Introduction to RL

Oleksii Hrinchuk

Resources

- Reinforcement Learning: An Introduction (Richard Sutton and Andrew Barto)
  Holy Bible for reinforcement learning

- UCL RL course (David Silver)
  Great course which covers basics of RL, closely follows Sutton-Barto book

- UC Berkeley deep RL course (Sergey Levine)
  Awesome course on deep RL

Syllabus

- Week 1 (reinforcement learning basics)
  — Markov decision processes
  — Model-based prediction and control
  — Model-free prediction and control
  Practice: tabular RL methods

- Week 2 (deep reinforcement learning)
  — Value based methods
  — Policy based methods
  — Advances in deep RL
  Practice: off-policy deep RL methods

Course logistics

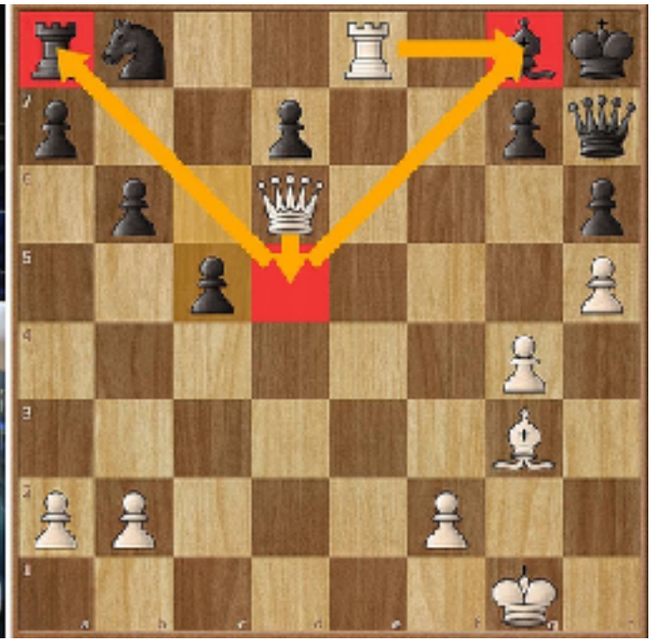
- Course materials: https://github.com/AlexGrinch/rl_course_acaml2018

- Slack channel for questions and discussions https://acaml2018.slack.com/messages/CCQGH9E3T

- Instructor: Oleksii Hrinchuk (Aleksey Grinchuk)
  - affiliation: MIPT, Skoltech
  - telegram: @rl_agent
  - email: oleksii.hrinchuk@skoltech.ru

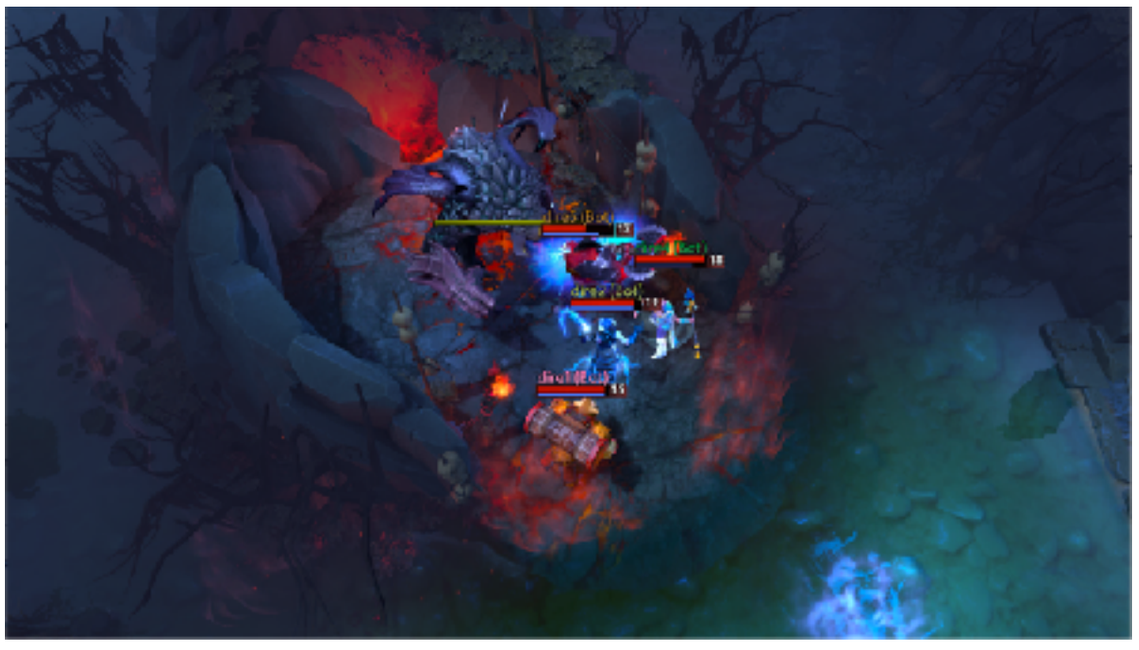
Do you like to play games?

Td-gammon (Tesauro, 1995)



Alpha Zero (DeepMind, 2017)

Alpha Go (DeepMind, 2016)

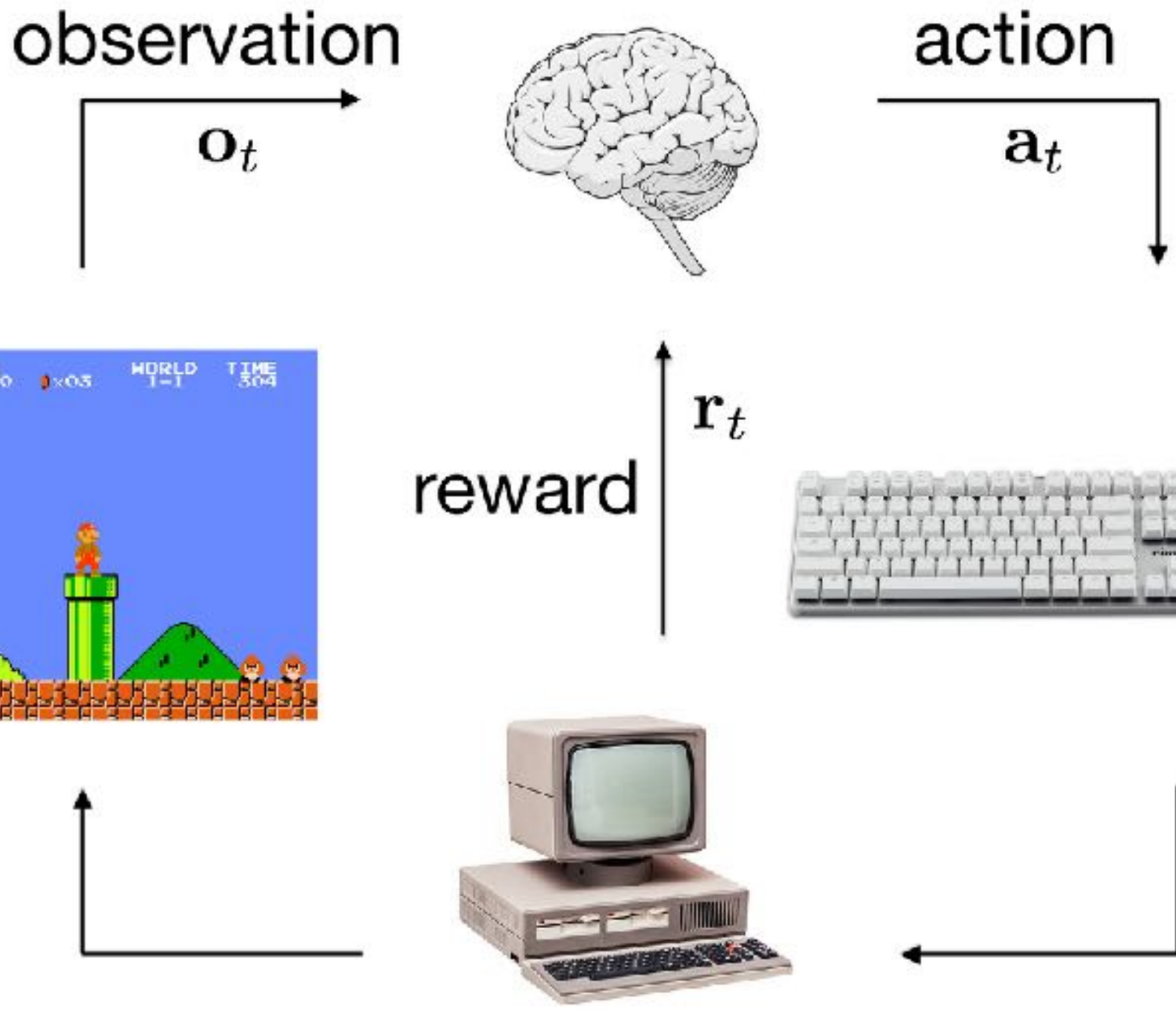


OpenAI Five (OpenAI, 2018)

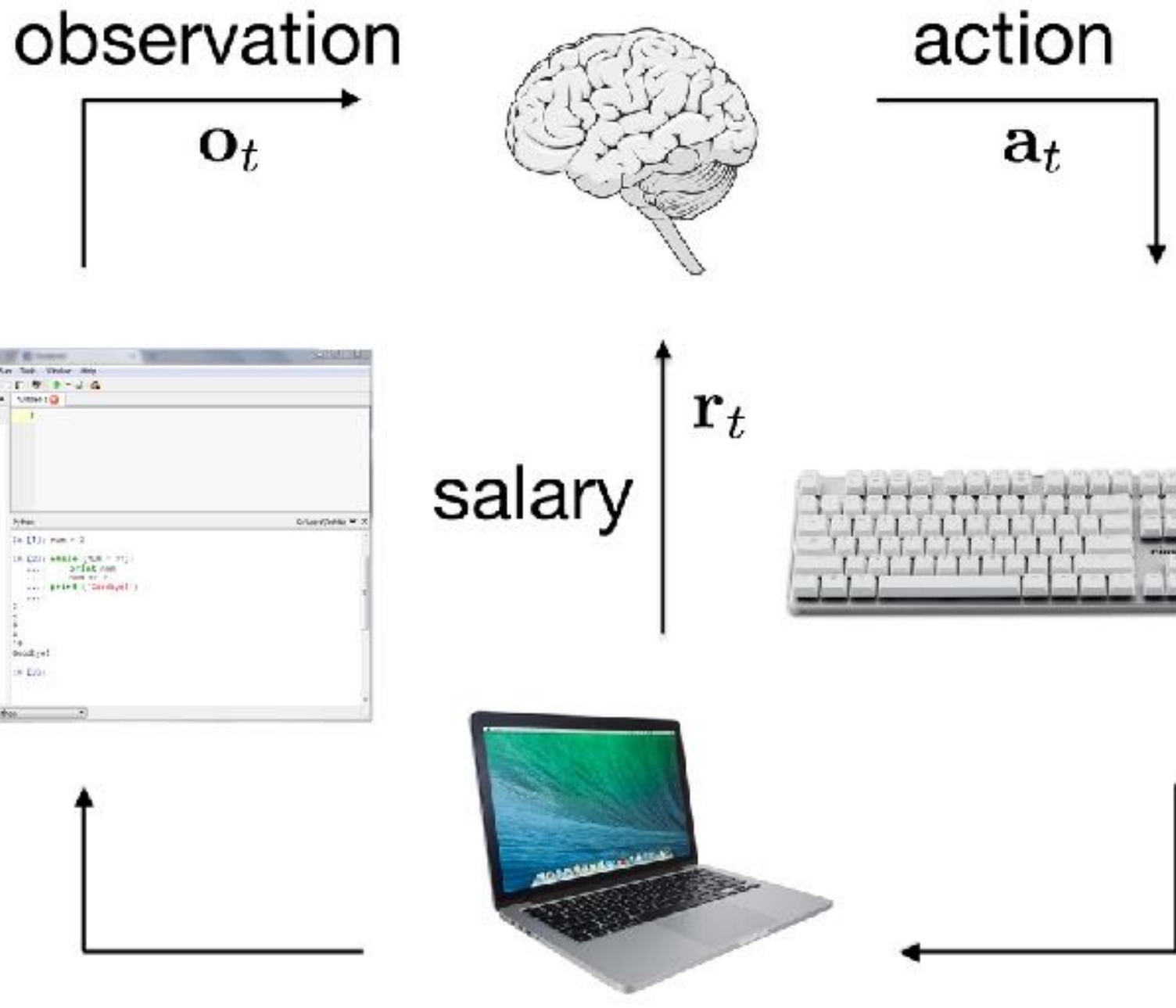
Stockfish vs. AlphaZero

Source: agadmator’s Chess Channel (https://youtu.be/7-MborNxYWE)

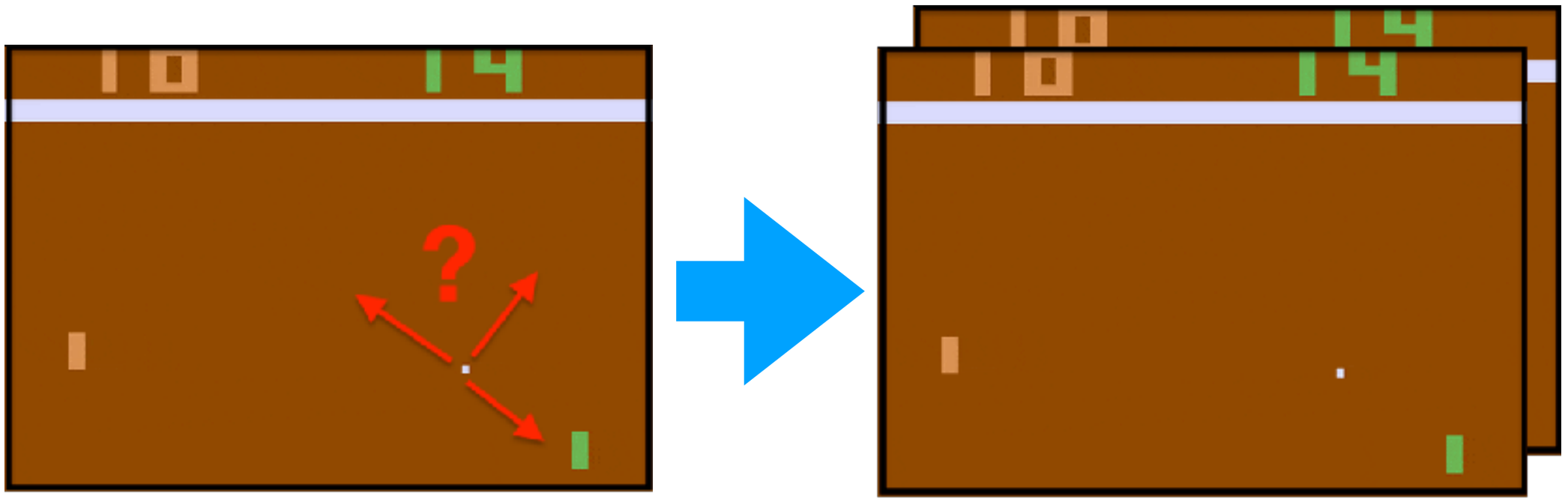
Markov Decision Process



Markov Decision Process



Observations and states



Observation $\mathbf{o}_t$

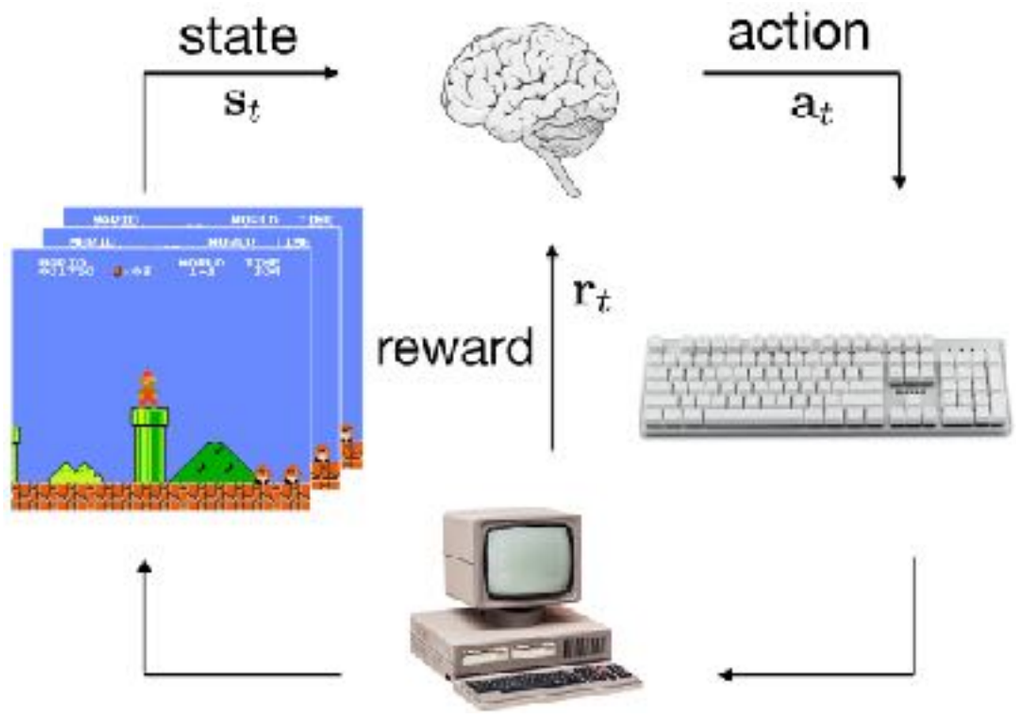
State $\mathbf{s}_t$

$$\mathbf{s}_t = [\mathbf{o}_{t-1}, \mathbf{o}_t]$$

Notation

- State $\mathbf{s}_t \in \mathcal{S}$
- Action $\mathbf{a}_t \in \mathcal{A}$
- Reward function

$$\mathbf{r}_t = r(\mathbf{s}_t, \mathbf{a}_t), \quad r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$



- Transition function
  - deterministic (forward dynamics)

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t), \quad f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$$

  - stochastic (transition probability)

$$\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t), \quad p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$$

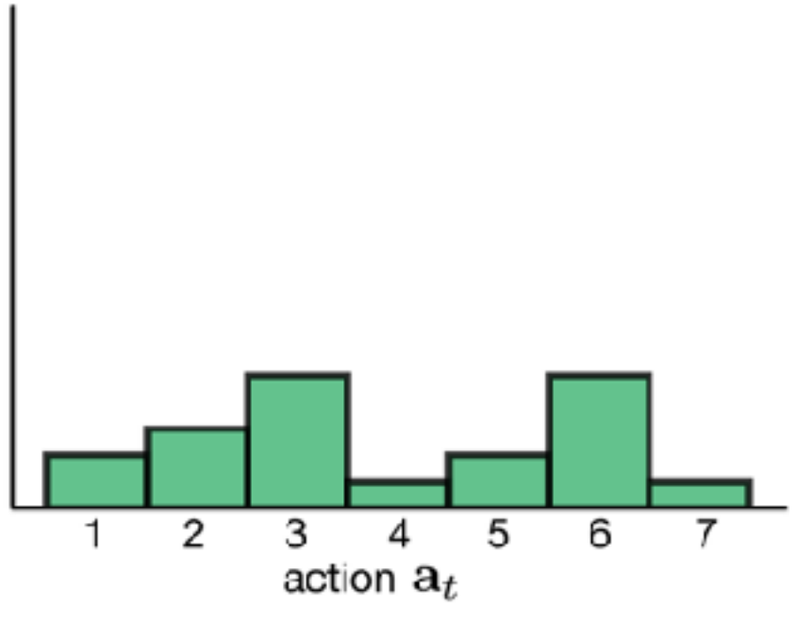
Policy

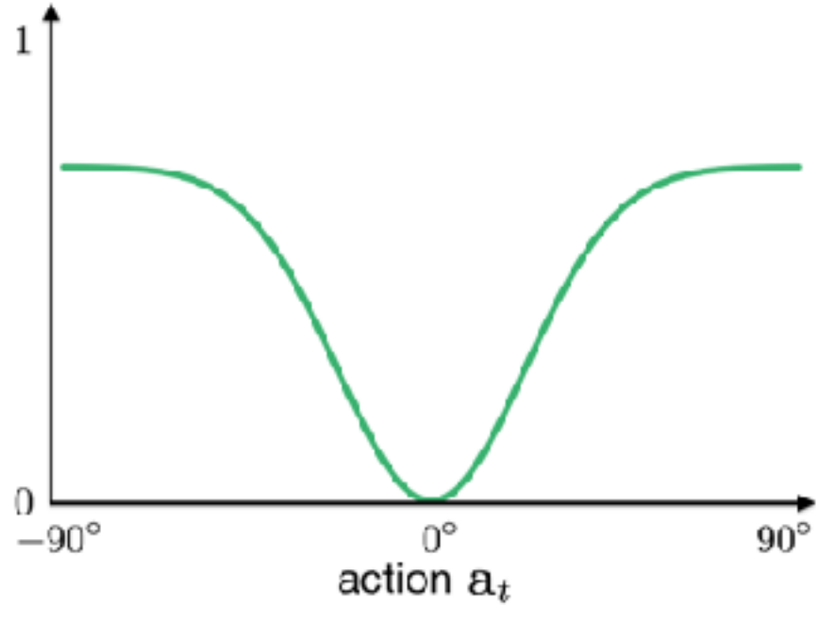
- Deterministic

$$\mathbf{a}_t = \mu(\mathbf{s}_t), \quad \mu : \mathcal{S} \rightarrow \mathcal{A}$$

- Stochastic

$$\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t), \quad \pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_{\geq}$$





Markov property

- Conditional probability distribution of future states of MDP depends only upon the present state, not on the sequence of events that preceded it.

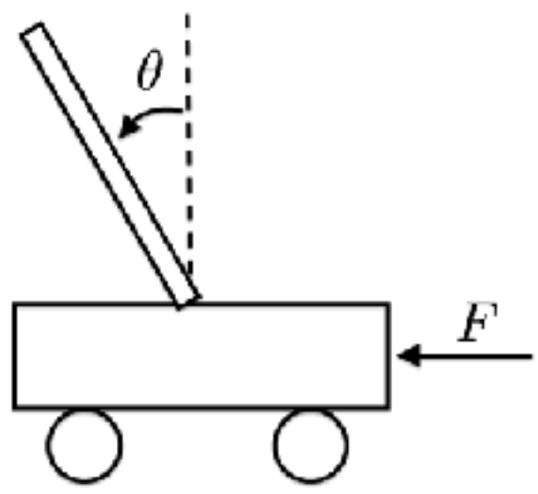
$$p(\mathbf{s}_{t+1}|\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_t, \mathbf{a}_t) = p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

- Which is not the case for observations

$$p(\mathbf{o}_{t+1}|\mathbf{o}_0, \mathbf{o}_1, \ldots, \mathbf{o}_t, \mathbf{a}_t) \neq p(\mathbf{o}_{t+1}|\mathbf{o}_t, \mathbf{a}_t)$$

$$\mathbf{s}_t = (\theta_t, \dot{\theta}_t), \quad \mathbf{o}_t = \theta_t$$

$$\ddot{\theta} = \frac{g \sin\theta - \alpha m l \dot{\theta}^2 \sin(2\theta)/2 - \alpha F \cos\theta}{4l/3 - \alpha m l \cos^2\theta}$$



Reinforcement learning

- Trajectory (episode)

$$\tau_\pi = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{a}_{T-1}, \mathbf{s}_T)$$

$$\mathbf{s}_0 \sim p(\mathbf{s}_0), \quad \mathbf{s}_t \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), \quad \mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$$

- Optimization problem

$$\mathbb{E}_{\tau_\pi}\left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right] \to \max_\pi$$

$$\mathbb{E}_{\tau_\pi}\left[\sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right] \to \max_\pi, \quad 0 \le \gamma \le 1$$

discount factor

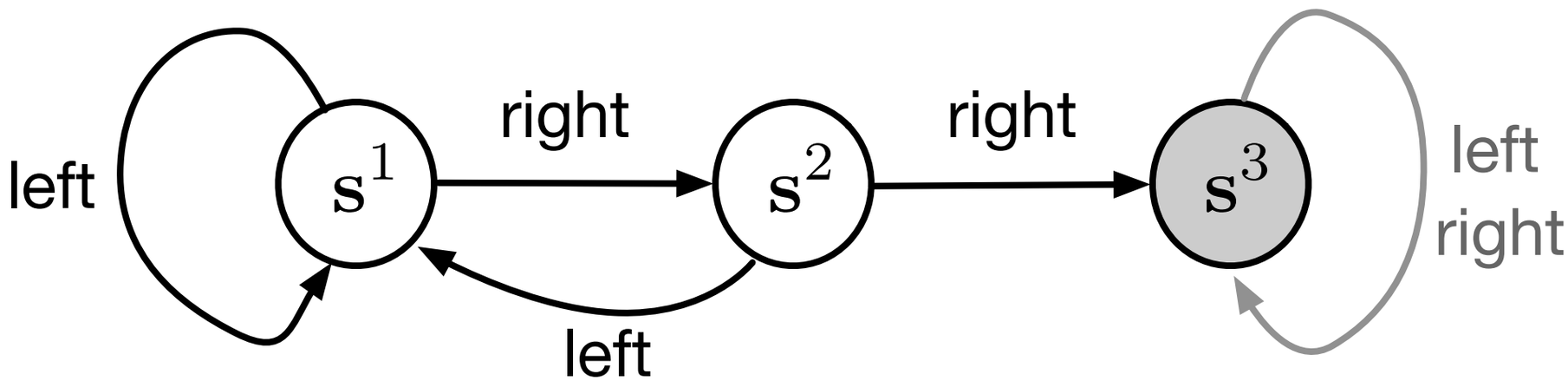
MDP as a graph

- Transition probability tensor $P_{\mathbf{s}^i \mathbf{s}^j \mathbf{a}^k} = p(\mathbf{s}^i | \mathbf{s}^j, \mathbf{a}^k)$

$$P[:,;,\mathbf{a}^l] = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad P[:,;,\mathbf{a}^r] = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

- Reward matrix $R_{\mathbf{s}^i \mathbf{a}^k} = r(\mathbf{s}^i, \mathbf{a}^k)$

$$R_{\mathbf{s}^i \mathbf{a}^k} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$



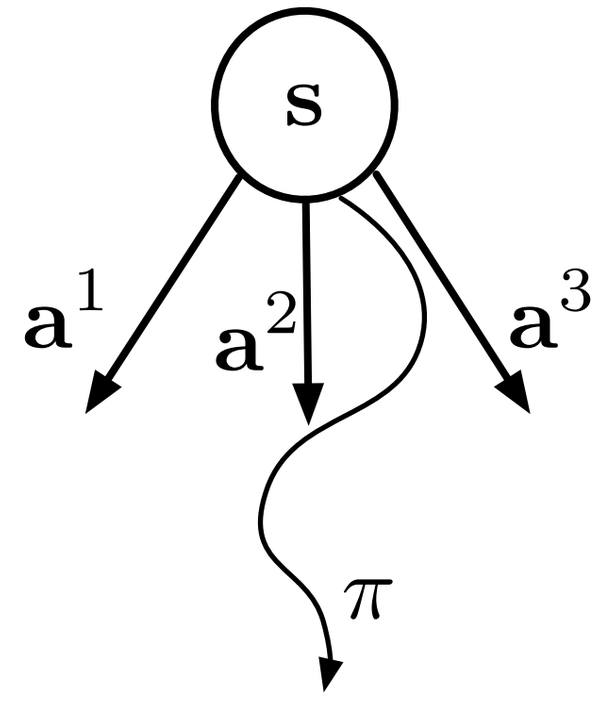
Value function

- Value function shows how good it is to act in accordance to policy π starting from some state $\mathbf{s}_t$

$$V^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \ldots\right]$$

- Optimal value function shows the maximum amount of reward we can get starting from some state $\mathbf{s}_t$

$$V^{*}(\mathbf{s}_t) = \max_{\pi} V^{\pi}(\mathbf{s}_t), \quad \forall \mathbf{s}_t \in \mathcal{S}$$



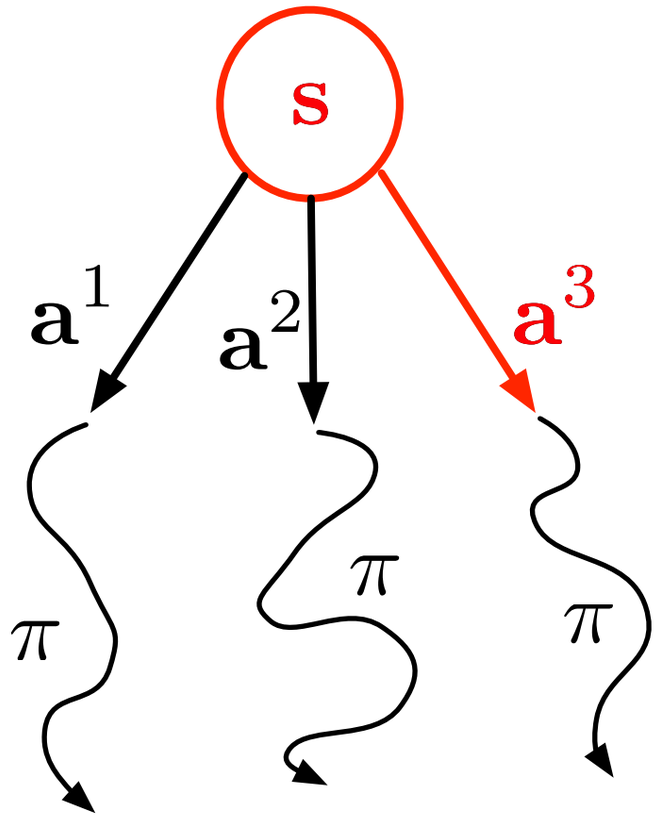
Q-function

- Q-function shows how good it is to act in accordance to policy π starting from some state $\mathbf{s}_t$ after taking action $\mathbf{a}_t$

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \ldots\right]$$

- Optimal Q-function shows the maximum amount of reward we can get starting from some state $\mathbf{s}_t$ after taking action $\mathbf{a}_t$

$$Q^{*}(\mathbf{s}_t, \mathbf{a}_t) = \max_{\pi} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t), \quad \forall(\mathbf{s}_t, \mathbf{a}_t)$$



Connection between V and Q

$$V^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \ldots \right]$$

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \ldots \right]$$

$$V^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$$

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V^{\pi}(\mathbf{s}_{t+1})$$

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \mathbf{a}_{t+1}} Q^{\pi}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$$

Bellman expectation equation

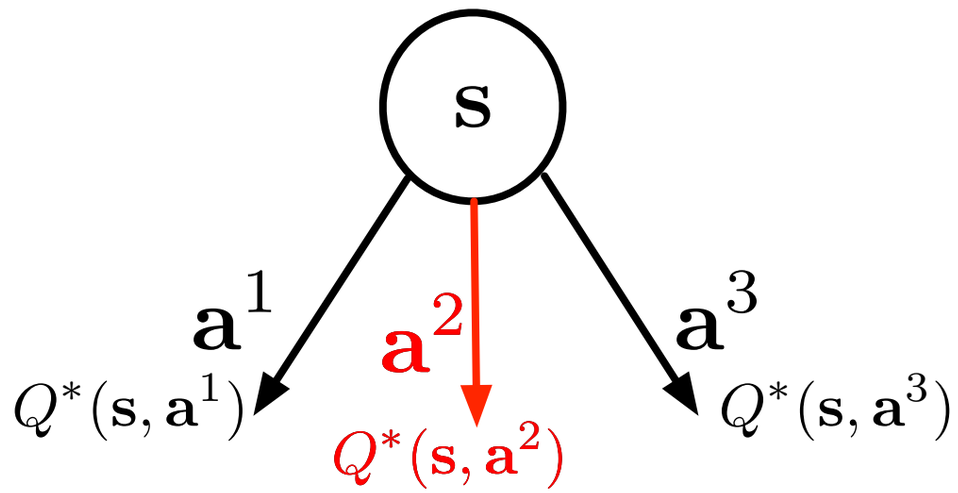
Optimal policy

- Suppose we know the optimal Q-function.

$$Q^*(\mathbf{s}_t, \mathbf{a}_t) = \max_{\pi} Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t), \quad \forall(\mathbf{s}_t, \mathbf{a}_t)$$

What will be the optimal policy?

$$\pi^*(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1, & \mathbf{a}_t = \arg\max_{\mathbf{a}_t \in \mathcal{A}} Q^*(\mathbf{s}_t, \mathbf{a}_t) \\ 0, & \text{otherwise} \end{cases}$$



Optimal policy

$$\pi^*(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1, \ \mathbf{a}_t = \arg\max\limits_{\mathbf{a}_t \in \mathcal{A}} Q^*(\mathbf{s}_t, \mathbf{a}_t) \\ 0, \ \text{otherwise} \end{cases}$$

$$\begin{aligned} \mathbb{E}_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t) &= \mathbb{E}_{\mathbf{a}_t \sim \pi^*} \left[Q^*(\mathbf{s}_t, \mathbf{a}_t) \right] \\ &= \sum_{\mathbf{a}_t \in \mathcal{A}} \pi^*(\mathbf{a}_t|\mathbf{s}_t) Q^*(\mathbf{s}_t, \mathbf{a}_t) \\ &= 1 \cdot Q^*(\mathbf{s}_t, \arg\max_{\mathbf{a}_t \in \mathcal{A}} Q^*(\mathbf{s}_t, \mathbf{a}_t)) \\ &= \max_{\mathbf{a}_t \in \mathcal{A}} Q^*(\mathbf{s}_t, \mathbf{a}_t) \end{aligned}$$

Optimal policy

$$V^*(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi^*}\left[Q^*(\mathbf{s}_t, \mathbf{a}_t)\right] = \max_{\mathbf{a}_t \in \mathcal{A}} Q^*(\mathbf{s}_t, \mathbf{a}_t)$$

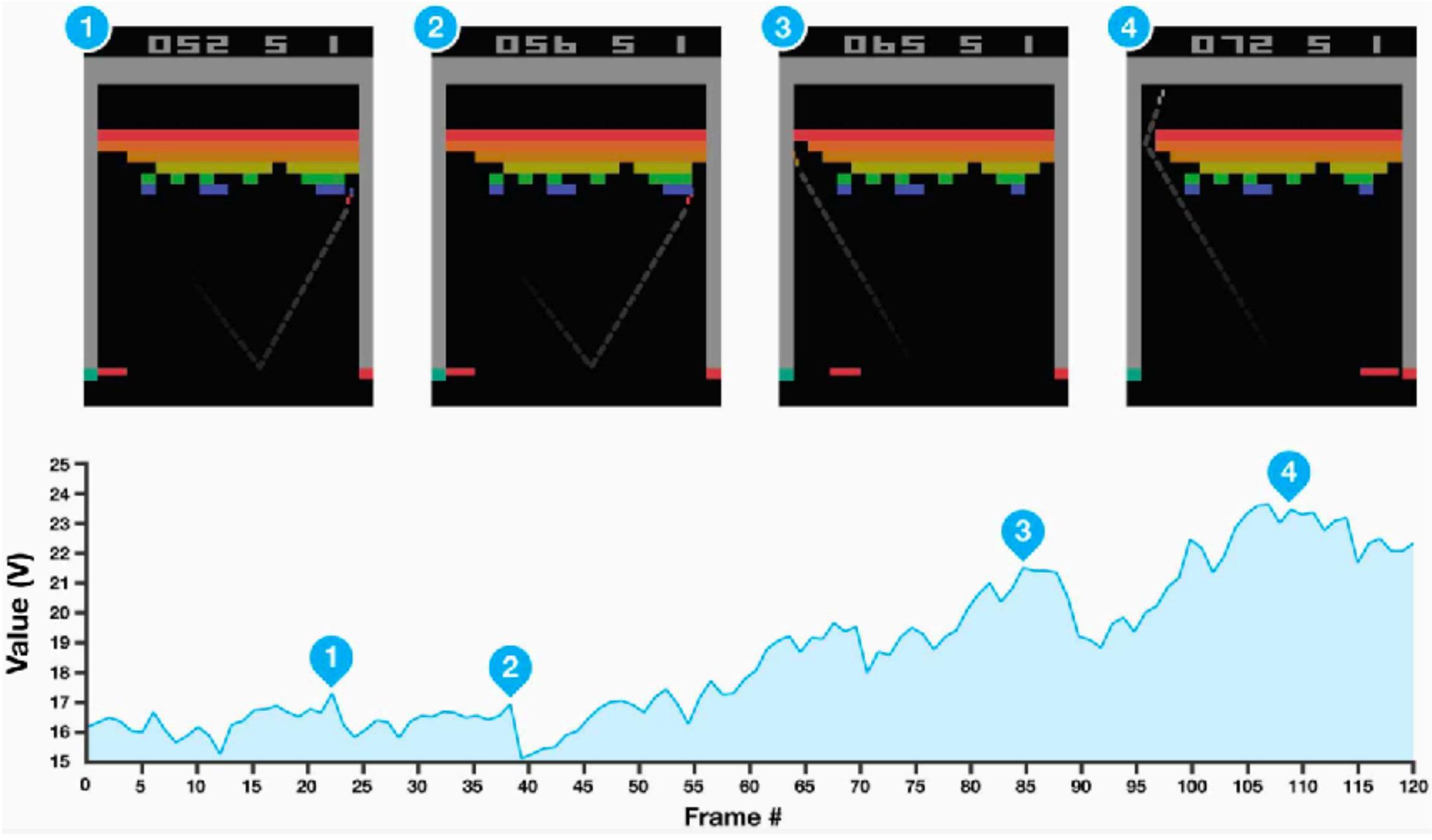
$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}\mathbf{a}_{t+1}} Q^{\pi}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$$

Bellman expectation equation

$$Q^*(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} \max_{\mathbf{a}_{t+1}} Q^*(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$$

Bellman optimality equation

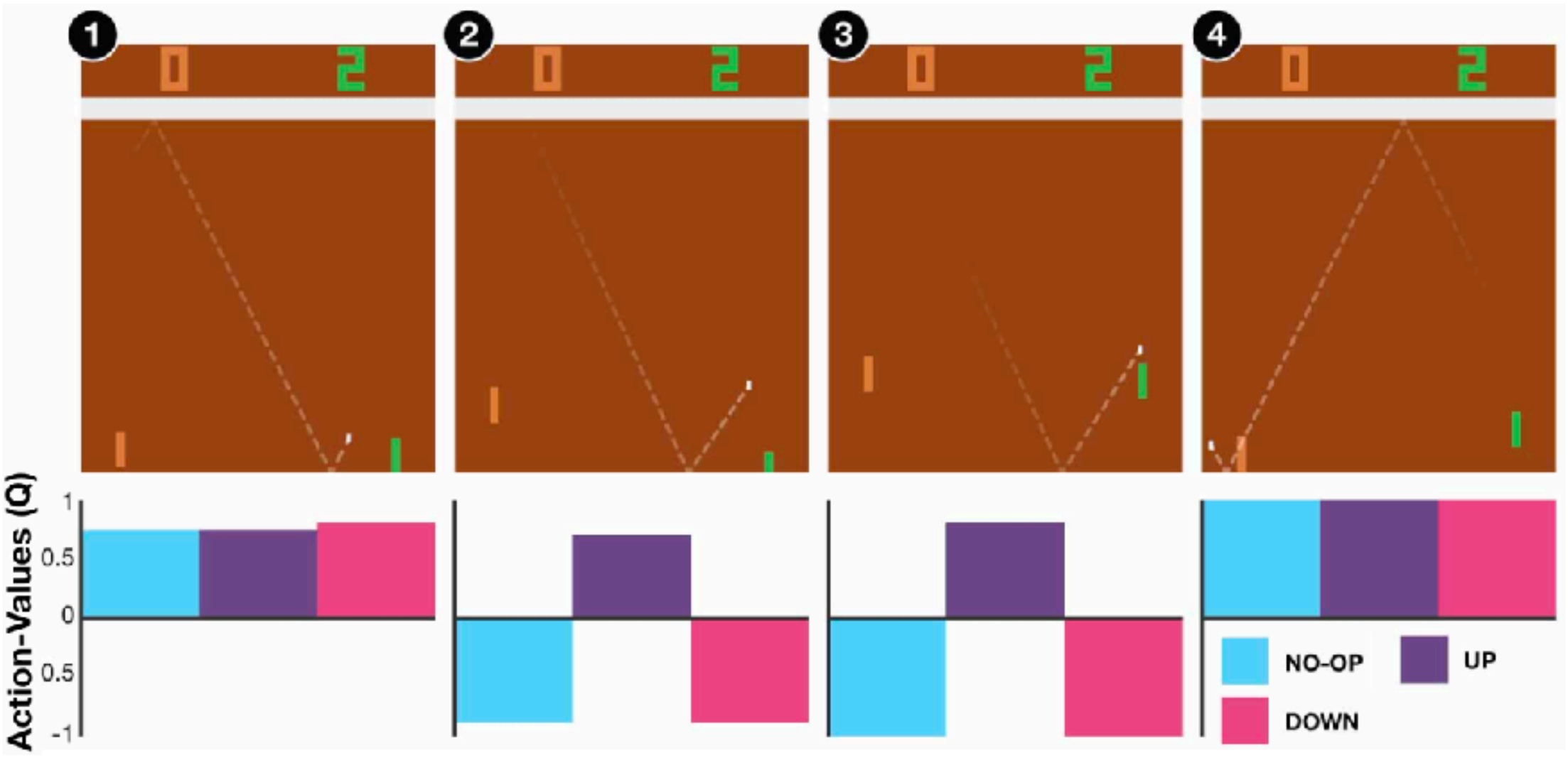
Value function in practice



https://www.youtube.com/watch?v=TmPfTpjtdgg

Source: Human-level control through deep reinforcement learning, Mnih et al. (Nature, 2015)

Q-function in practice



https://www.youtube.com/watch?v=riD029cVRA4

Source: Human-level control through deep reinforcement learning, Mnih et al. (Nature, 2015)

Homework 1:
Creating your own world

- Create your own environment with discrete action space

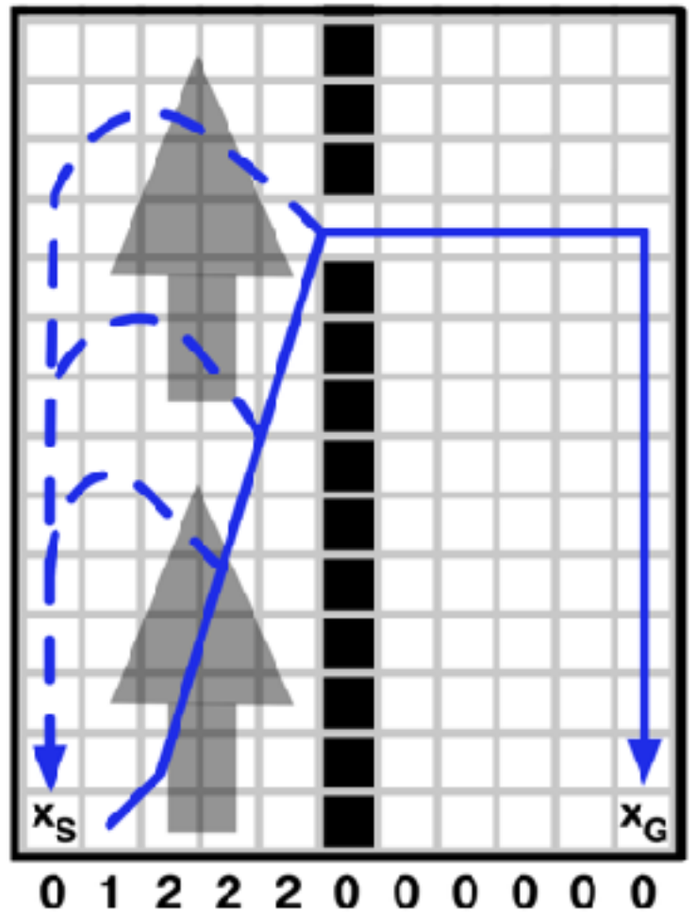
- Environment size (problem difficulty) should be adjustable

- The environment should include the adjustable stochastic component

- The environment should support both coordinate and visual representation of the states

Example: Windy Grid World

- Each state has probability 0.1 of moving in a random direction
- The transition is affected by the wind moving the agent northward
- The reward for transitioning to the goal state $\mathbf{x_G}$ is 1, otherwise 0



Code specification

```
class Environment:

    def __init__(
            self,
            world_size=(10, 10),
            stochasticity=0.1,
            visual=False):
        """
        Parameters
        ----------
        world_size: size of the environment (problem difficulty)
        stochasticity: amount of stochasticity
        visual: if True, images of the world are considered states
        """
        ...

    def reset(self):
        """
        Sample state from initial states distribution s ~ p(s_0)
        and set the agent into this state. Return s.
        """
        ...
        return state

    def step(self, action):
        """
        Take the action in the environment. Return next state, obtained
        immediate reward, and indicator whether the episode is finished.
        """
        ...
        return next_state, reward, done
```