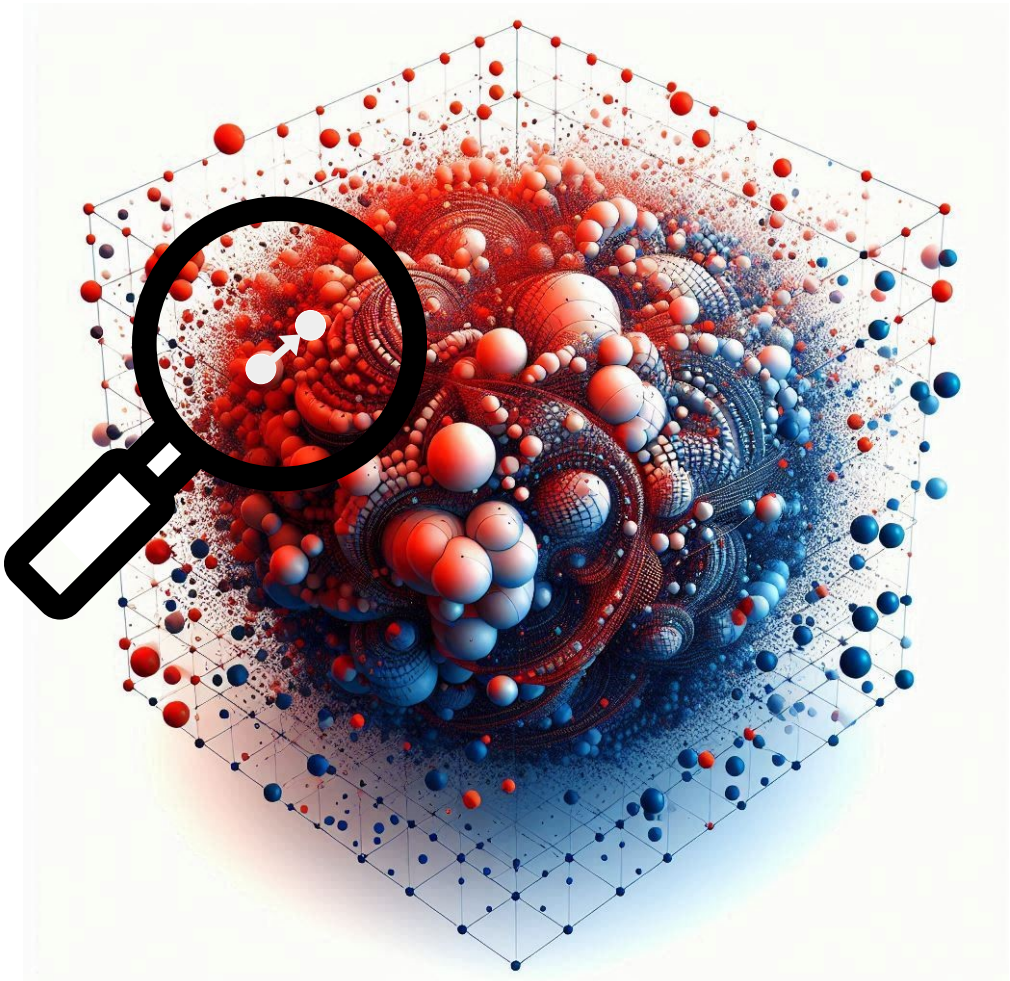
$$\mathcal{L} = \frac{1}{|B|} \sum_i \mathcal{L}_\theta(b_i)$$

$$\mathcal{L}_\theta = \frac{1}{2} \left(r_t + \max_{a_{t+1}} Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t) \right)^2$$

Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Target Network – Double Q-Learning

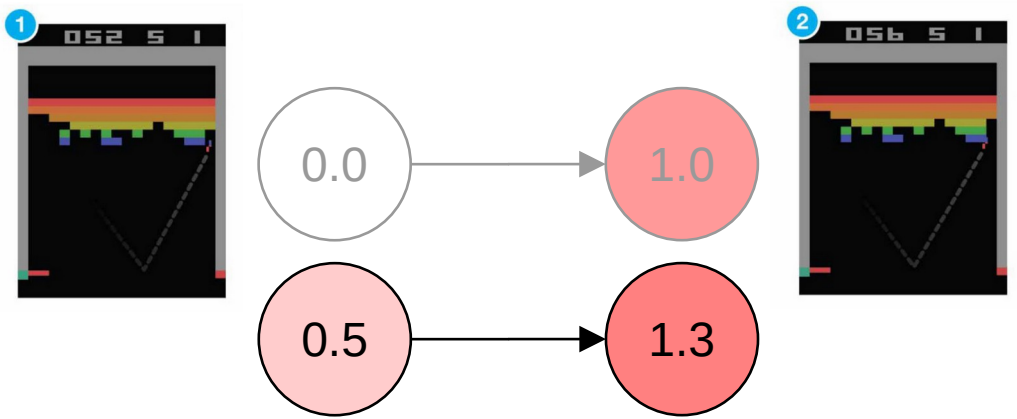


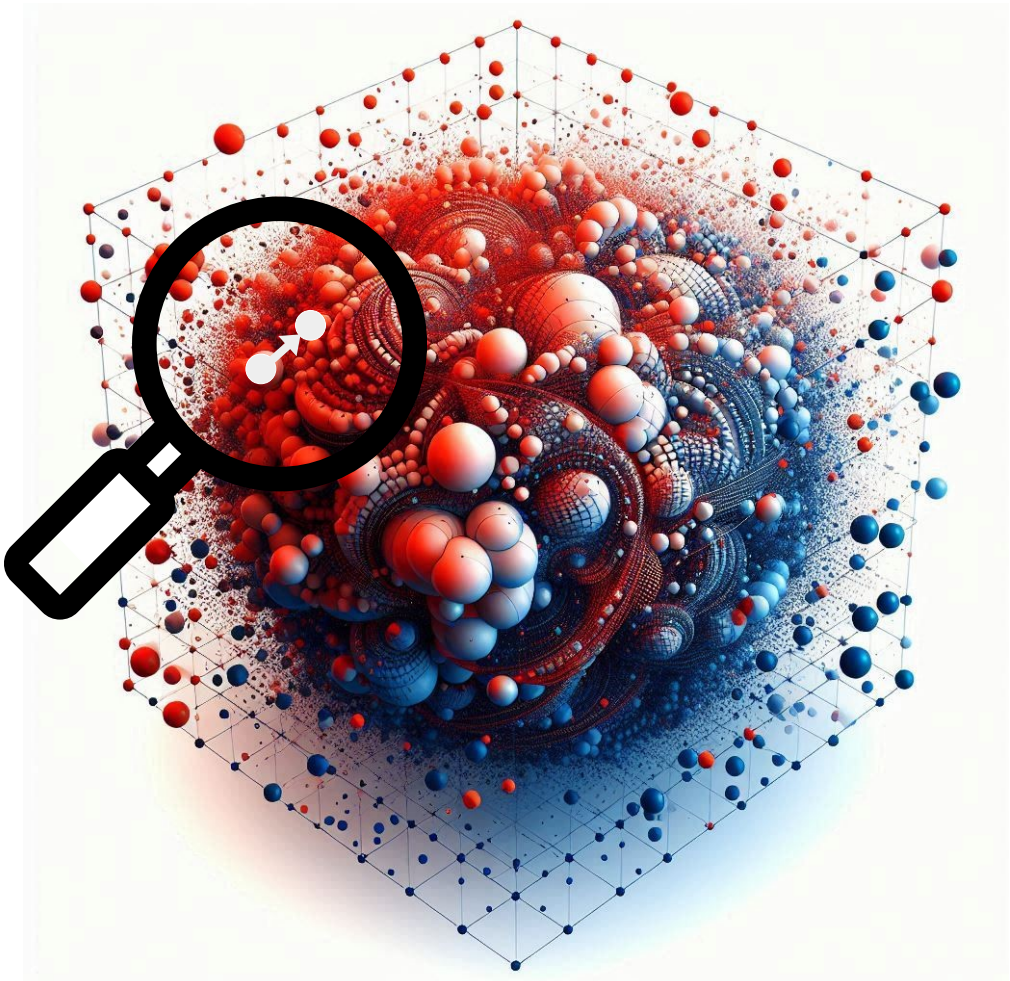


$$\mathcal{L}_\theta = \frac{1}{2}\left(r_t + \max_{a_{t+1}} Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)\right)^2$$

Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Target Network – Double Q-Learning

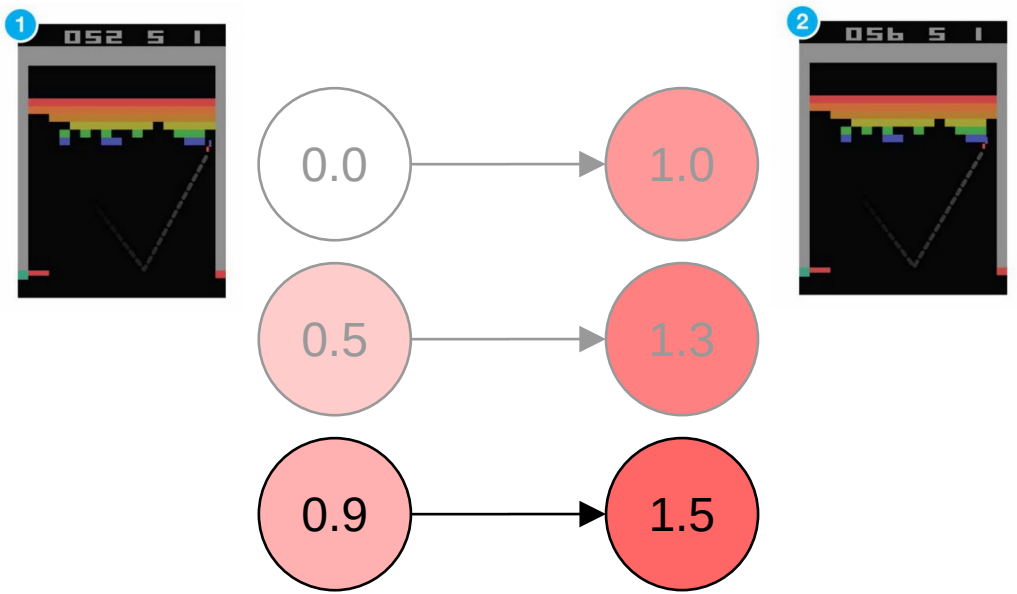


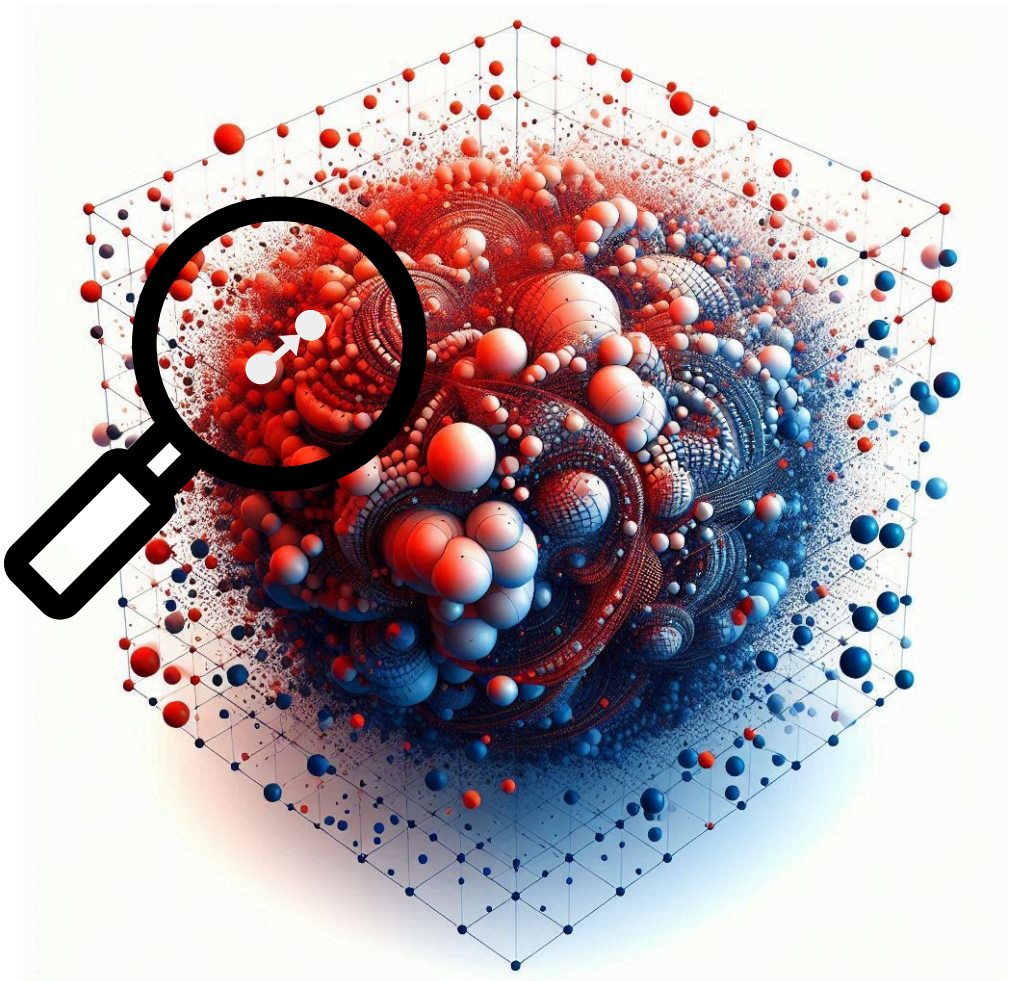


$$\mathcal{L}_\theta = \frac{1}{2}\left(r_t + \max_{a_{t+1}} Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)\right)^2$$

Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Target Network – Double Q-Learning

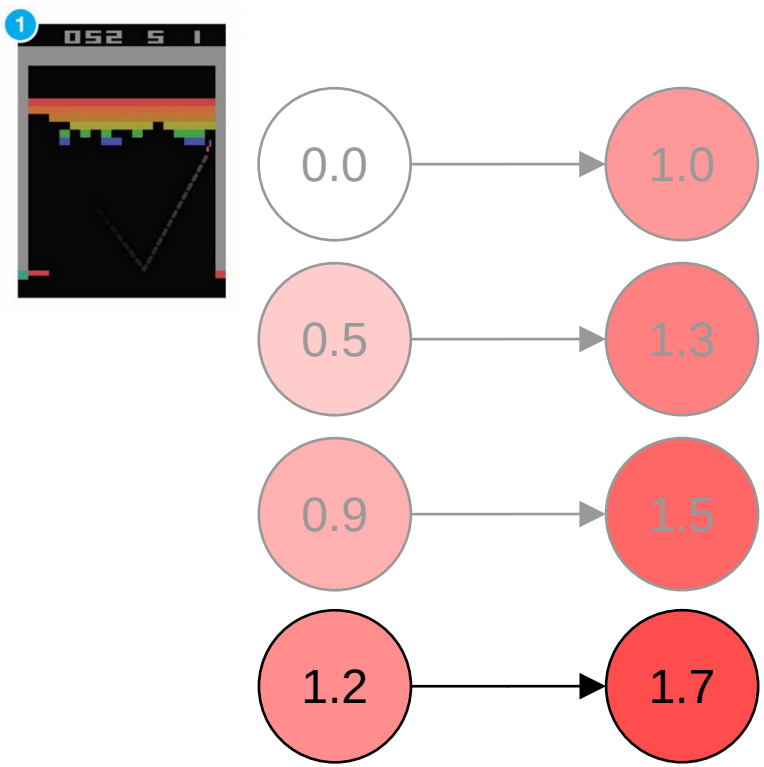




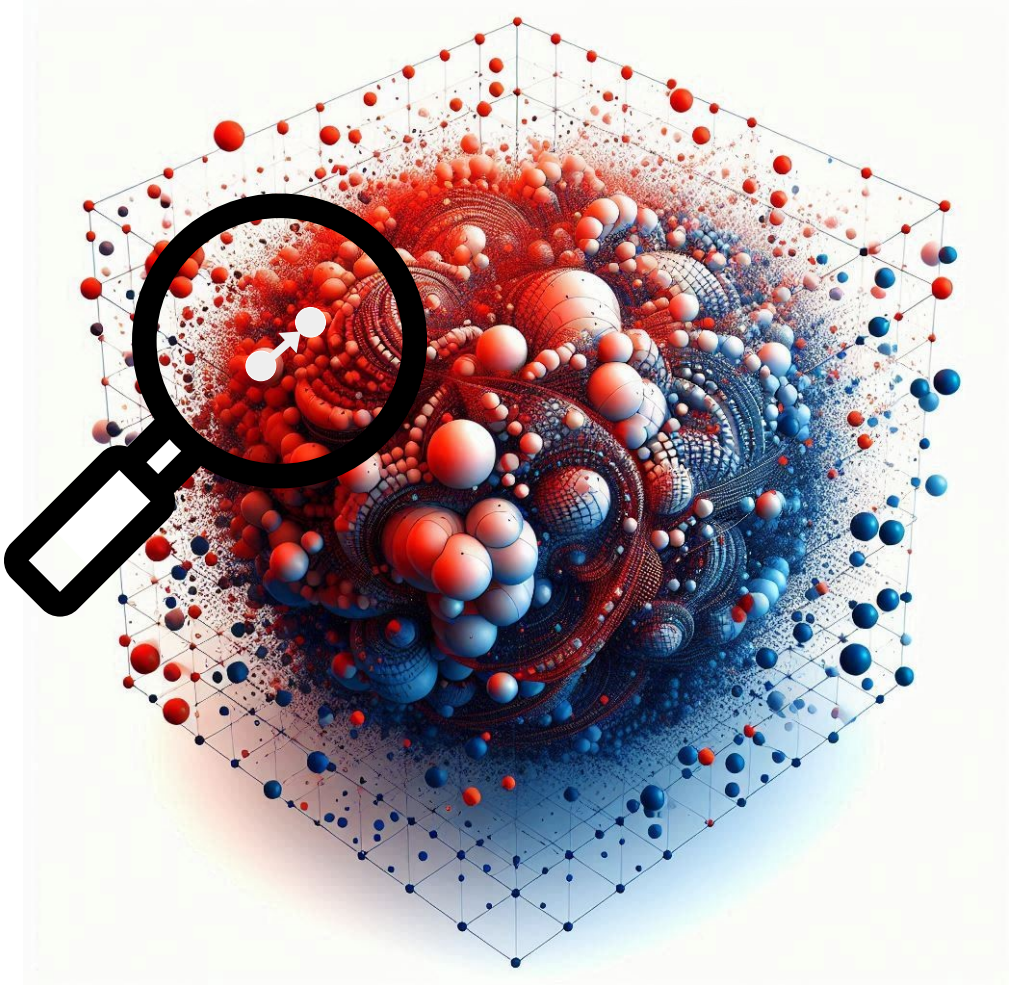
$$\mathcal{L}_\theta = \frac{1}{2}\left(r_t + \max_{a_{t+1}} Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)\right)^2$$

Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Target Network – Double Q-Learning

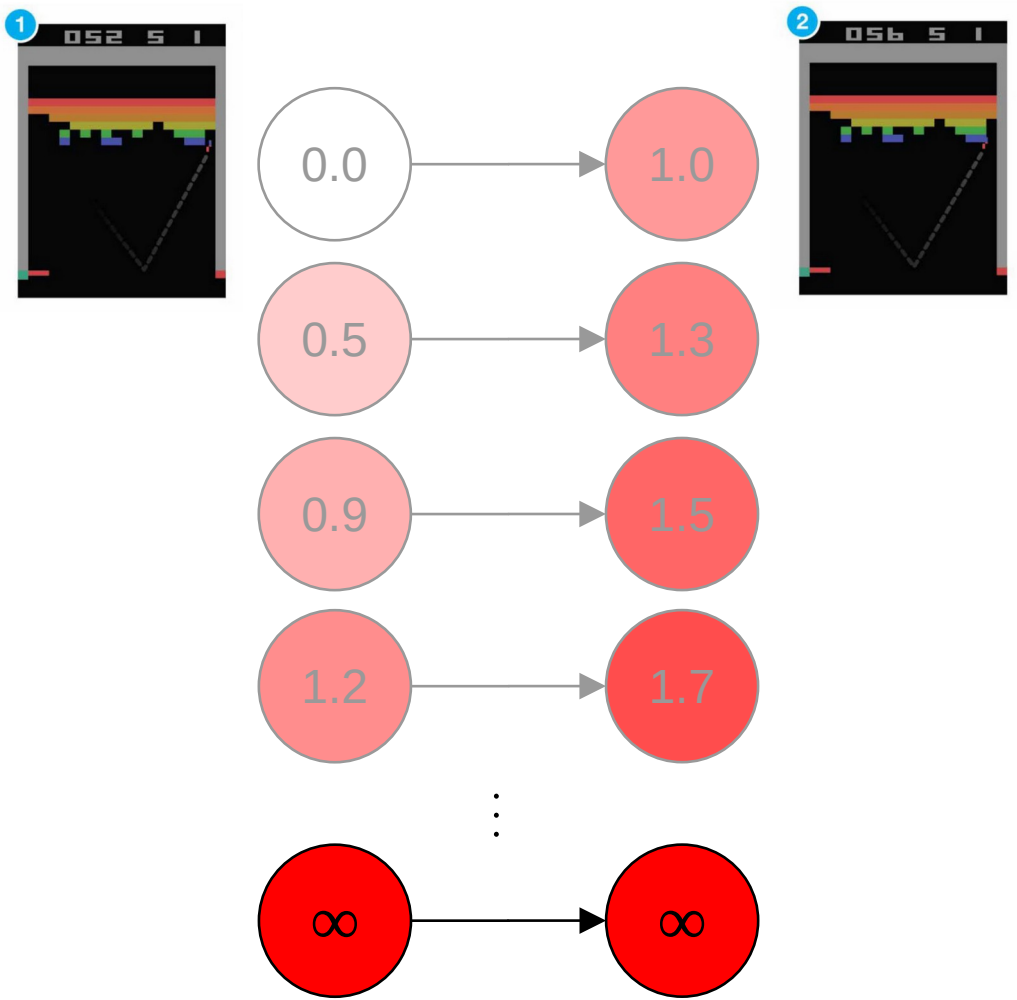


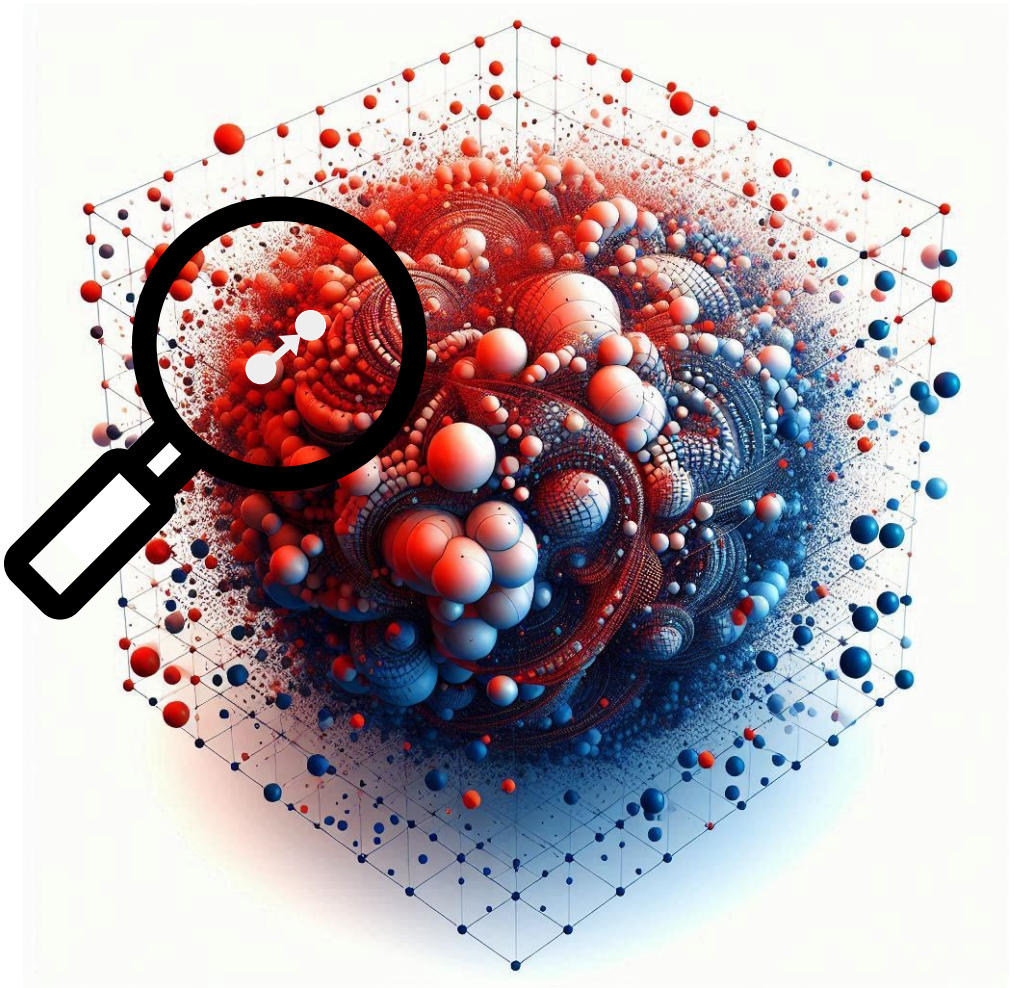
$$\mathcal{L}_\theta = \frac{1}{2}\left(r_t + \max_{a_{t+1}} Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)\right)^2$$



Mnih et al. (2015) “Human-level control through deep reinforcement learning”

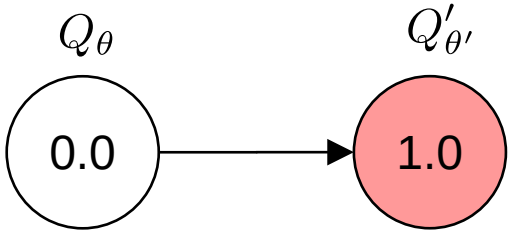
Target Network – Double Q-Learning





Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Target Network – Double Q-Learning



$$\mathcal{L}_\theta = \frac{1}{2}\left(\underbrace{r_t + \max_{a_{t+1}} Q'_{\theta'}(s_{t+1}, a_{t+1})}_{\text{target}} - \underbrace{Q_\theta(s_t, a_t)}_{\text{prediction}}\right)^2$$

Target Network
$Q'_{\theta'}(s_t, a_t)$

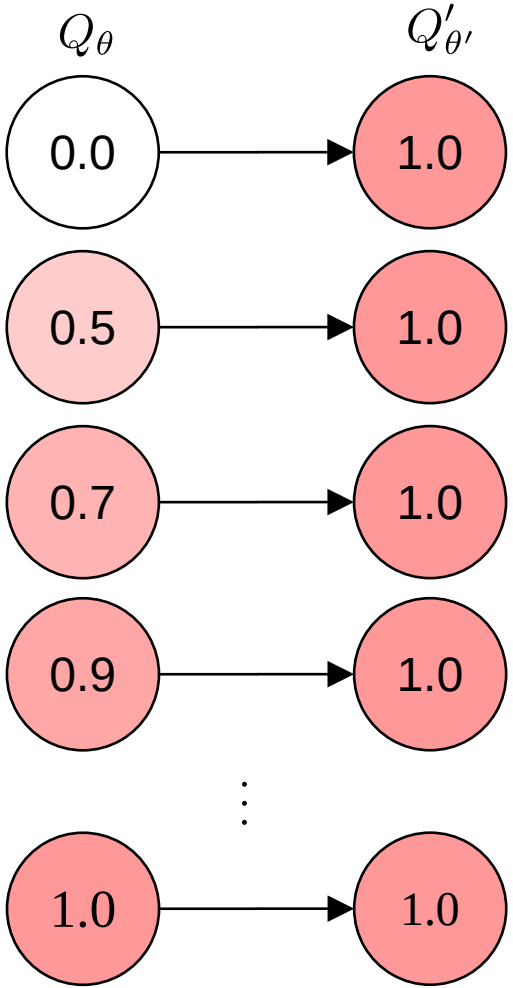
Prediction Network
$Q_\theta(s_t, a_t)$

Update parameters
every C iterations
$\theta' \leftarrow \theta$

Input
(s_t, a_t, r_t, s_{t+1})

Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Target Network – Double Q-Learning



$$\mathcal{L}_\theta = \frac{1}{2}\left(\underbrace{r_t + \max_{a_{t+1}} Q'_{\theta'}(s_{t+1}, a_{t+1})}_{\text{target}} - \underbrace{Q_\theta(s_t, a_t)}_{\text{prediction}}\right)^2$$

Target Network
$Q'_{\theta'}(s_t, a_t)$

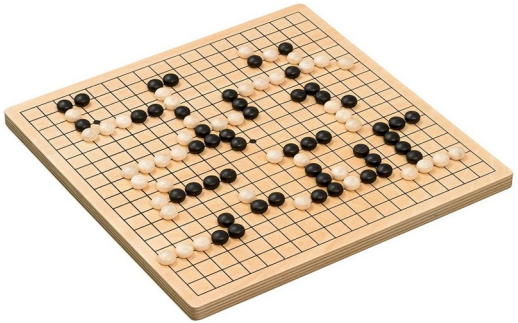
Prediction Network
$Q_\theta(s_t, a_t)$

Update parameters every C iterations
$\theta' \leftarrow \theta$

Input
(s_t, a_t, r_t, s_{t+1})

Mnih et al. (2015) “Human-level control through deep reinforcement learning”

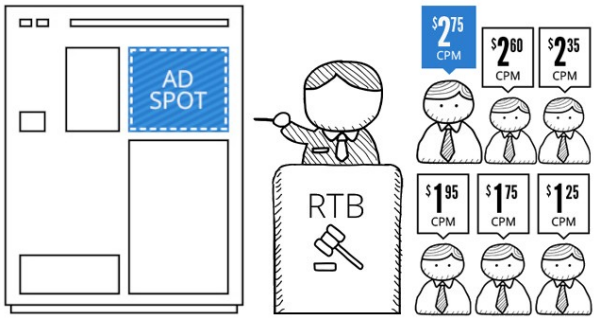
(Some) Applications



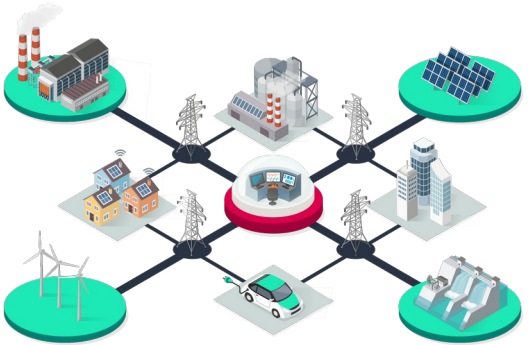
From Games to Programming

Robot Control

Generative AI: ChatGPT

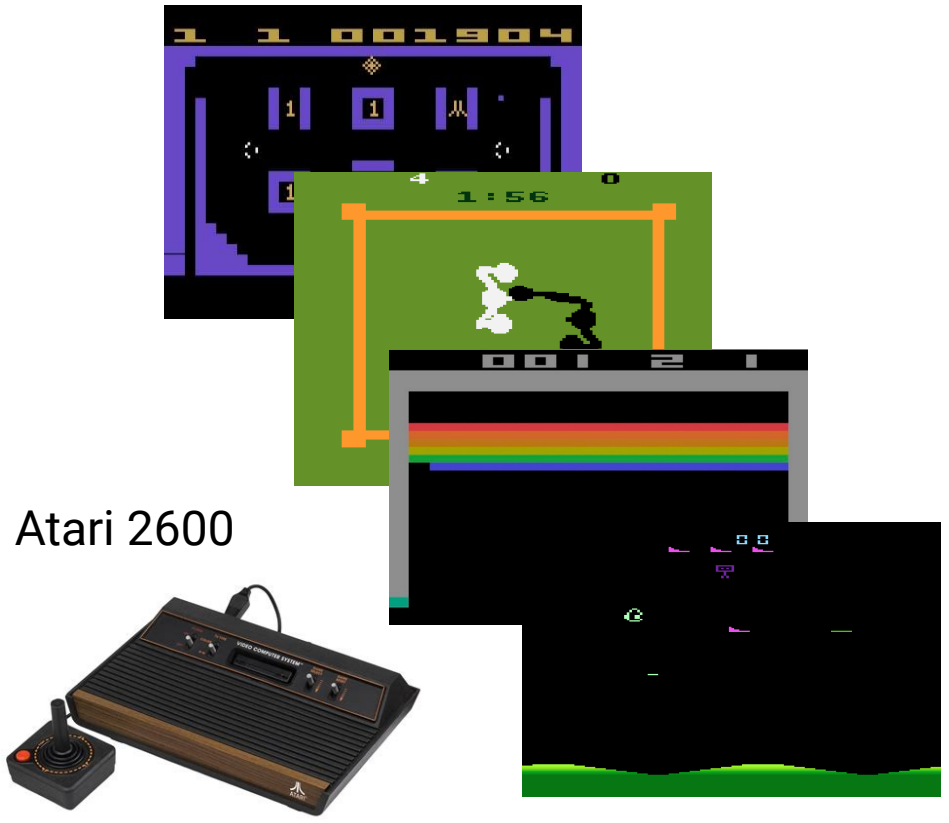


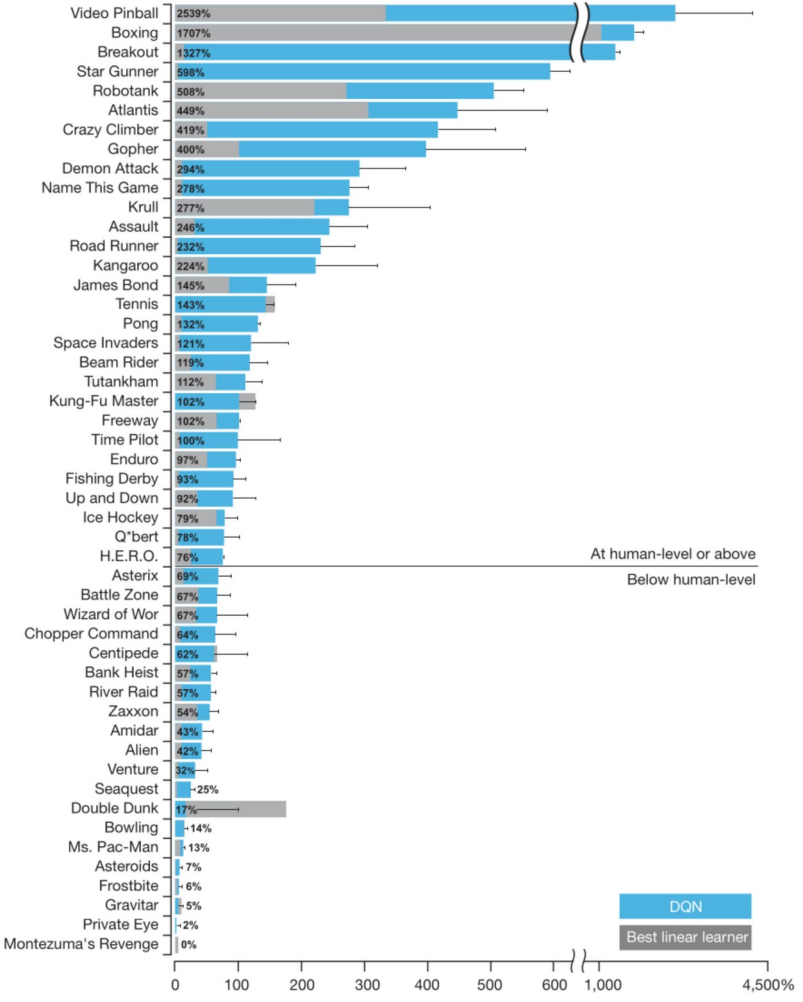
Marketing: Real Time Bidding



Industry: Smart Grids

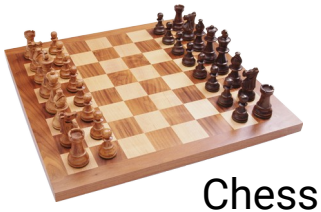
Games: DQN



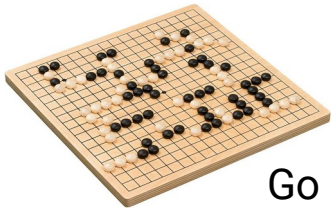


Mnih et al. (2015) “Human-level control through deep reinforcement learning”

Games: AlphaZero



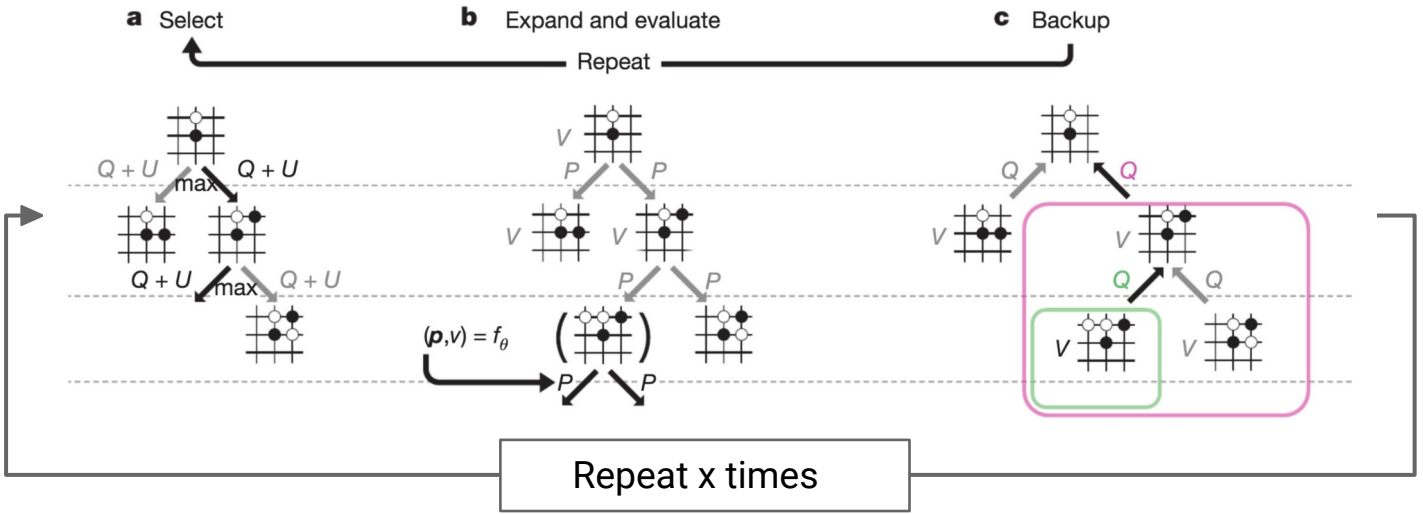
Chess



Go

Shōgi

Monte Carlo Tree Search (MCTS):



Silver et al. (2018) "A general reinforcement learning algorithm that masters chess, shogi and Go through self-play"

Google DeepMind

Games to Programming: AlphaDev

AssemblyGame

Sort operation of numbers: `variable_sort_2([5, 8])`

Assembler code

```
variable_sort_2(int, int*):
        ; Determine the number of elements
        cmp    $2, %edi
        ; Exit if less than 2 elements
        jne     .Label
        ; Below routine sorts 2 elements
        mov    (%rsi), %eax
        mov    4(%rsi), %ecx
        cmp    %eax, %ecx
        mov    %eax, %edx
        cmovl   %ecx, %edx
        mov    %edx, (%rsi)
        cmovg %ecx, %eax
        mov    %eax, 4(%rsi)

.Label:
        retq
```

State: $s_t = \langle C_t, M_t \rangle$

current code

memory after executing code on input

→ AlphaZero with Transformer model

Mankowitz et al. (2023) “Faster sorting algorithms discovered using deep reinforcement learning”

Google DeepMind

Games to Programming: AlphaDev

Results

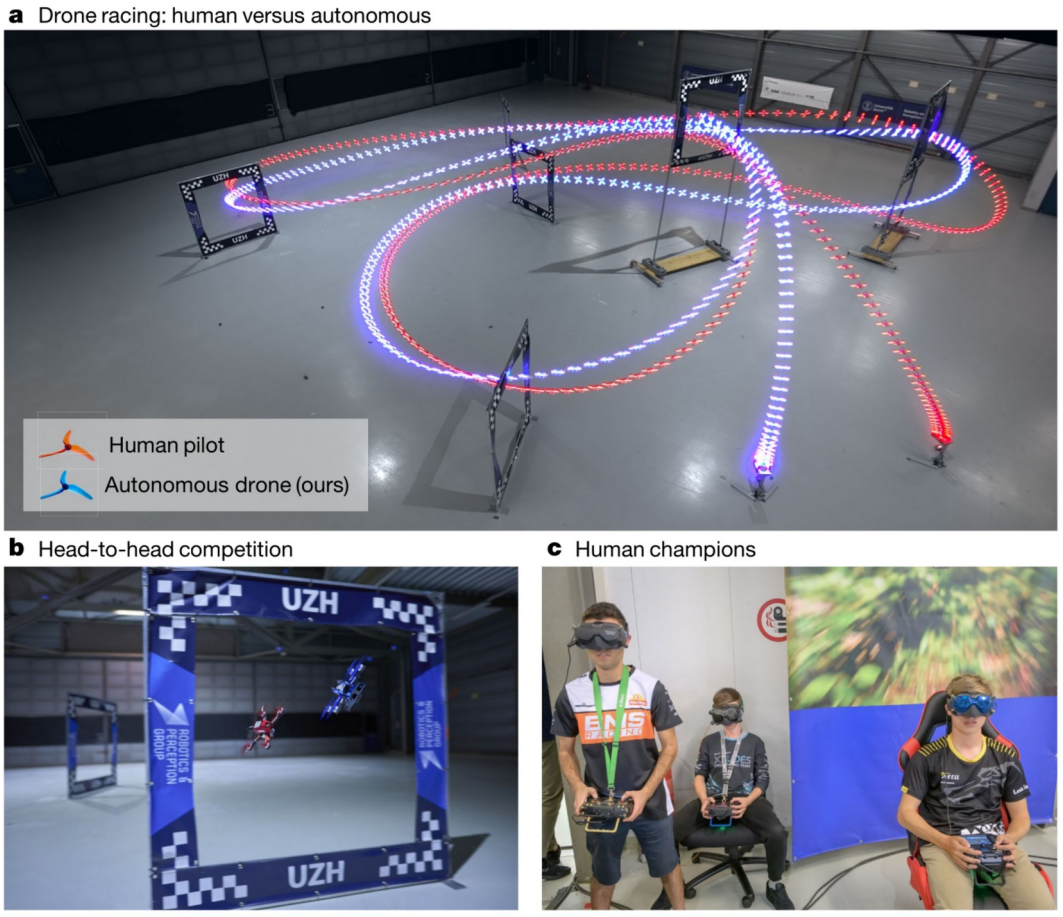
(a) Algorithm	AlphaDev	Human benchmarks
	Length	Length
Sort 3	17	18
Sort 4	28	28
Sort 5	42	46
VarSort3	21	33
VarSort4	37	66
VarSort5	63	115
VarInt	27	31

(b) Algorithm	AlphaDev	Human benchmarks
	Latency ± (lower, upper)	Latency ± (lower, upper)
VarSort3	236,498 ± (235,898, 236,887)	246,040 ± (245,331, 246,470)
VarSort4	279,339 ± (278,791, 279,851)	294,963 ± (294,514, 295,618)
VarSort5	312,079 ± (311,515, 312,787)	331,198 ± (330,717, 331,850)
VarInt	97,184 ± (96,885, 97,847)	295,358 ± (293,923, 296,297)
Competitive	75,973 ± (75,420, 76,638)	86,056 ± (85,630, 86,913)

Mankowitz et al. (2023) “Faster sorting algorithms discovered using deep reinforcement learning”

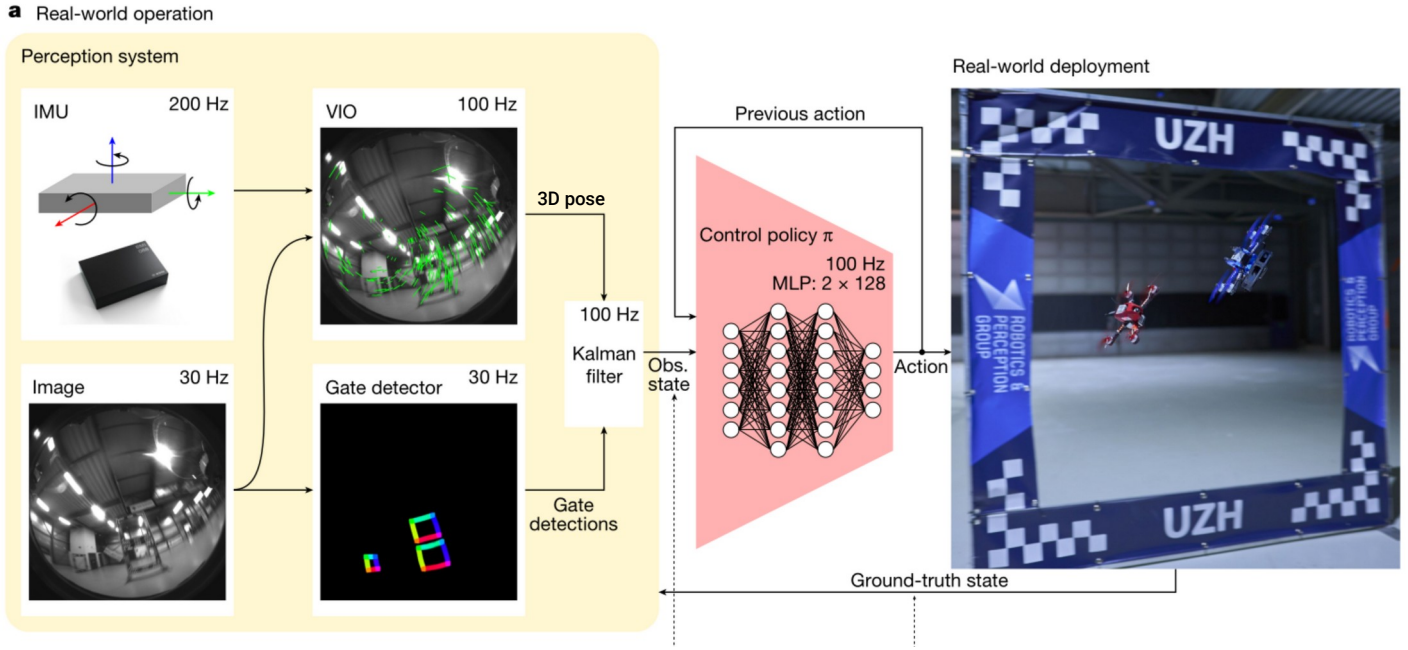
Google DeepMind

Robotics: Drones



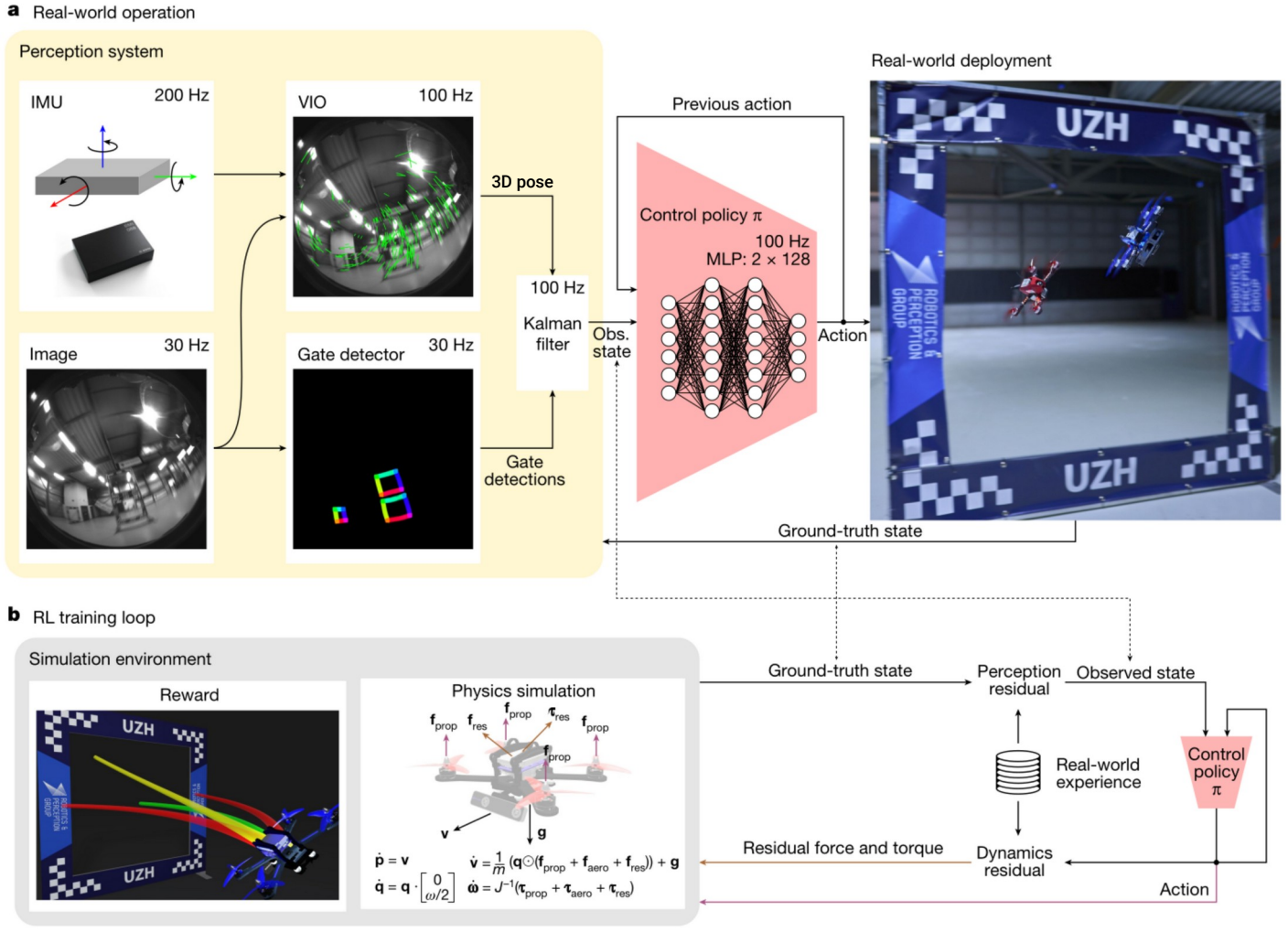
Kaufmann et al. (2023) “Champion-level drone racing using deep reinforcement learning”

Robotics: Drones



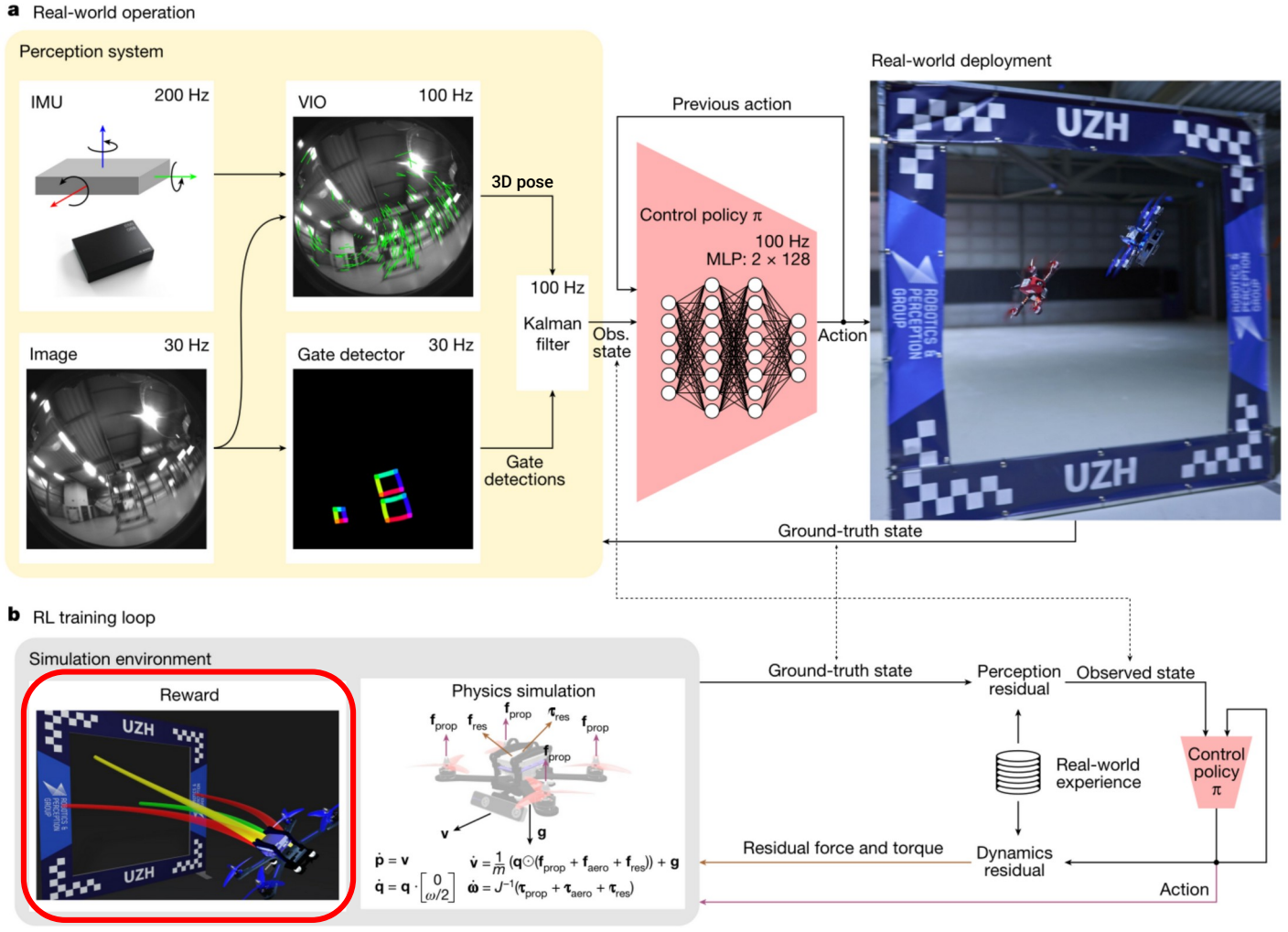
Kaufmann et al. (2023) “Champion-level drone racing using deep reinforcement learning”

Robotics: Drones



Kaufmann et al. (2023) “Champion-level drone racing using deep reinforcement learning”

Robotics: Drones

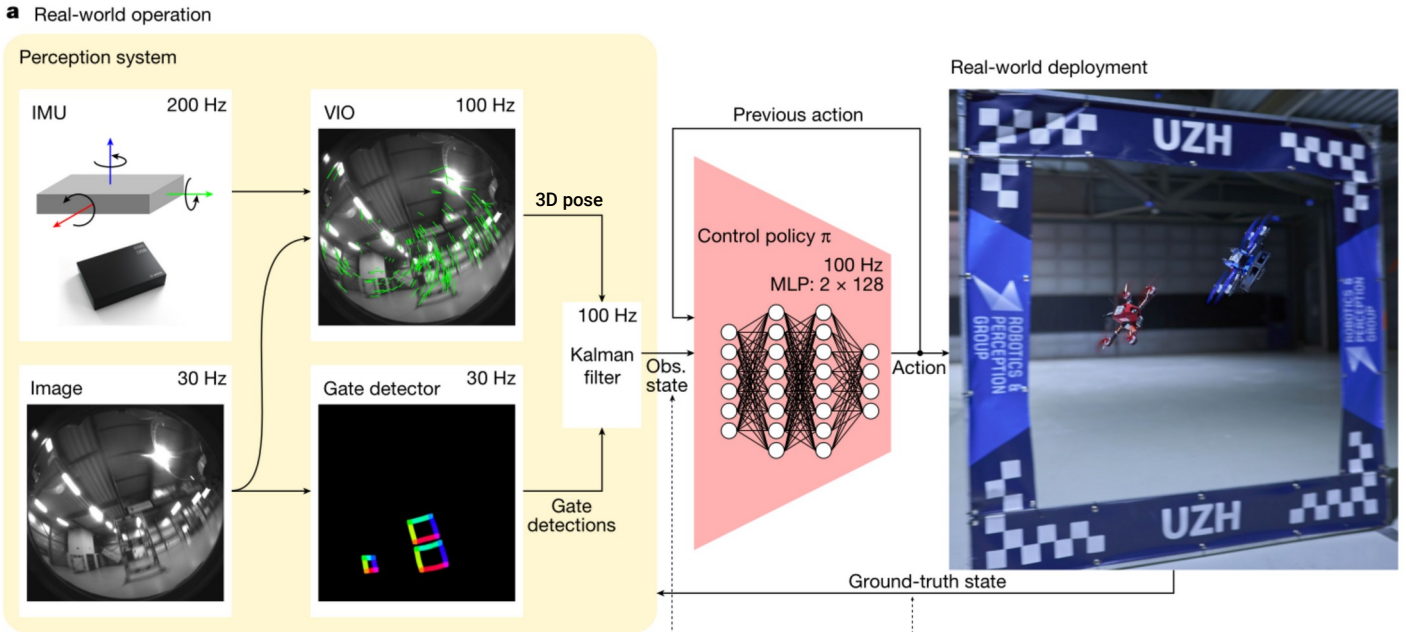


Reward:

progress to center of the next gate
+
keeping the gate in the camera view

Kaufmann et al. (2023) “Champion-level drone racing using deep reinforcement learning”

Robotics: Drones



Sim2Real Problem:

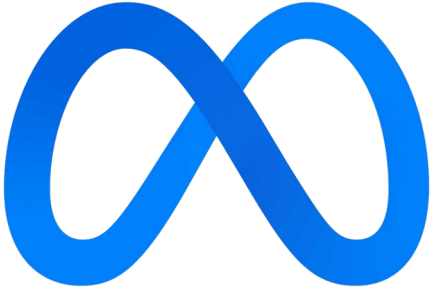
- Difference between simulated and real dynamics
- Noisy state estimation from real sensory data

→ Use real data to train GP and k-nearest neighbor regression models

Kaufmann et al. (2023) “Champion-level drone racing using deep reinforcement learning”

Large Language Models (LLMs)

Google ChatGPT

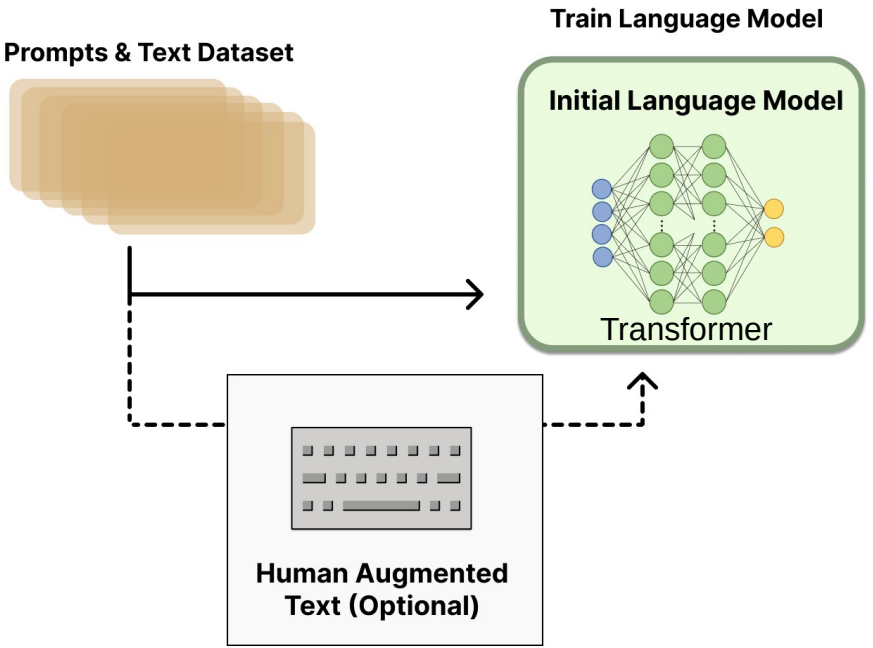


Meta AI Llama

Microsoft Copilot

Reinforcement Learning from Human Feedback (RLHF)

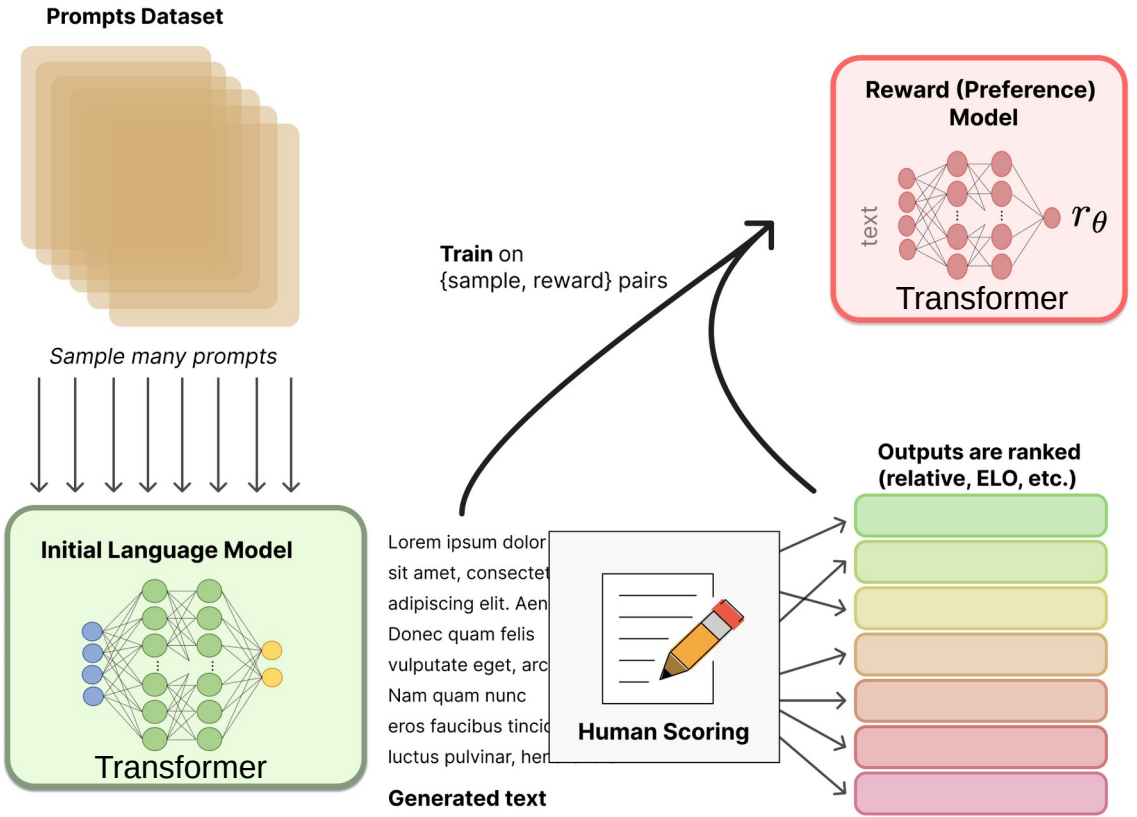
1) Train LLM



Lambert et al. (2022) "Illustrating Reinforcement Learning from Human Feedback (RLHF)"

Reinforcement Learning from Human Feedback (RLHF)

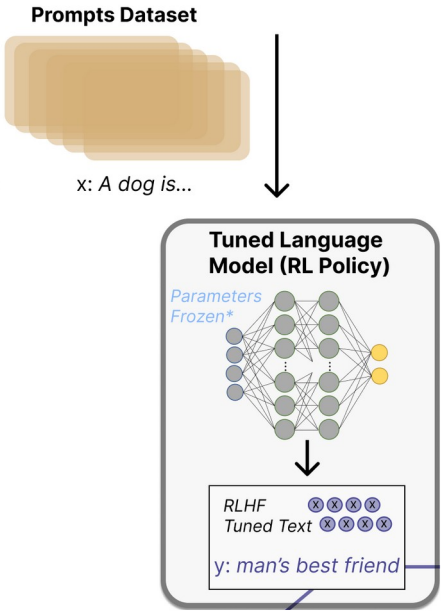
2) Train reward model



Lambert et al. (2022) "Illustrating Reinforcement Learning from Human Feedback (RLHF)"

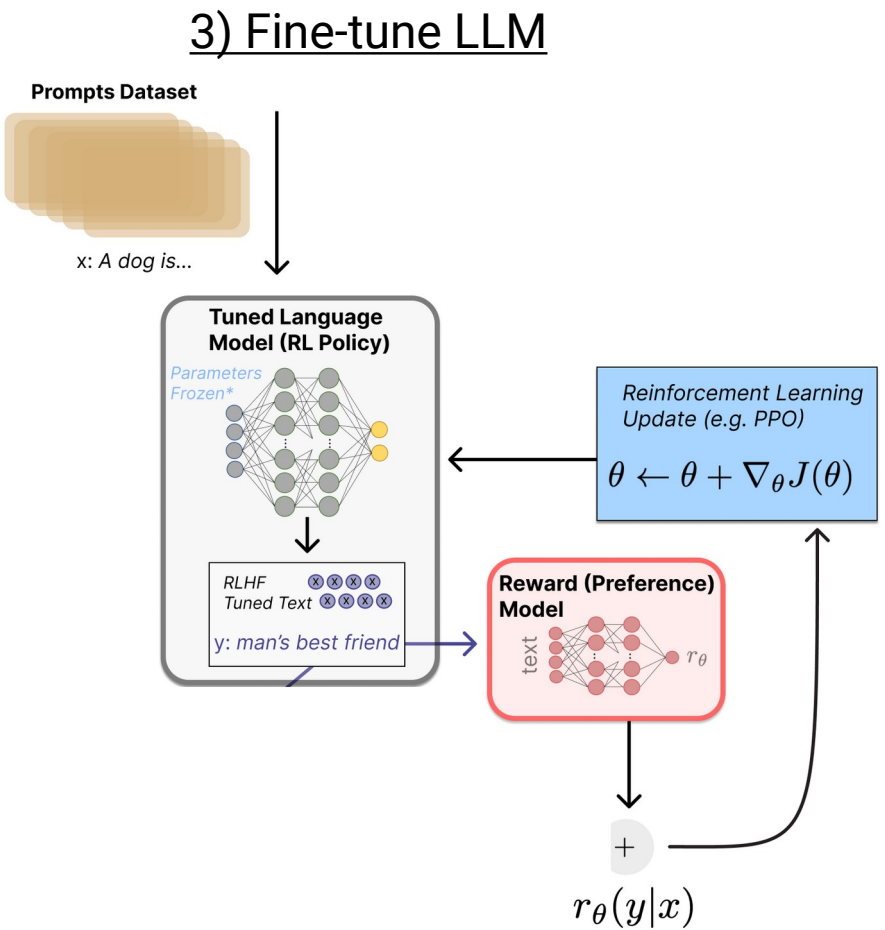
Reinforcement Learning from Human Feedback (RLHF)

3) Fine-tune LLM



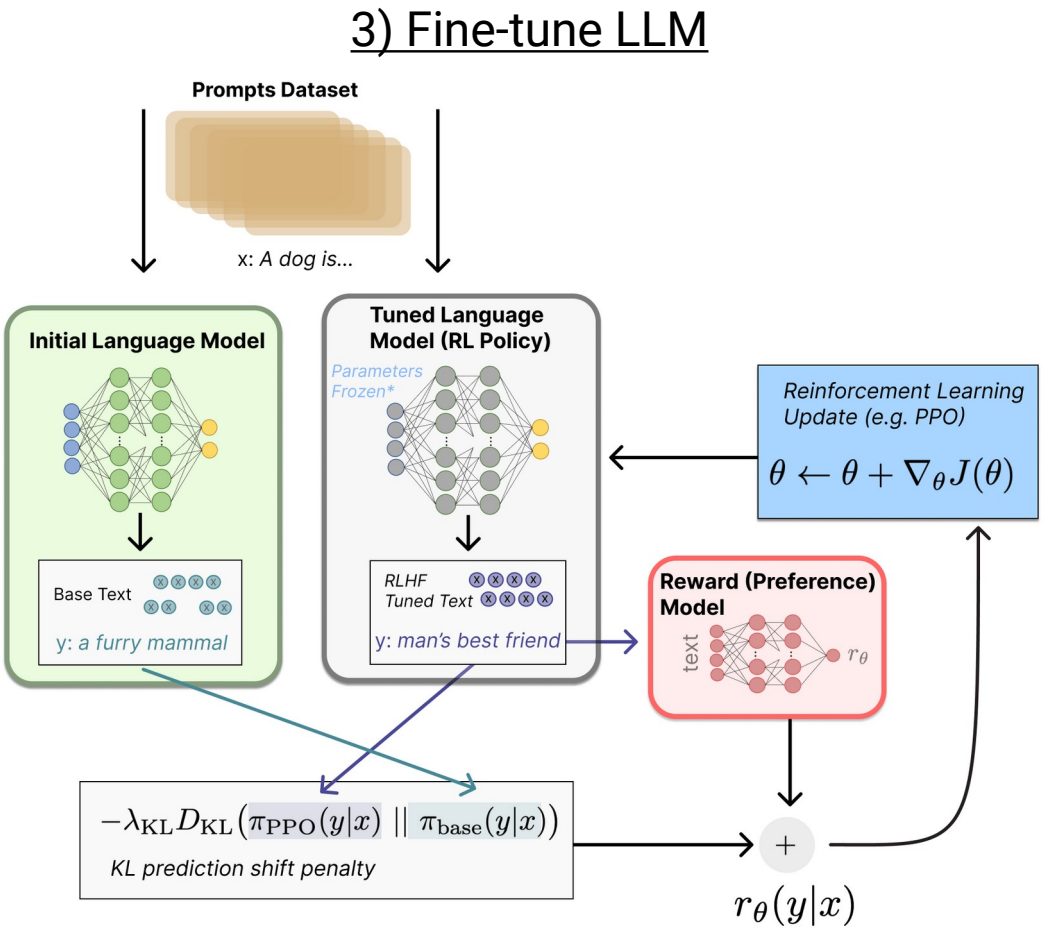
Lambert et al. (2022) "Illustrating Reinforcement Learning from Human Feedback (RLHF)"

Reinforcement Learning from Human Feedback (RLHF)



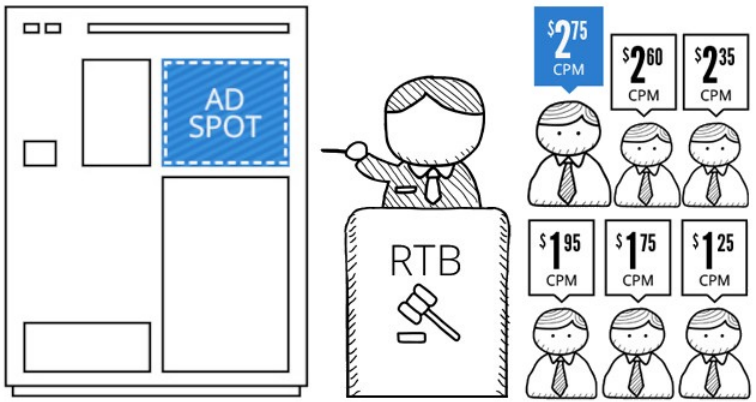
Lambert et al. (2022) "Illustrating Reinforcement Learning from Human Feedback (RLHF)"

Reinforcement Learning from Human Feedback (RLHF)

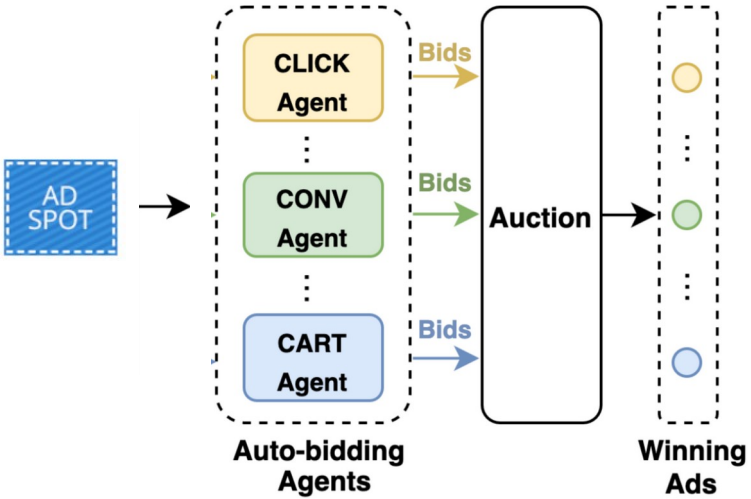


Lambert et al. (2022) "Illustrating Reinforcement Learning from Human Feedback (RLHF)"

Marketing: Real Time Bidding (RTB)



Provider: Alibaba

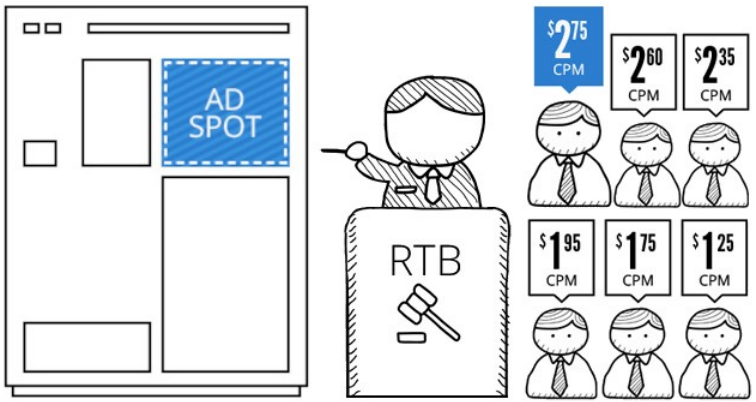


Muti-agent RL formulation:

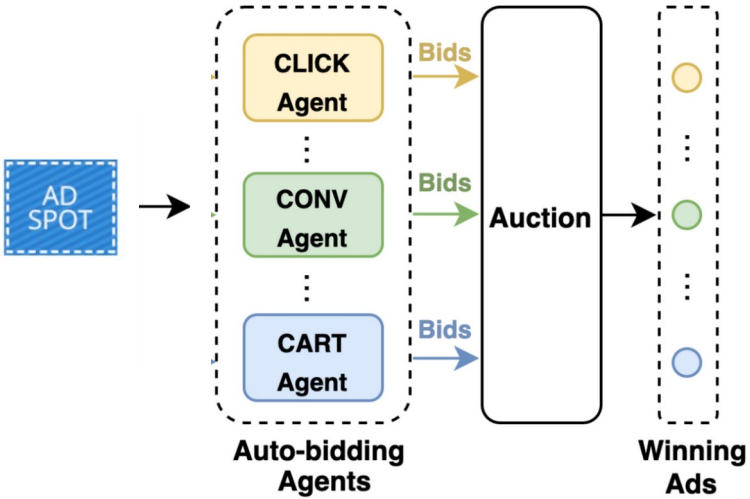
- State: $s_i = (B_i, v_i, T_i)$ (Budget, Impression Value, Timesteps left)
- Action: $b_i = \pi_i(s_i)$ (Bid)
- Reward: Competitive - $r_i = v_i$ Cooperative - $r_i = v_{\text{winner}}$

Wen et al. (2023) " Cooperative-Competitive Multi-Agent Framework for Auto-bidding in Online Advertising"

Marketing: Real Time Bidding (RTB)



Provider: Alibaba



Problems with training of agents:

- If competitive ($r_i = v_i$): Certain agents can dominate and hurt social welfare
- If cooperative ($r_i = v_{\text{winner}}$): Agents might cooperate too strongly and hurt revenue

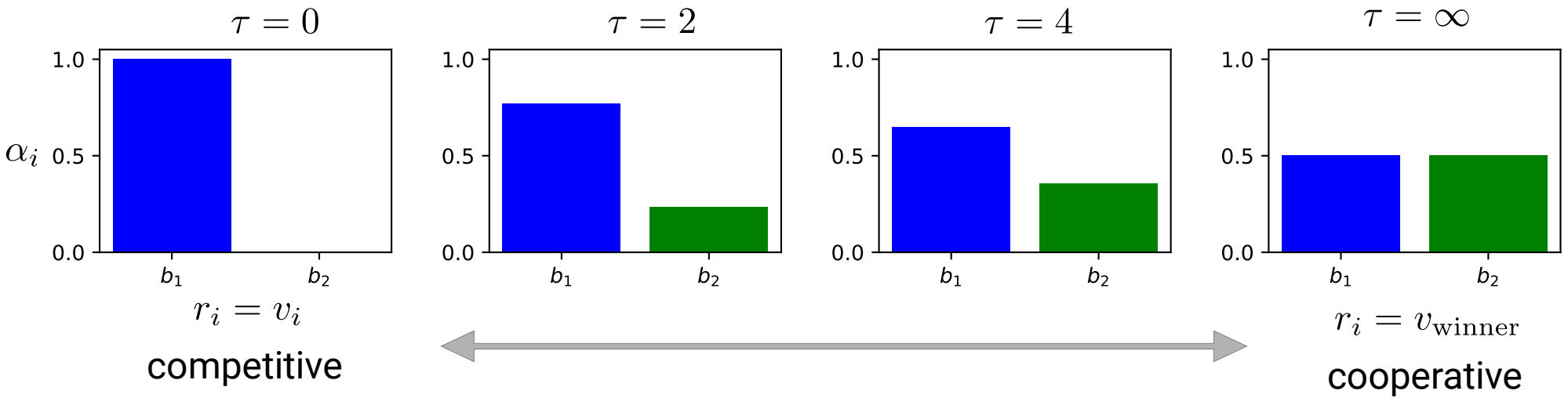
Wen et al. (2023) " Cooperative-Competitive Multi-Agent Framework for Auto-bidding in Online Advertising"

Marketing: Real Time Bidding (RTB)

Mixed cooperative-competitive reward:

$$r_i = \alpha_i \cdot v_{\text{winner}} \qquad \alpha_i = \frac{\exp(b_i \cdot \tau^{-1})}{\sum_j \exp(b_j \cdot \tau^{-1})}$$

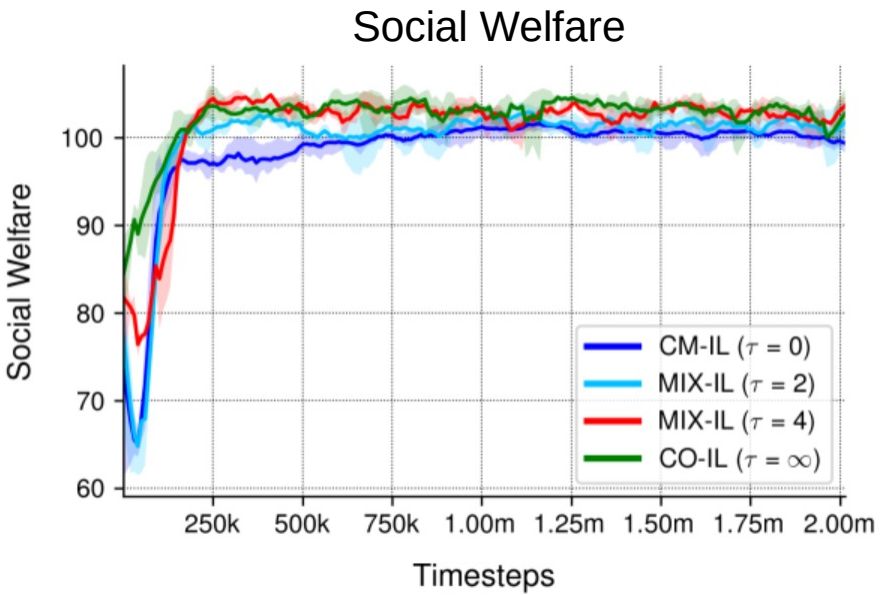
Two agents with different bids: $b_1 = 1.3 \mid b_2 = 1.0$

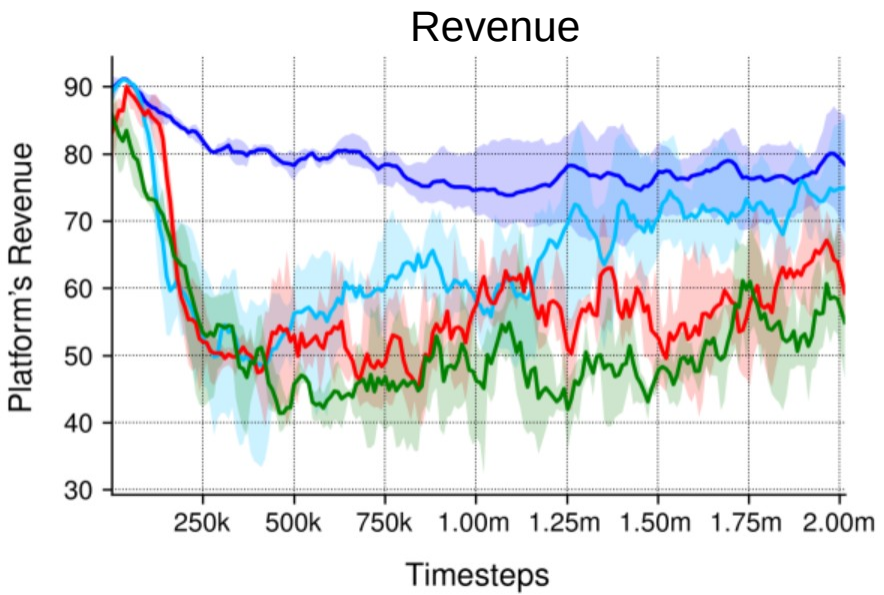


Wen et al. (2023) " Cooperative-Competitive Multi-Agent Framework for Auto-bidding in Online Advertising"

Marketing: Real Time Bidding (RTB)

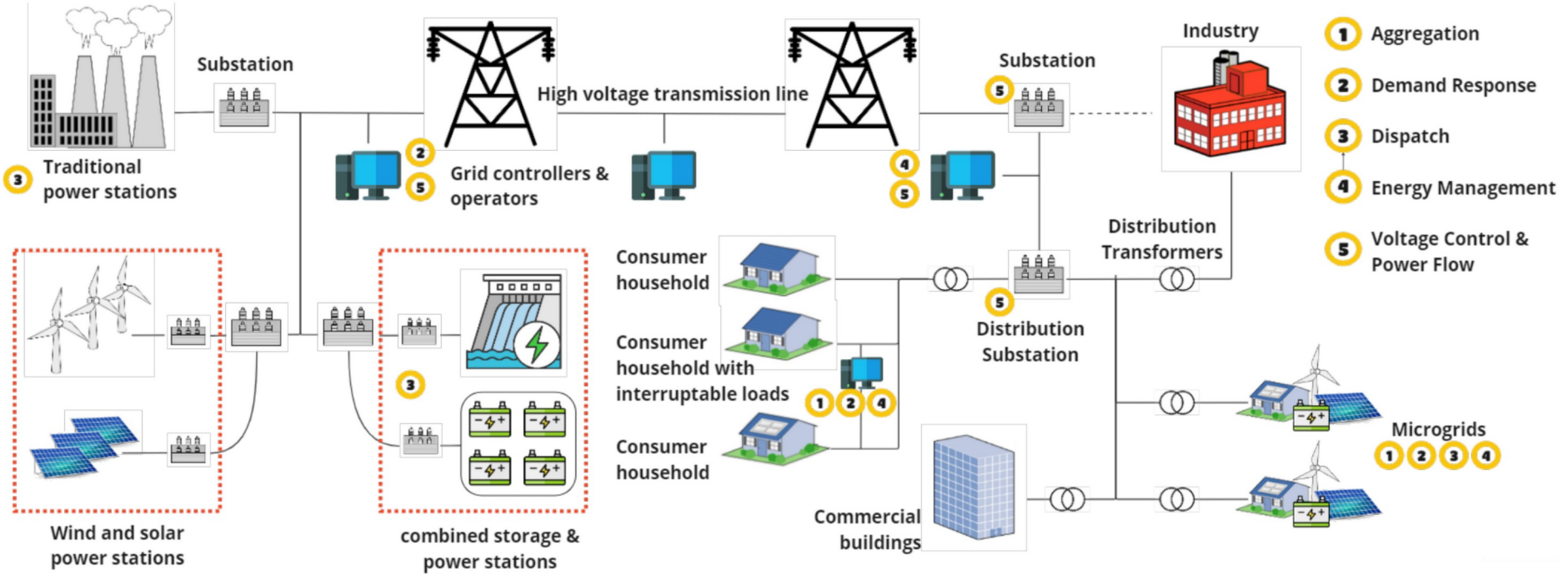
Results





Wen et al. (2023) " Cooperative-Competitive Multi-Agent Framework for Auto-bidding in Online Advertising"

Industry: Smart Grids

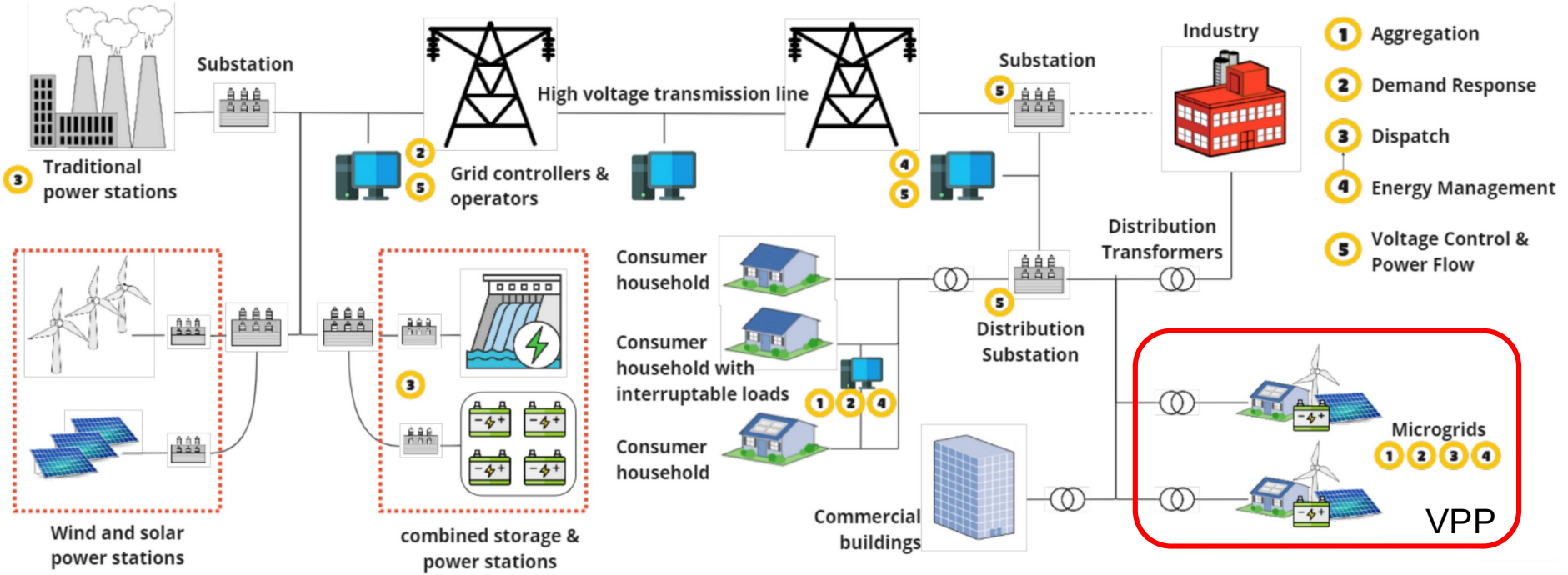


Keywords

- Multi-Agent RL
- Offline RL
- Safe RL

Ponse et al. (2024) "Reinforcement Learning for Sustainable Energy: A Survey"

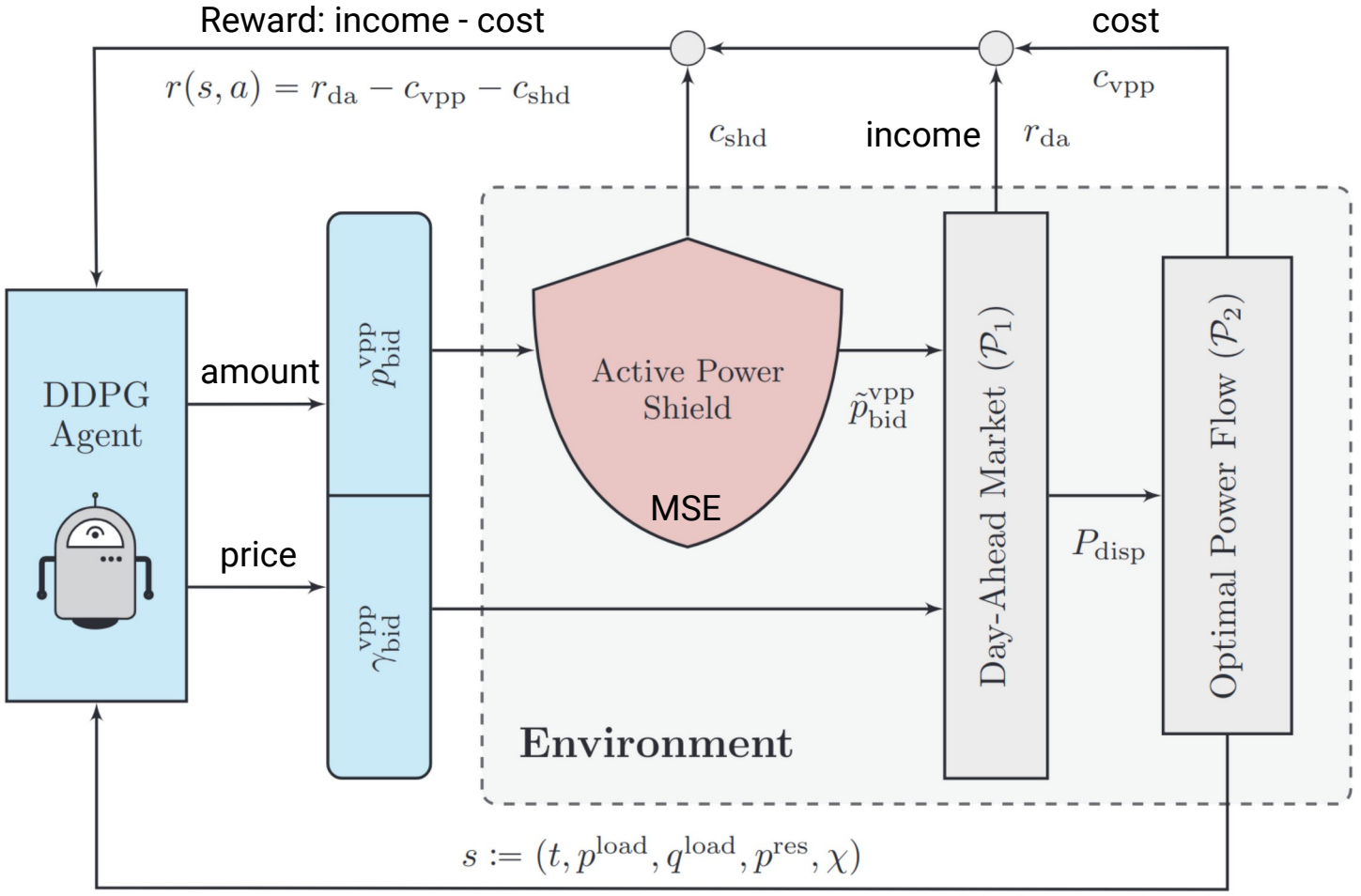
Industry: Safe Bidding of Virtual Power Plants (VPPs)



Safe Bidding: Sell power to network (price, amount), but stay in safe generation / load boundaries

Stanojev et al. (2023) "Safe Reinforcement Learning for Strategic Bidding of Virtual Power Plants in Day-Ahead Markets"

Industry: Safe Bidding of Virtual Power Plants (VPPs)



State: (timepoint in h of day, load, generation, storage capacity)

Industry: Strategic Bidding of Virtual Power Plants (VPPs)

Results

PERFORMANCE OF THE THREE DIFFERENT RL MODELS, WITH ALL VALUES GIVEN IN €/DAY.

Model	Day-Ahead	Balancing Cost	Net Market Profit	c_{vpp}	c_{sh}
uRL	162819.1	67163.8	95655.3	-	-
sRL	154594.4	0	154594.4	93052.12	10E-05

Stanojev et al. (2023) "Safe Reinforcement Learning for Strategic Bidding of Virtual Power Plants in Day-Ahead Markets"

Thank You!

Contact: ✉ chris.reinke@inria.fr

🌐 www.scirei.net