

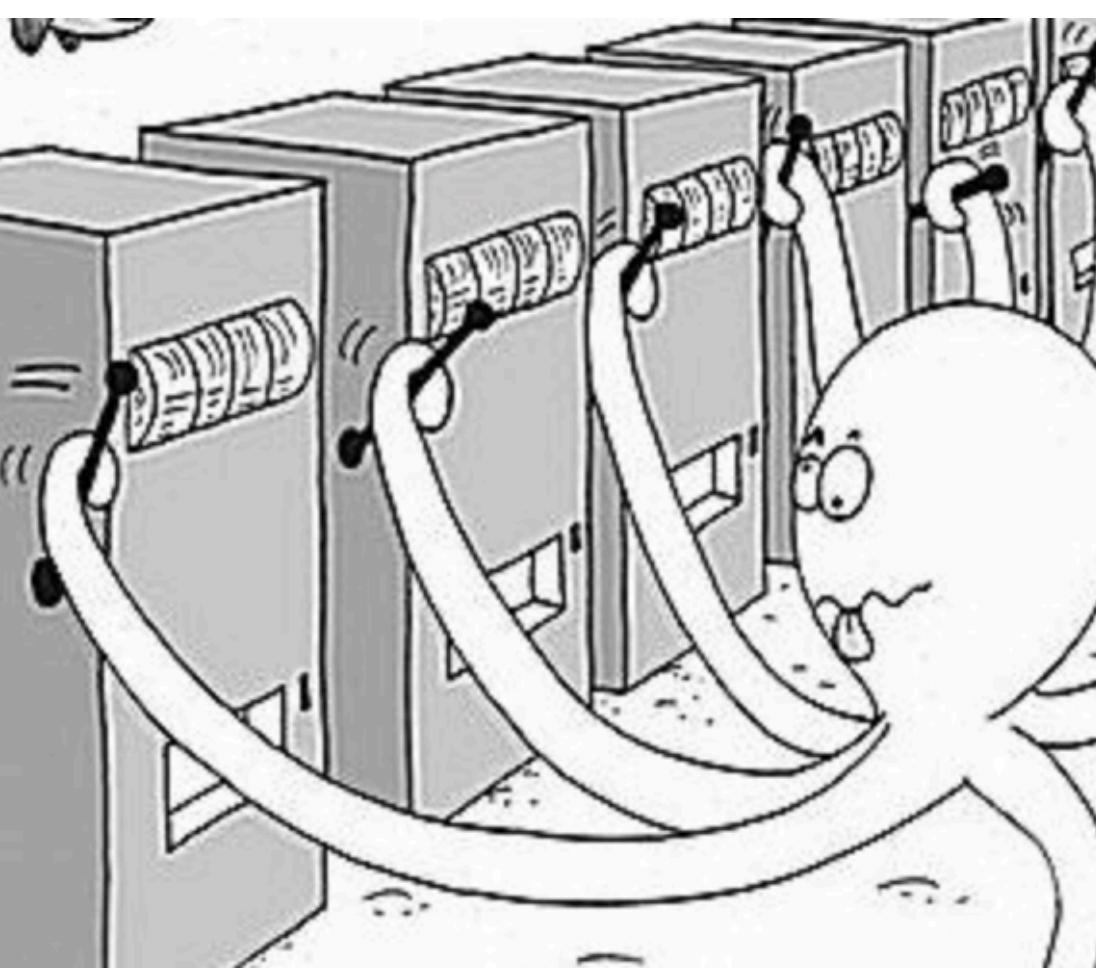
An introduction to Reinforcement Learning

21st of June 2022

Recap: Multi-armed bandits

Greedy action selection:

$$P(a_t = a) = \begin{cases} 1 & \text{if } a_t = \operatorname{argmax}_a V_t(a) \\ 0 & \text{otherwise} \end{cases}$$



Softmax action selection:

$$P(a_t = a) = \frac{e^{V_t(a) \cdot \beta}}{\sum_{i=1}^N e^{V_t(a_i) \cdot \beta}}$$

Action is governed by a **policy**:

$$\pi(a, s) = P(a_t = a \mid s_t = s)$$

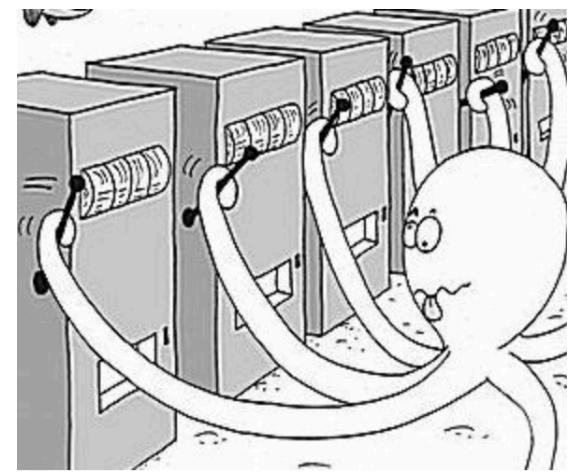
Epsilon-greedy action selection:

$$P(a_t = a) = \begin{cases} 1 - \epsilon & \text{if } a_t = \operatorname{argmax}_a V_t(a) \\ \epsilon/N & \text{otherwise} \end{cases}$$

Upper-confidence-bound (UCB) action selection:

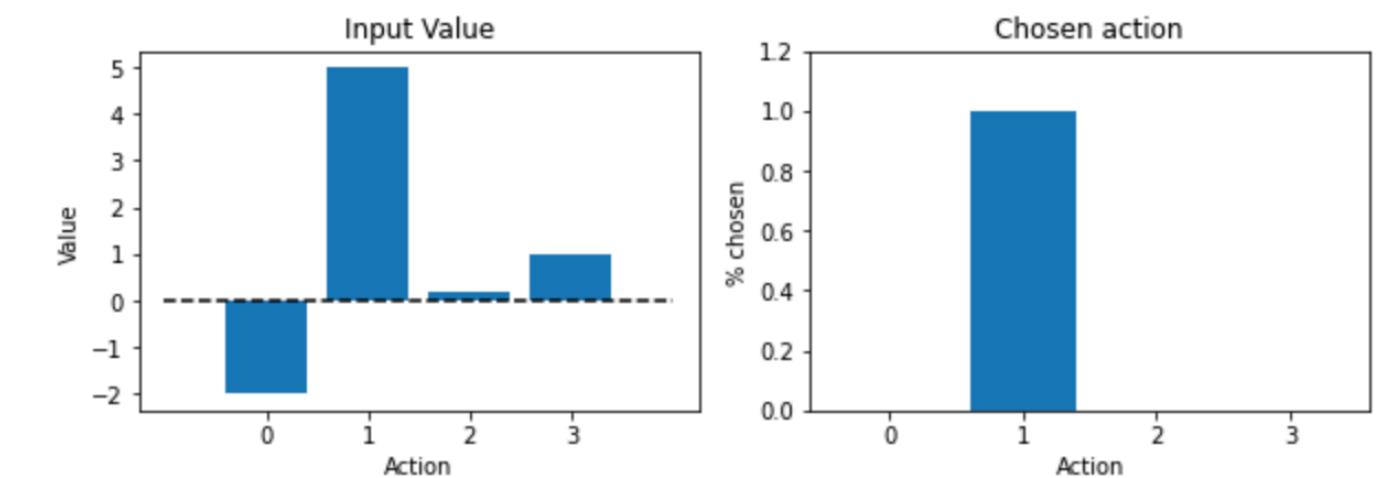
$$P(a_t = a) = \operatorname{argmax}_a [V_t(a) + c \cdot \sqrt{\frac{\ln t}{N_t(a)}}]$$

Performance comparison

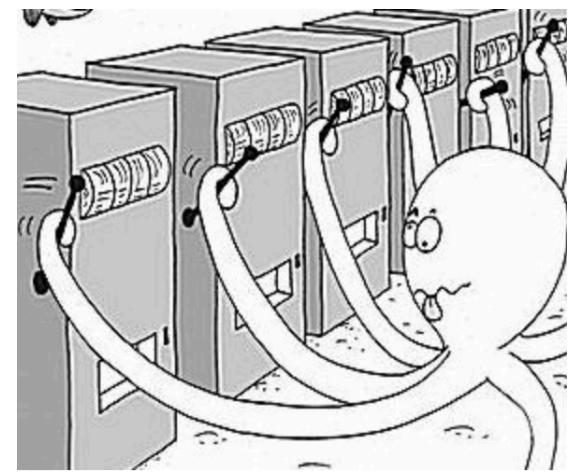


N arms = 10
N simulations = 1000

Greedy action selection:
$$P(a_t = a) = \begin{cases} 1 & \text{if } a_t = \operatorname{argmax}_a V_t(a) \\ 0 & \text{otherwise} \end{cases}$$



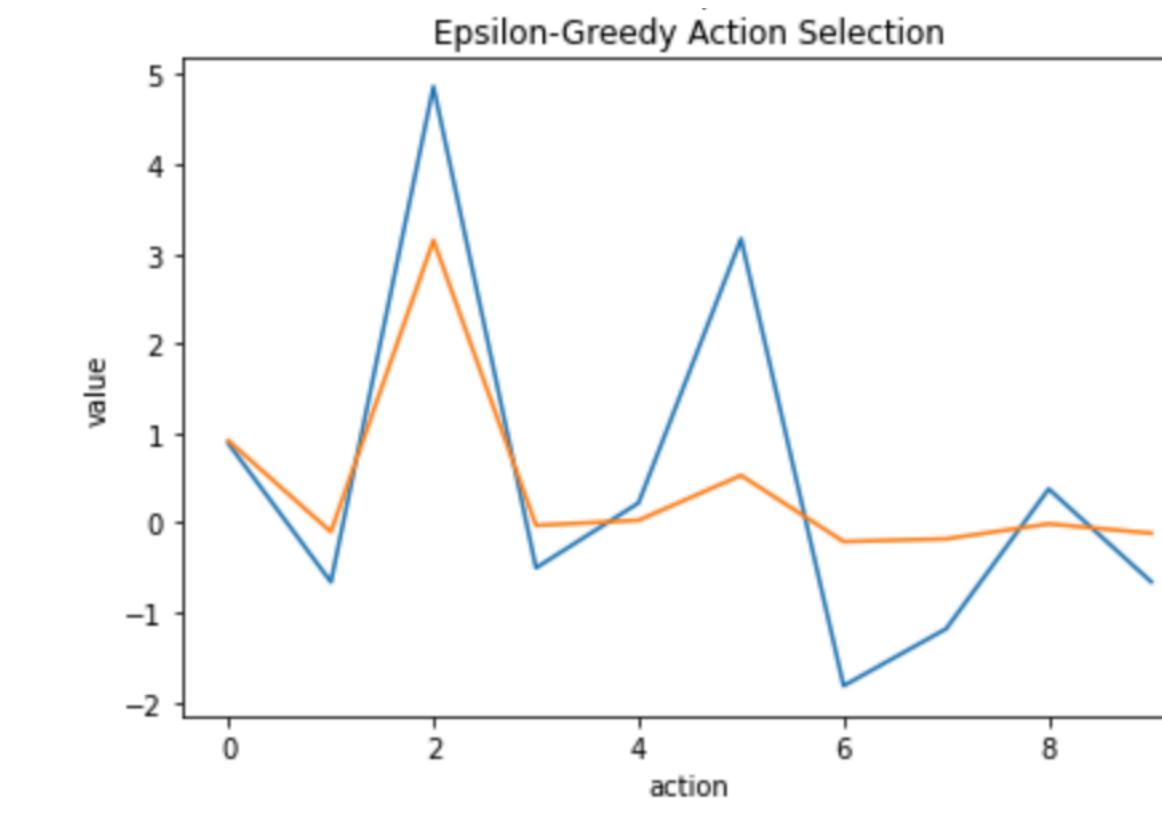
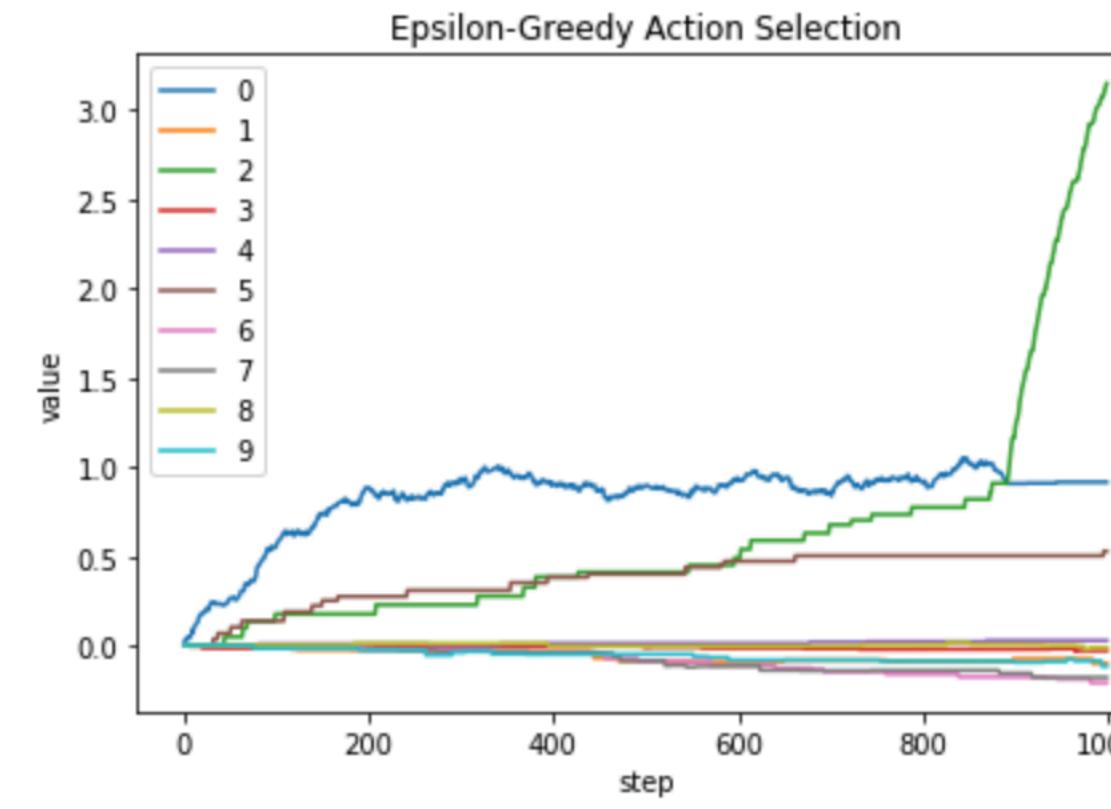
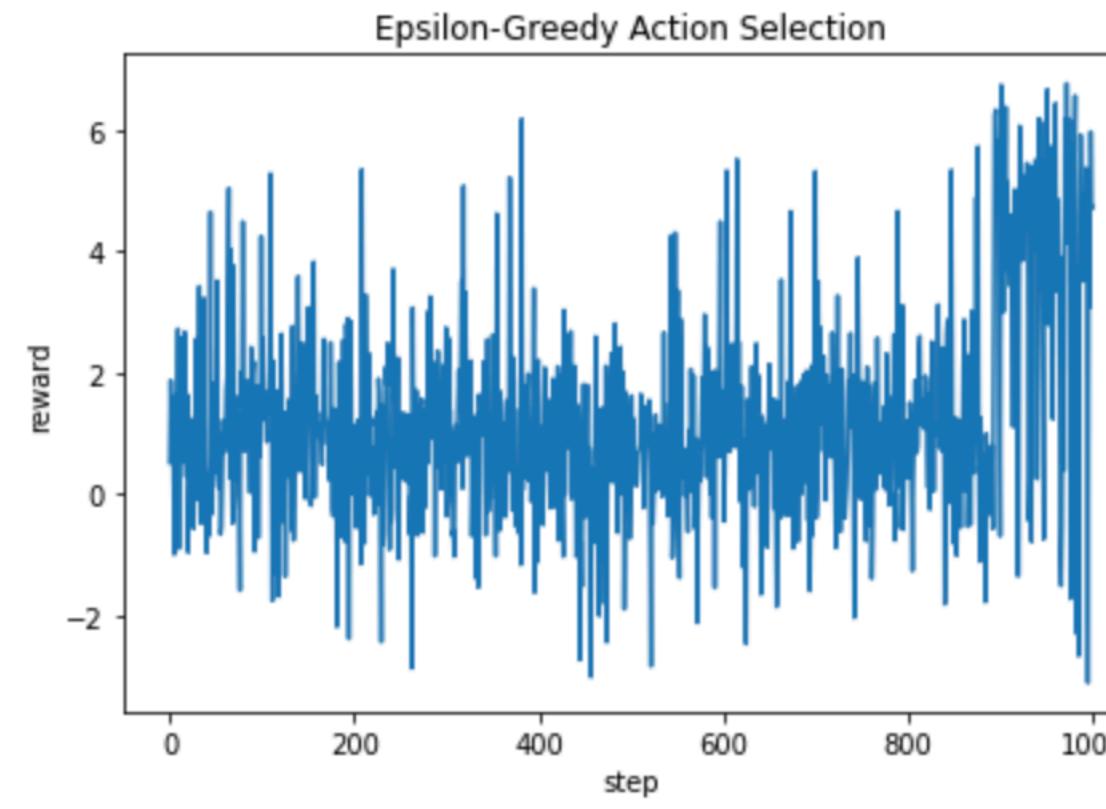
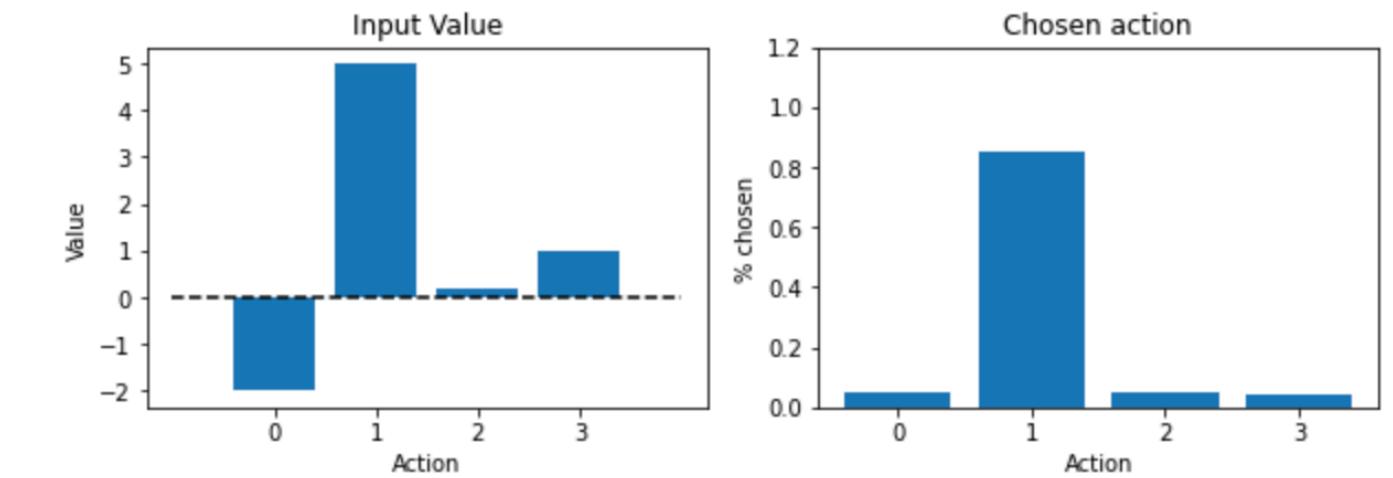
Performance comparison



N arms = 10
N simulations = 1000

Epsilon-greedy action selection:

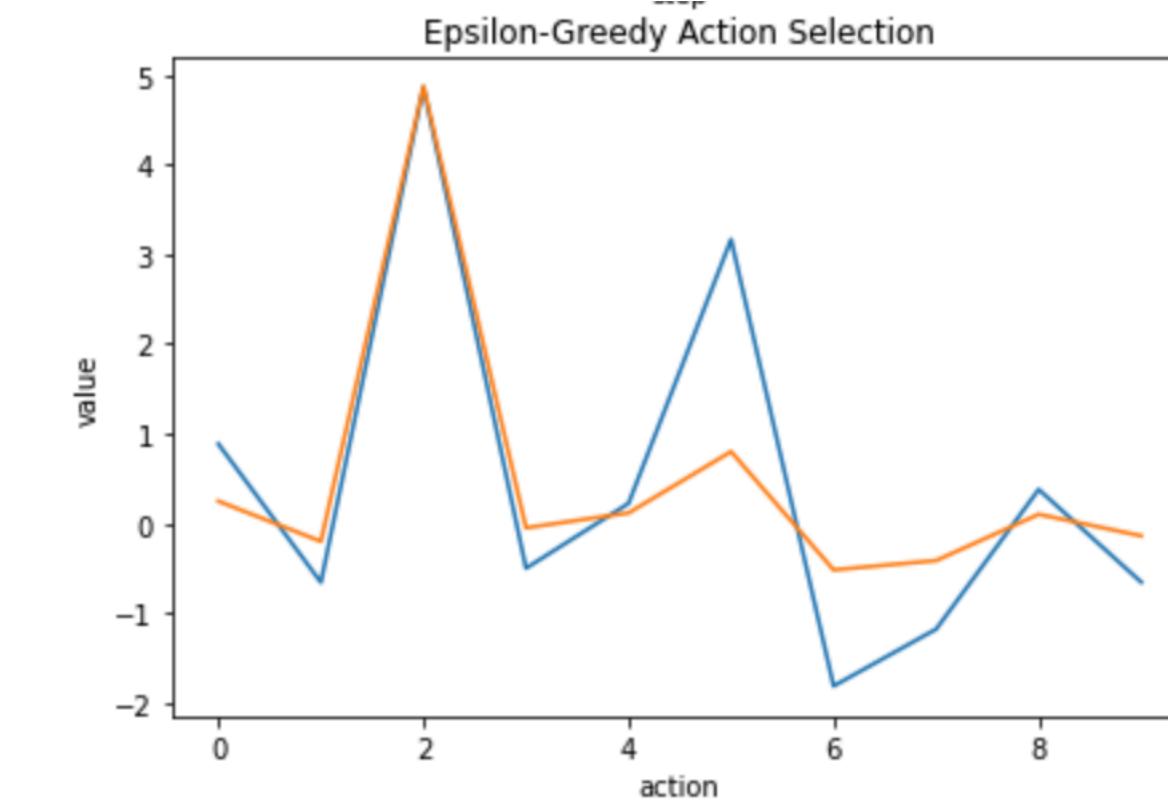
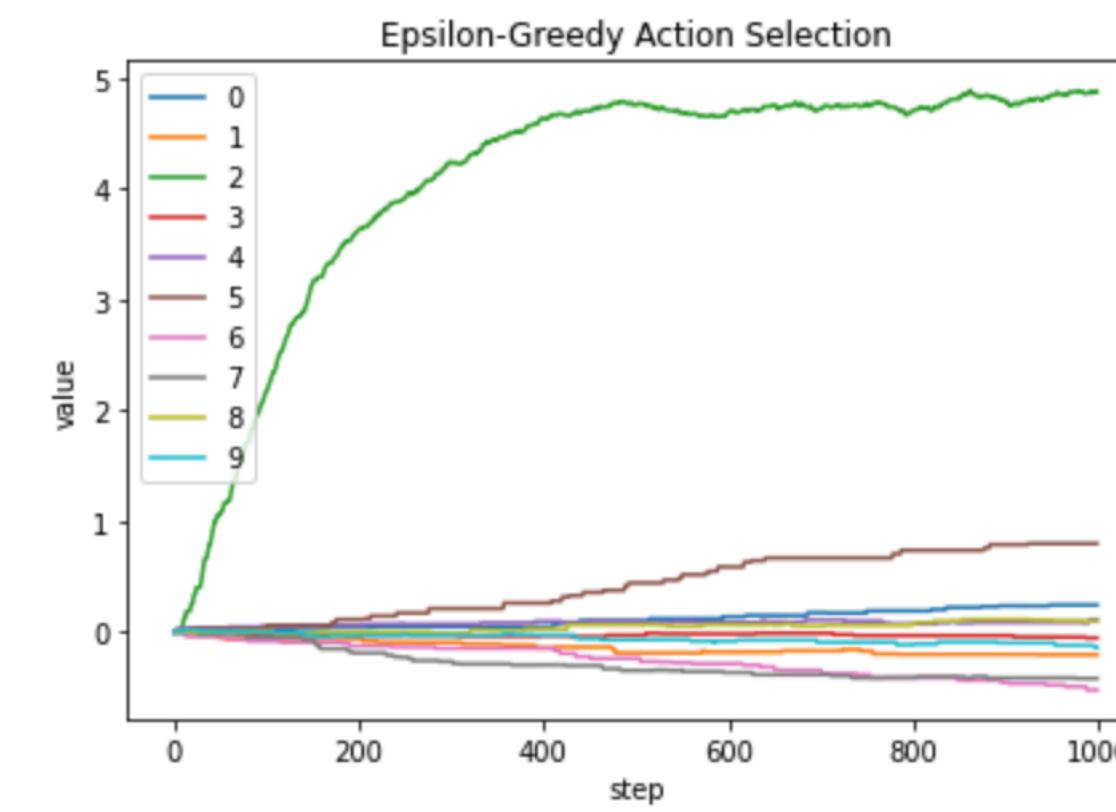
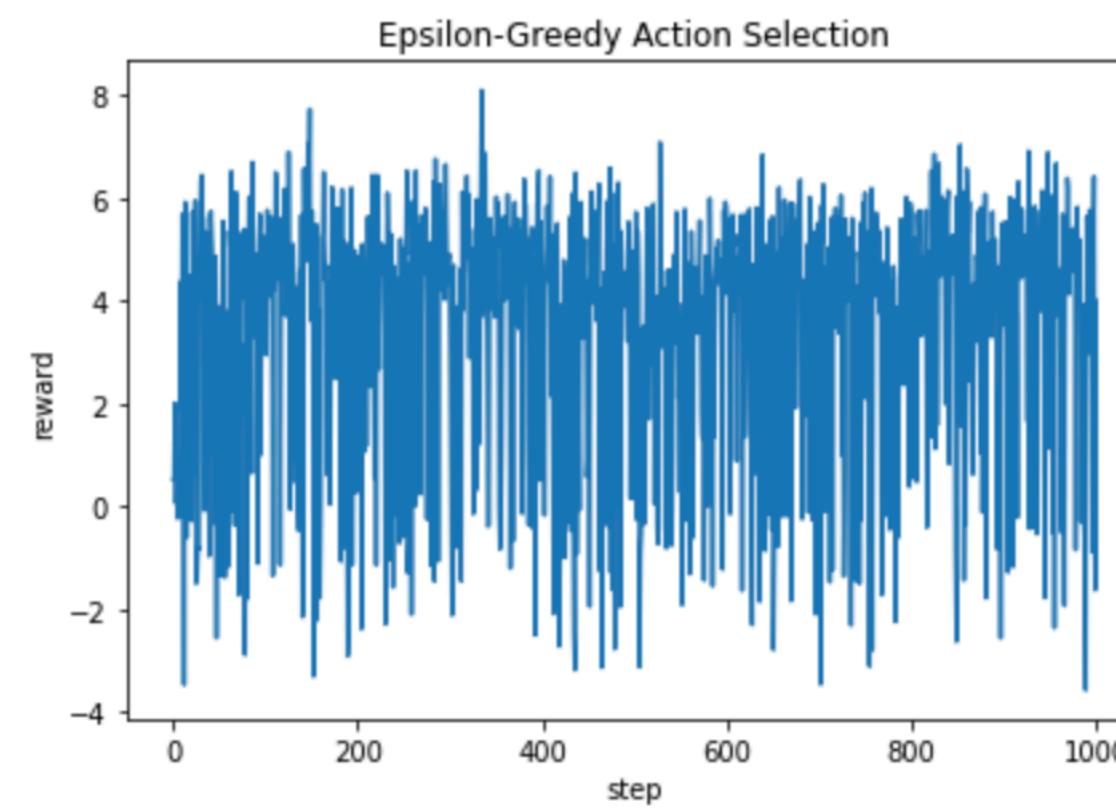
$$P(a_t = a) = \begin{cases} 1 - \epsilon & \text{if } a_t = \operatorname{argmax}_a V_t(a) \\ \epsilon/N & \text{otherwise} \end{cases}$$



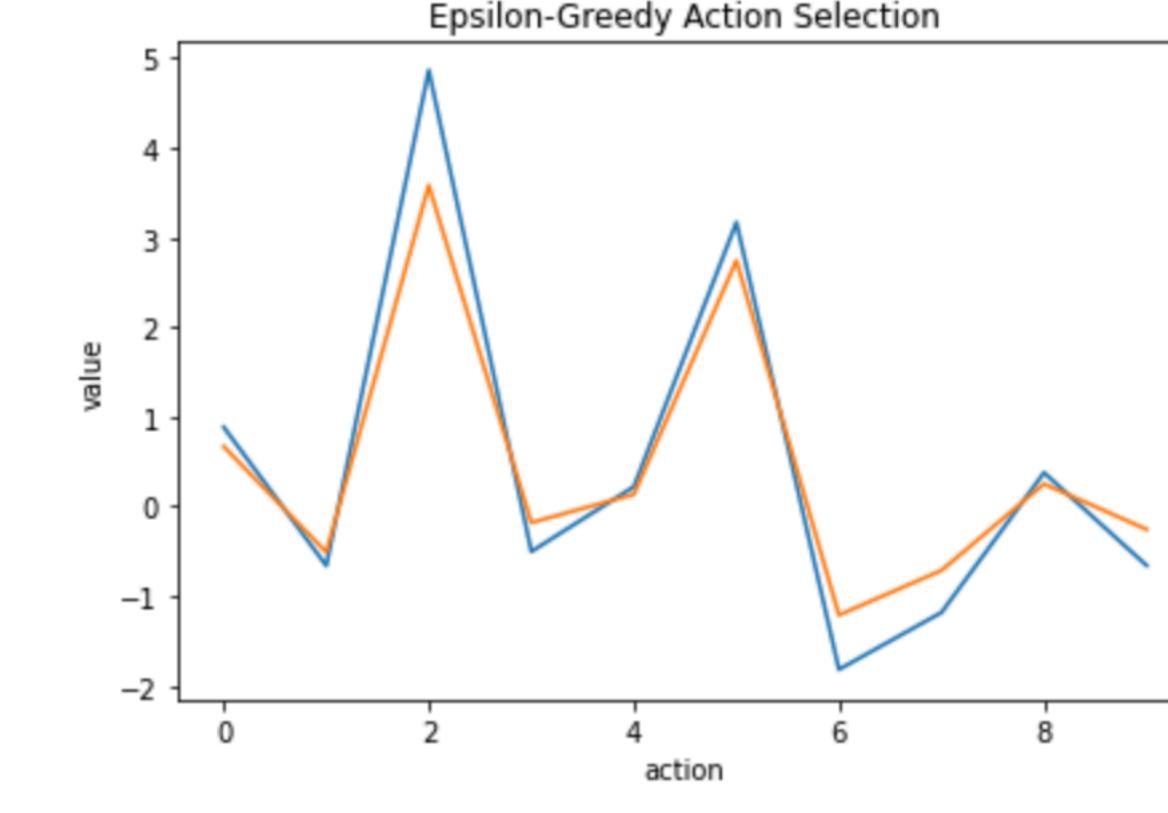
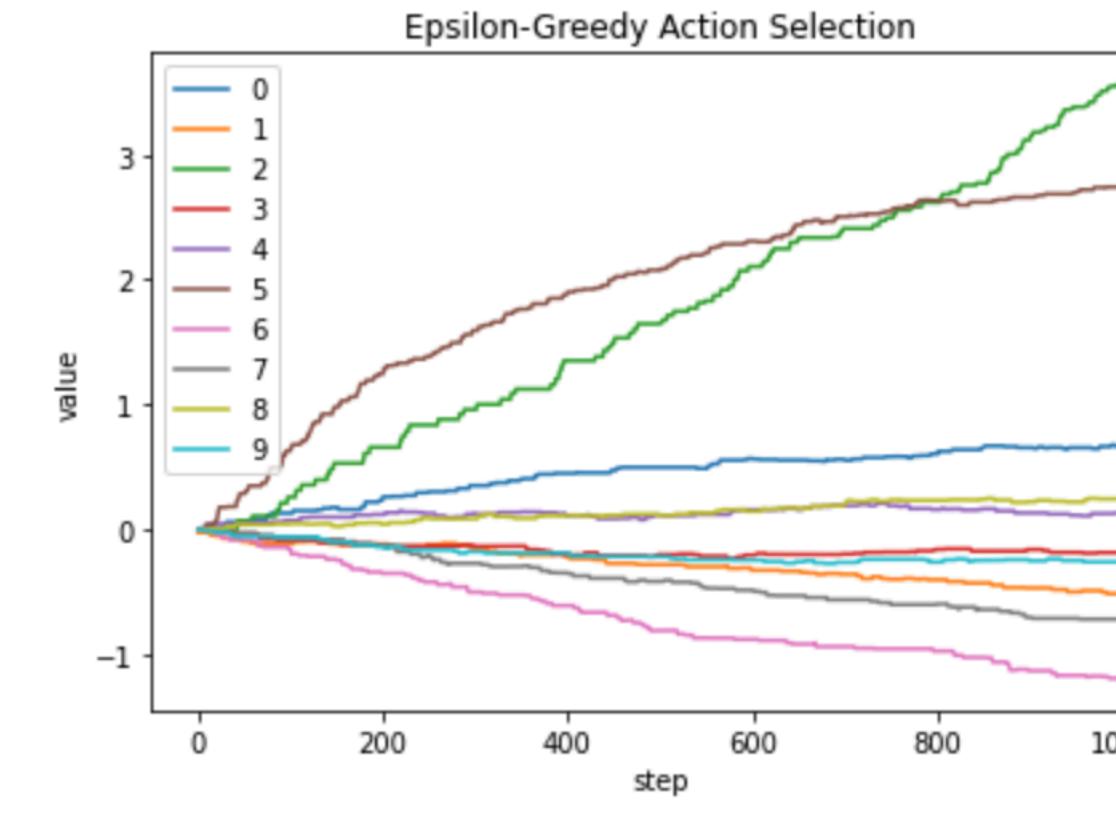
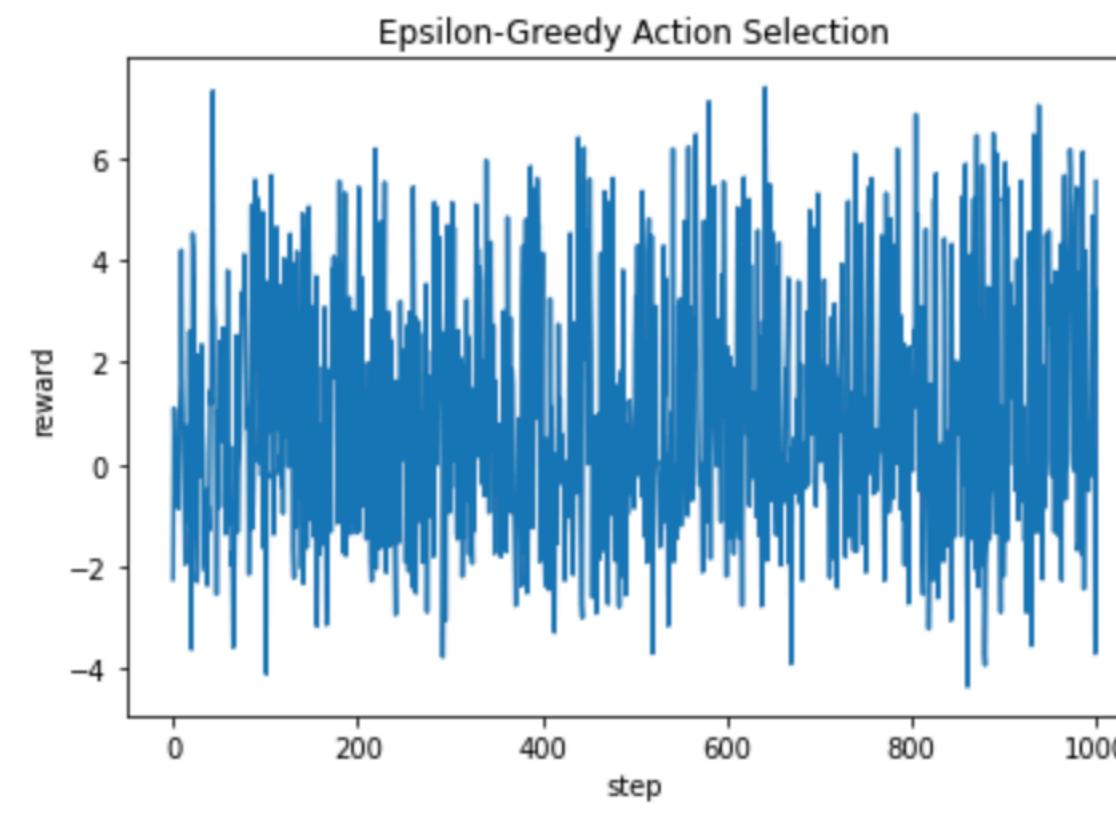
How will your choice of ϵ affect these plots? (Here: $\epsilon = 0.15$)

Perfomance comparison

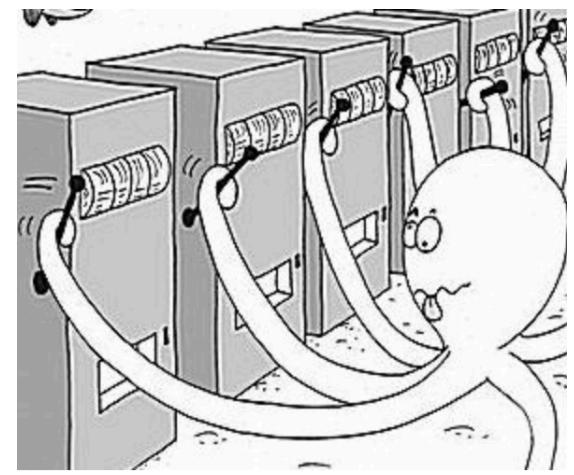
$\epsilon = 0.35$



$\epsilon = 0.85$



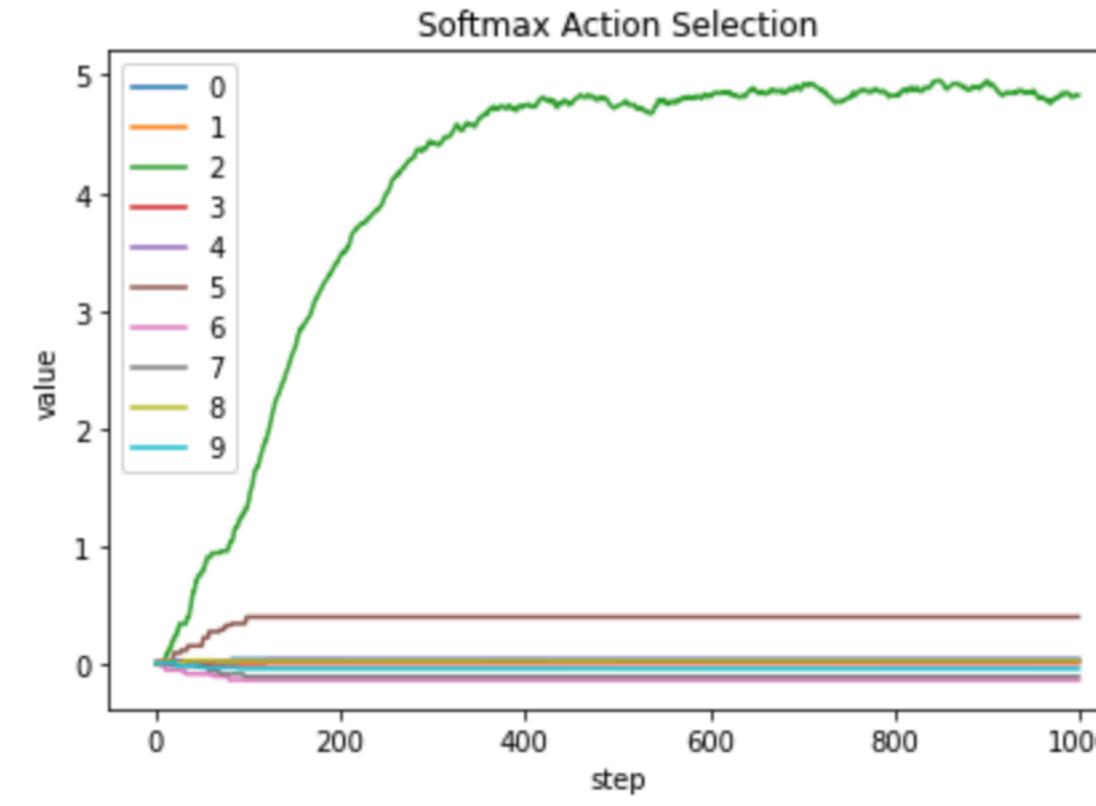
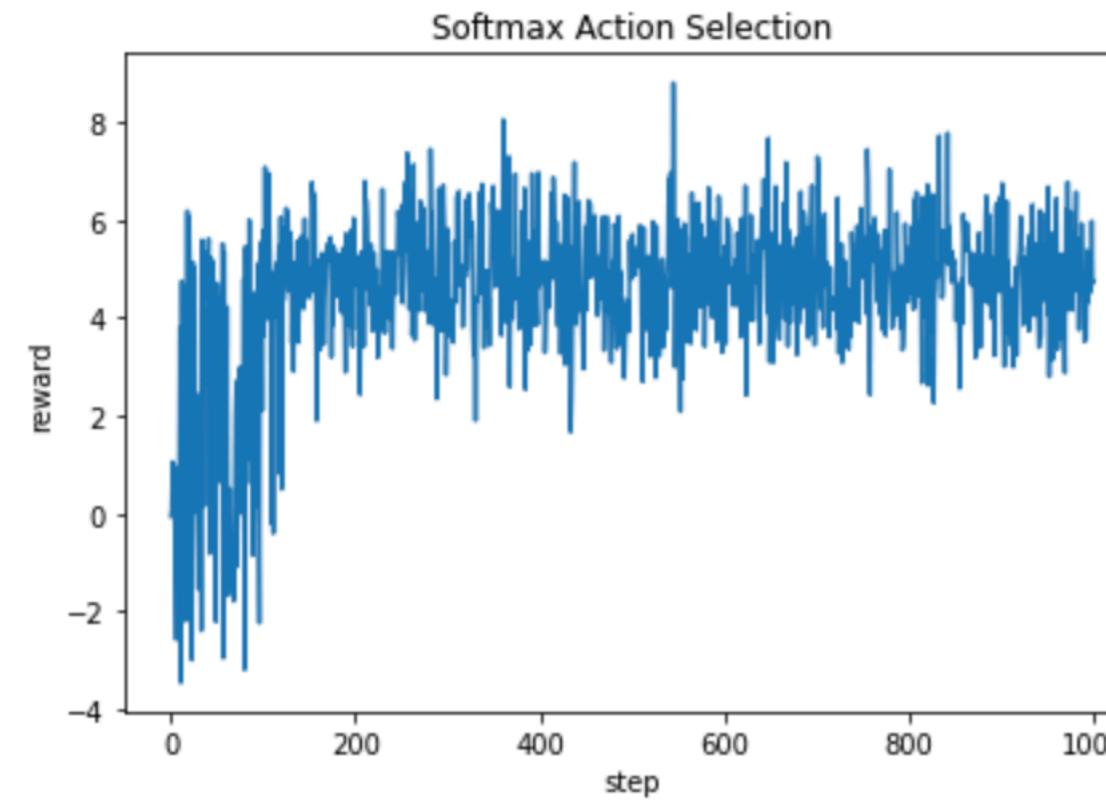
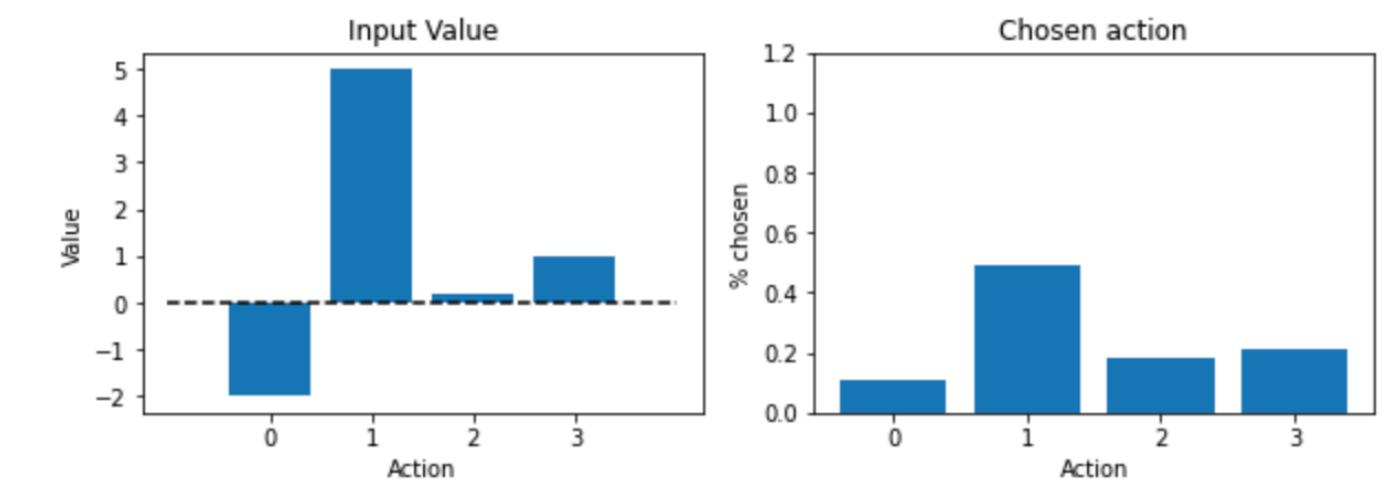
Performance comparison



N arms = 10
N simulations = 1000

Softmax action selection:

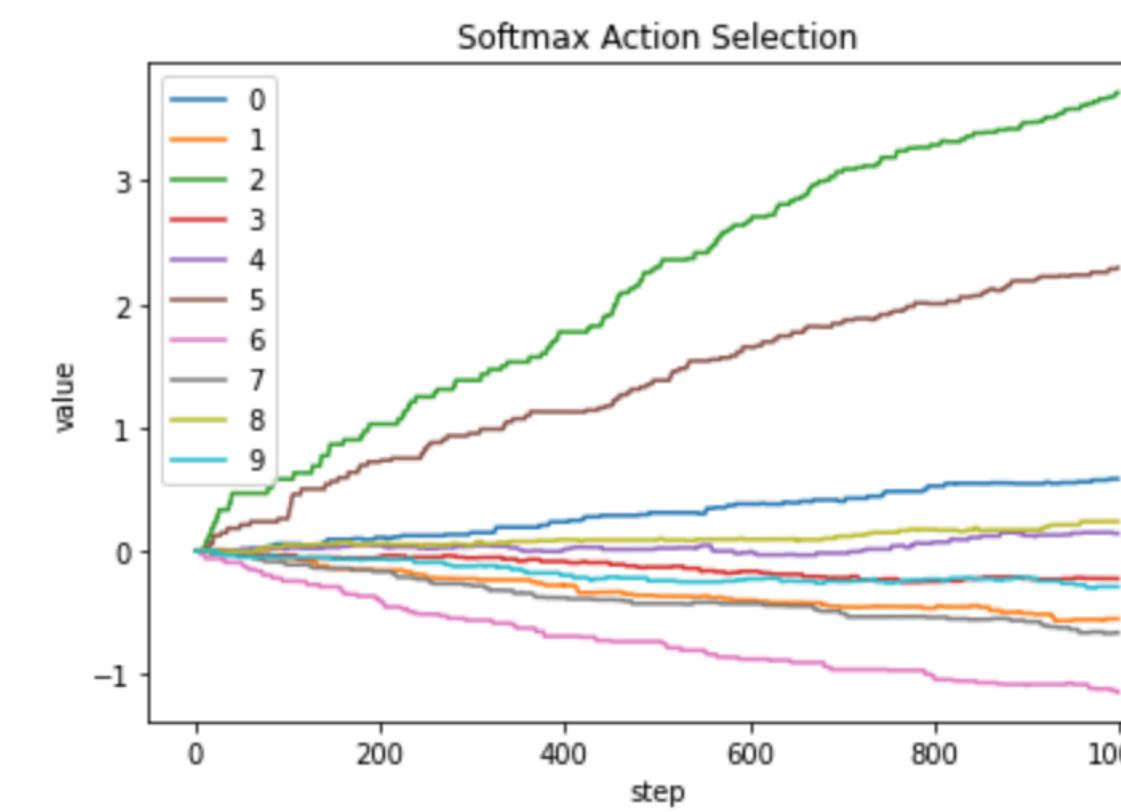
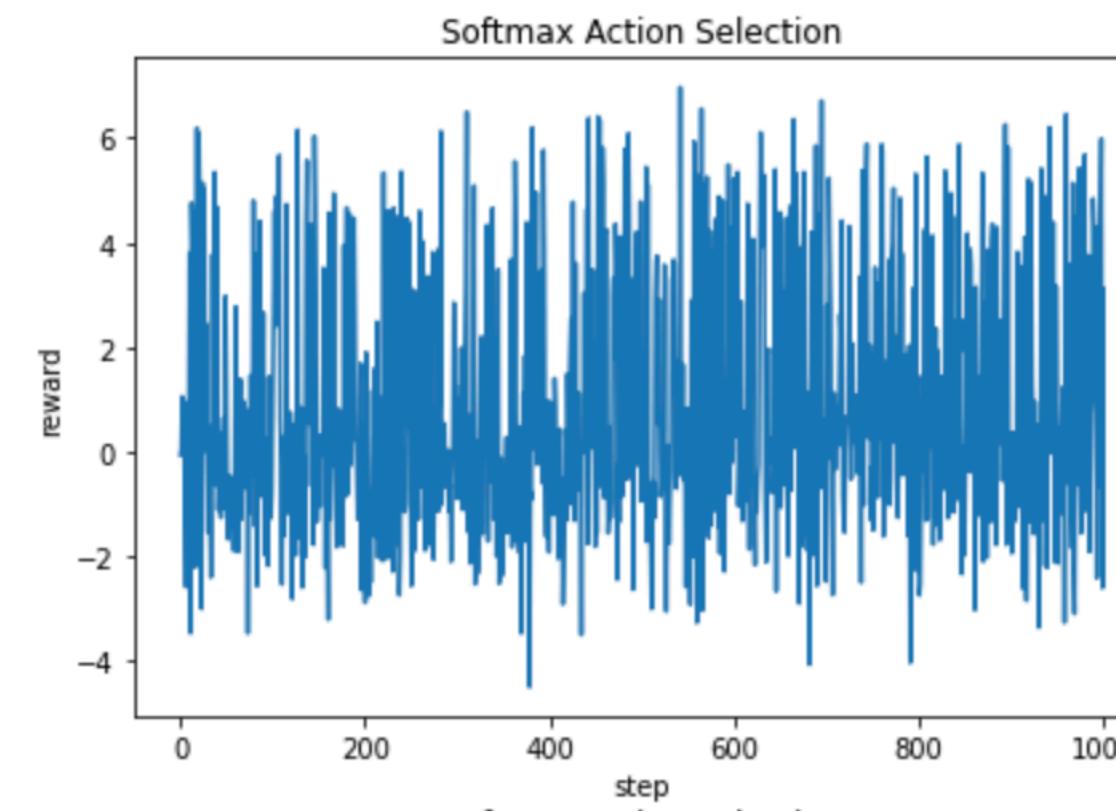
$$P(a_t = a) = \frac{e^{V_t(a) \cdot \beta}}{\sum_{i=1}^N e^{V_t(a_i) \cdot \beta}}$$



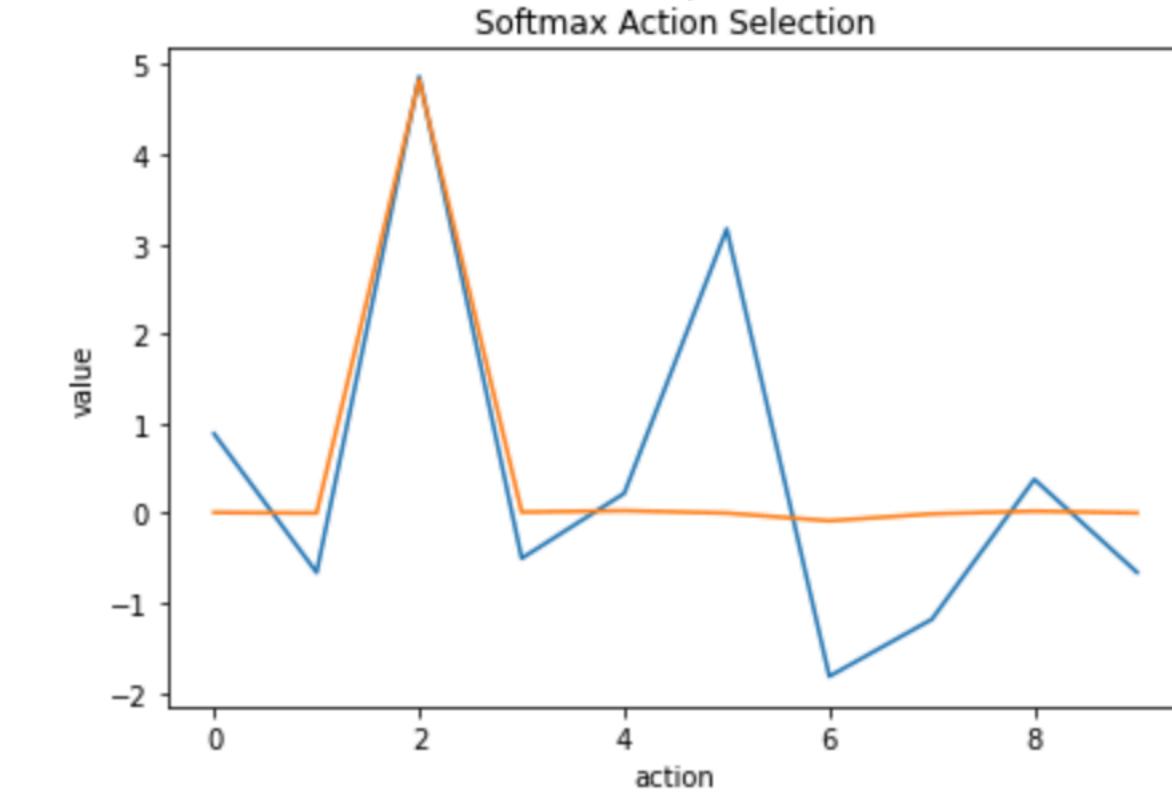
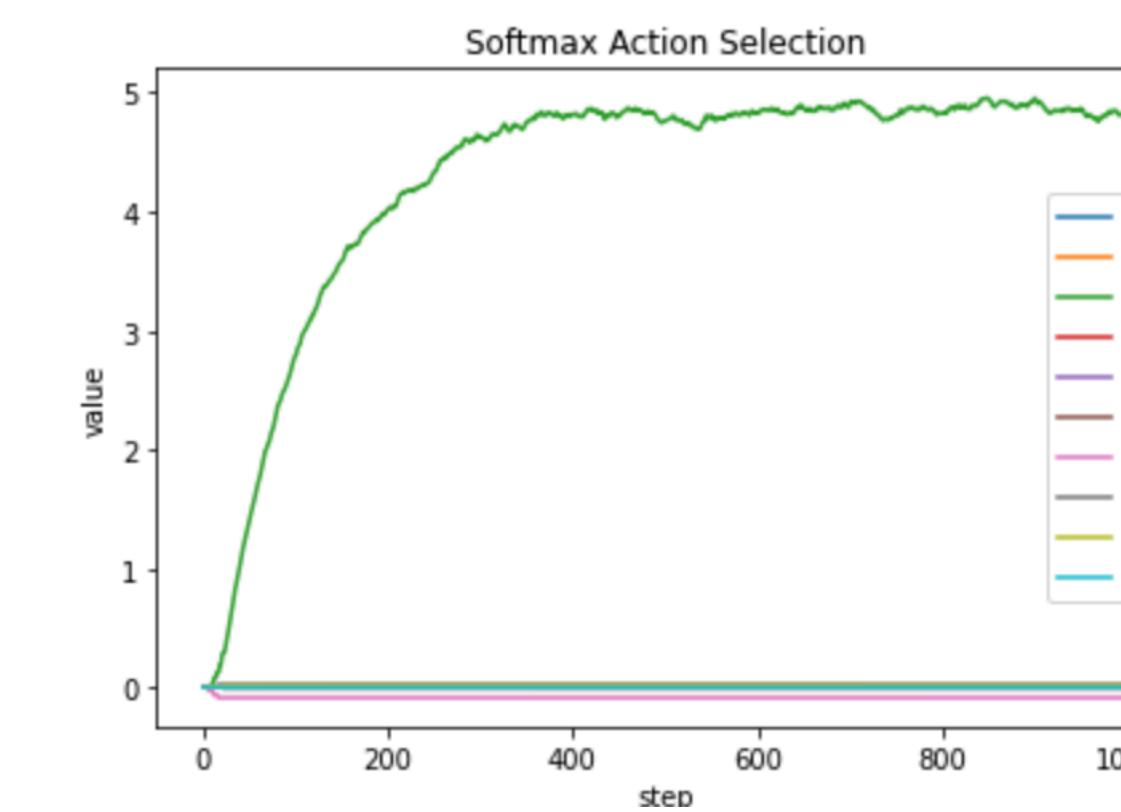
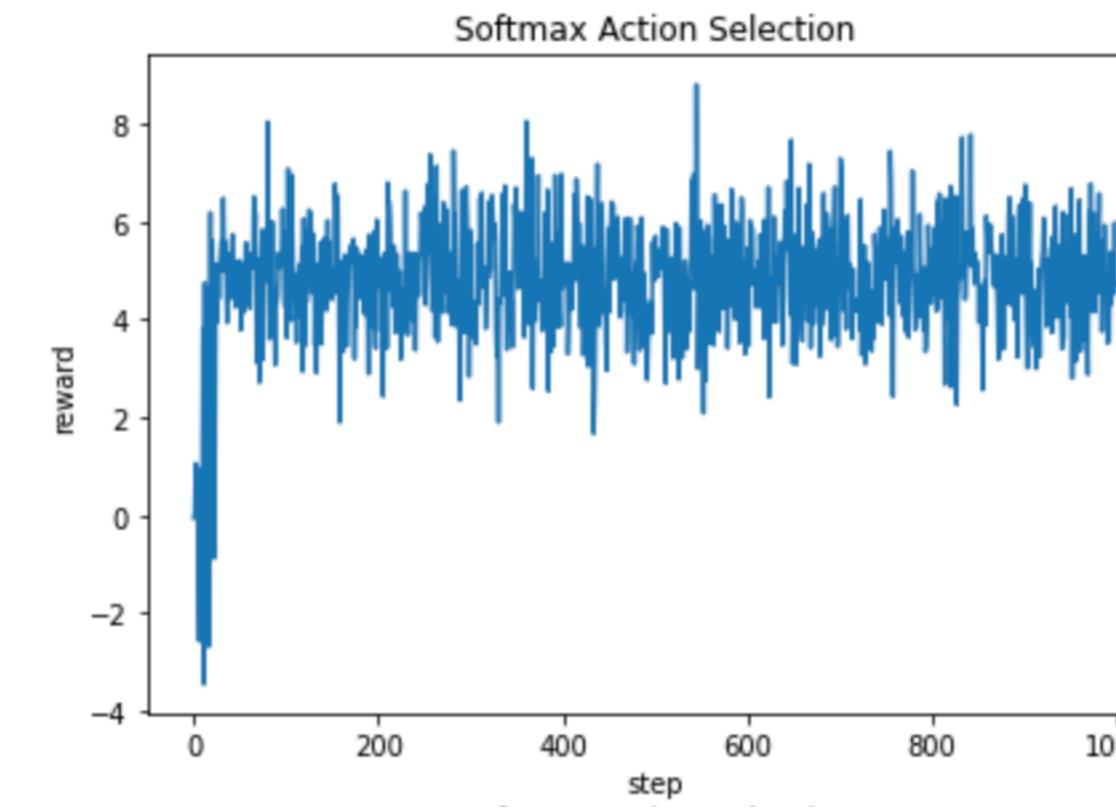
How will your choice of β affect these plots? (Here: $\beta = 2$)

Perfomance comparison

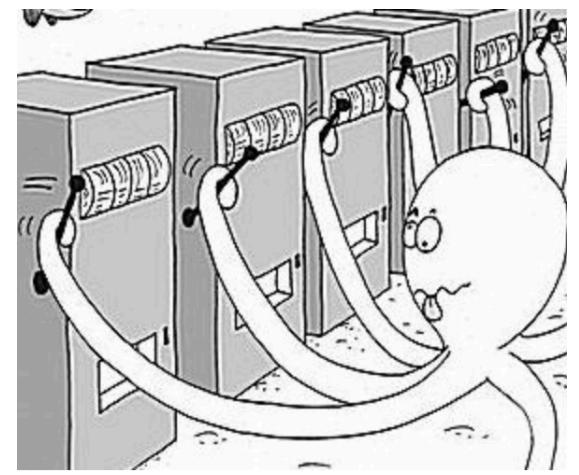
$$\beta = 0.2$$



$$\beta = 10$$



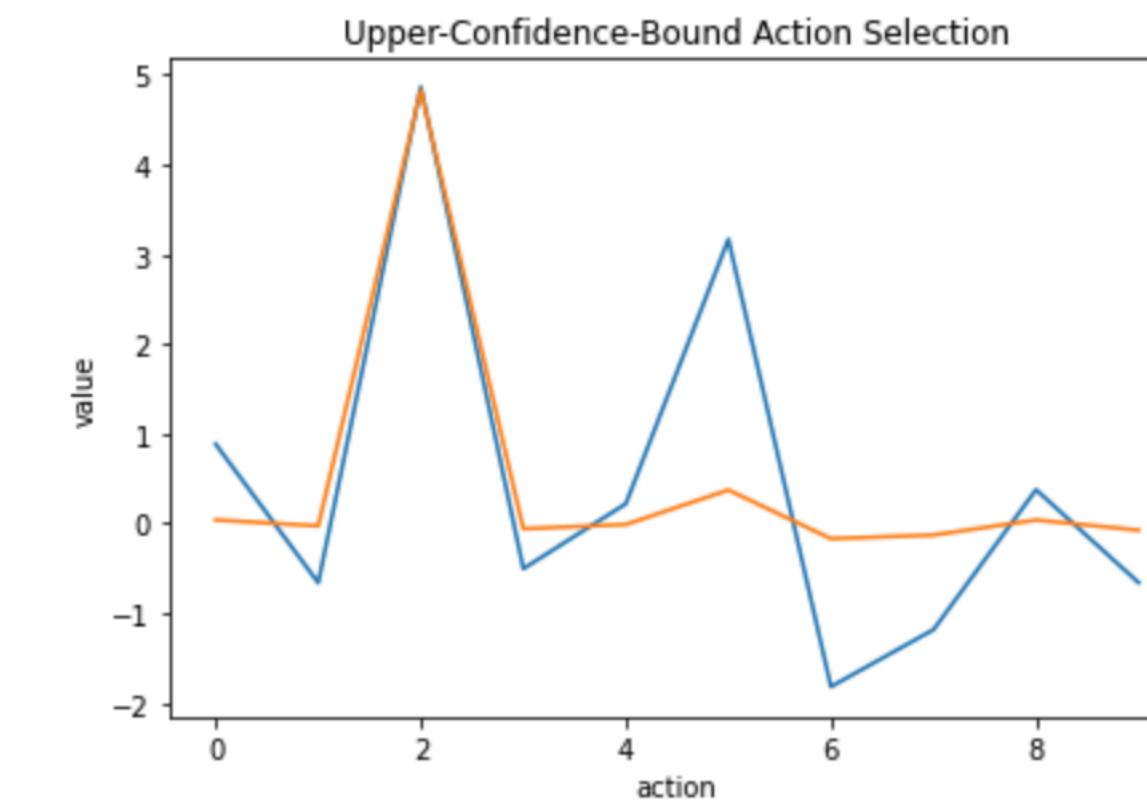
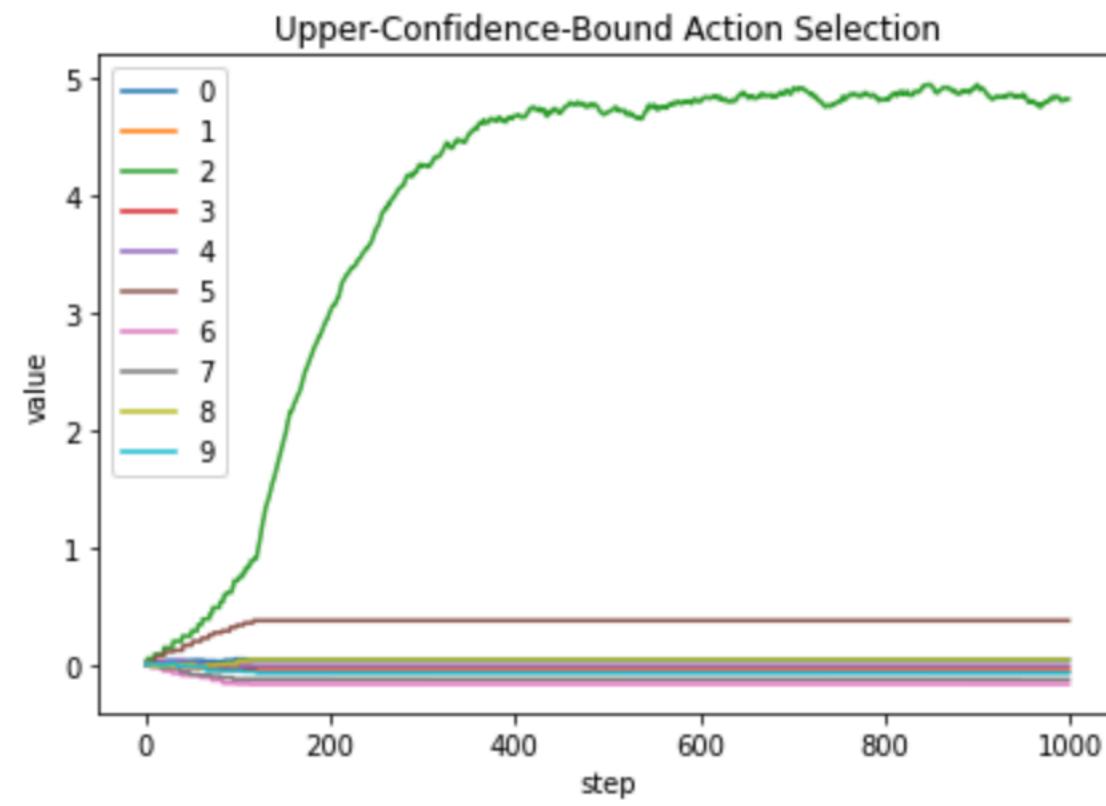
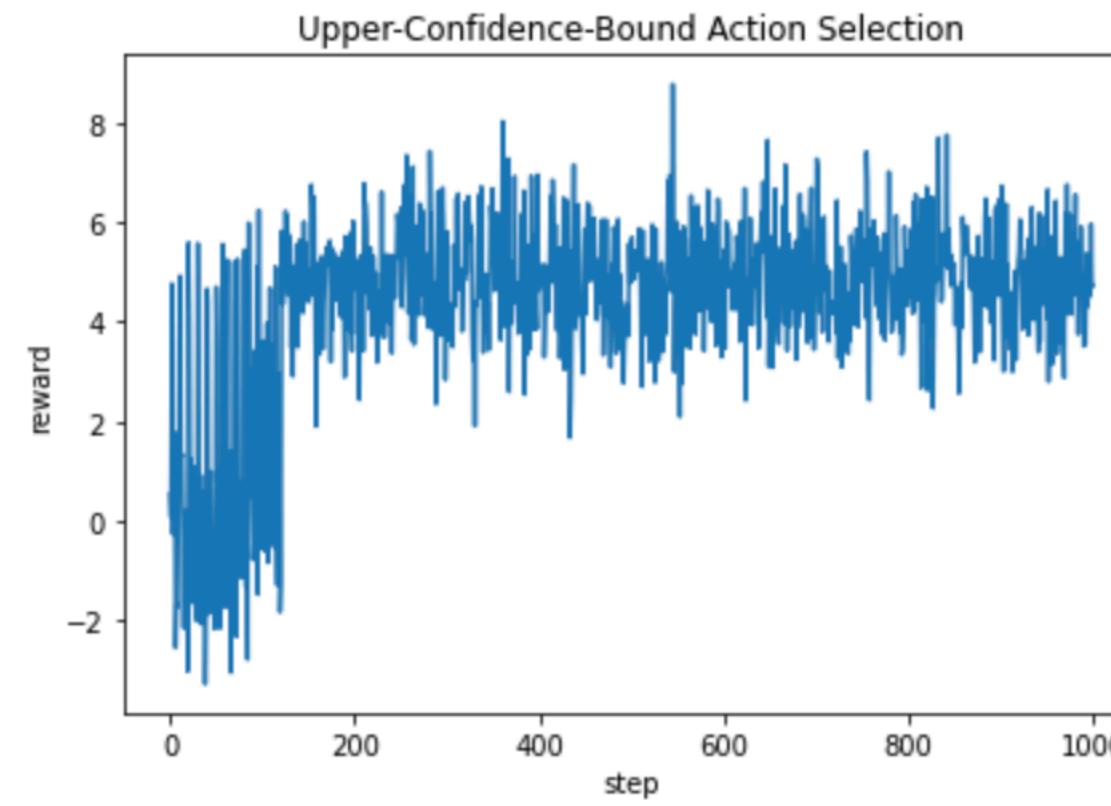
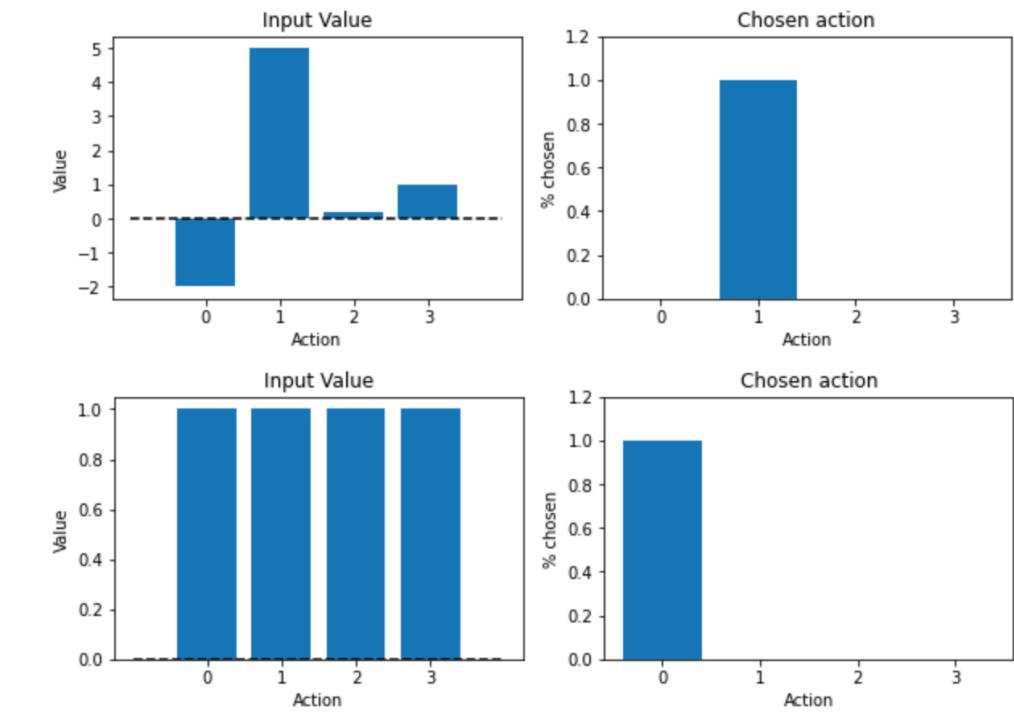
Performance comparison



N arms = 10
N simulations = 1000

**Upper-confidence-bound
(UCB) action selection:**

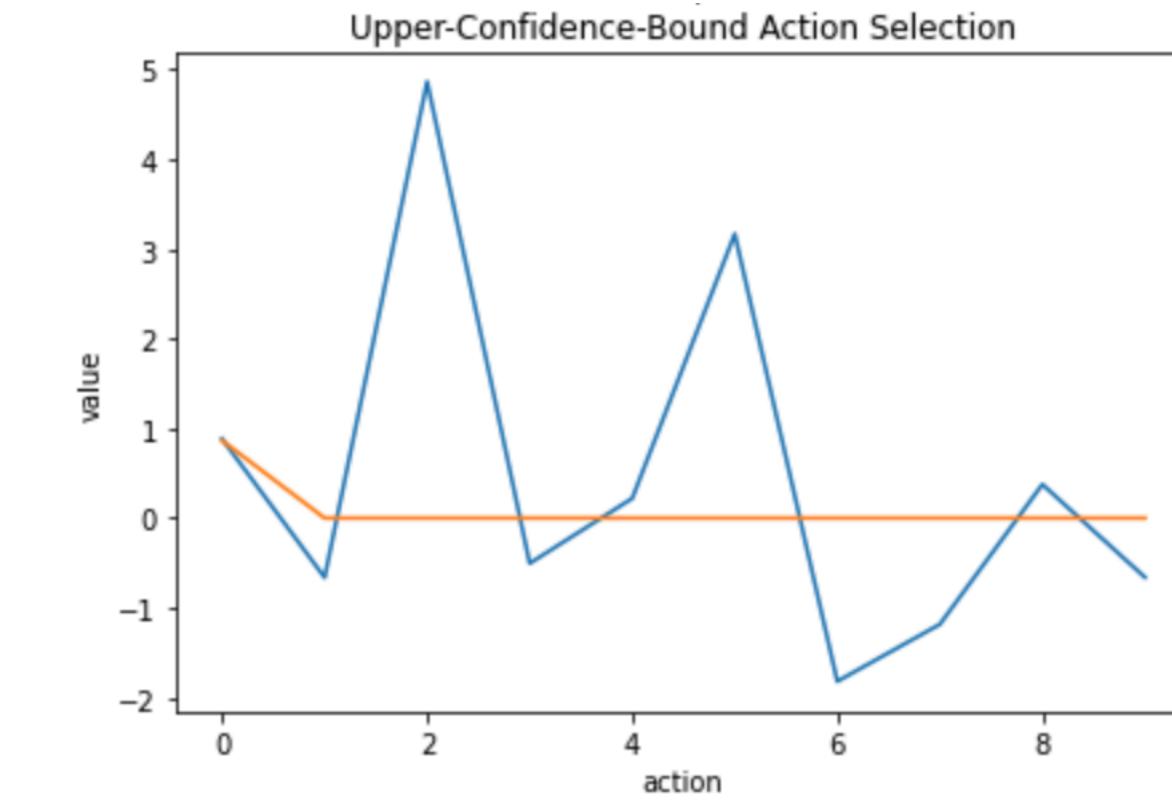
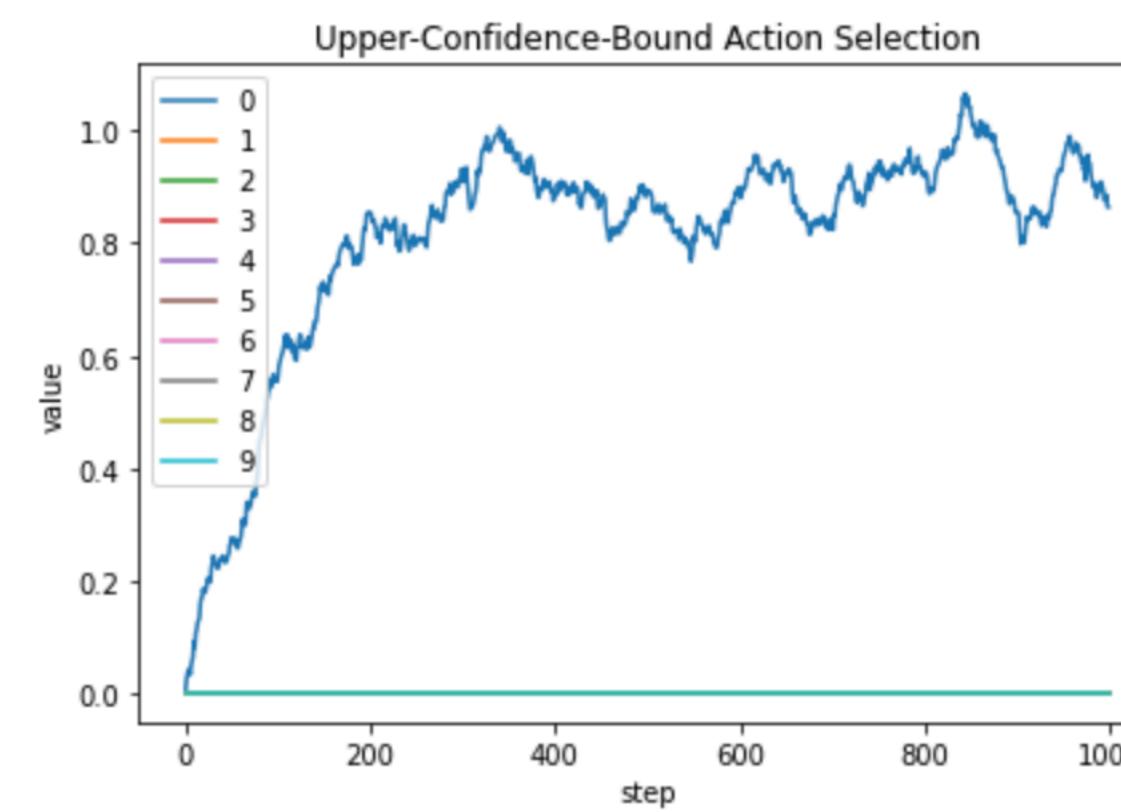
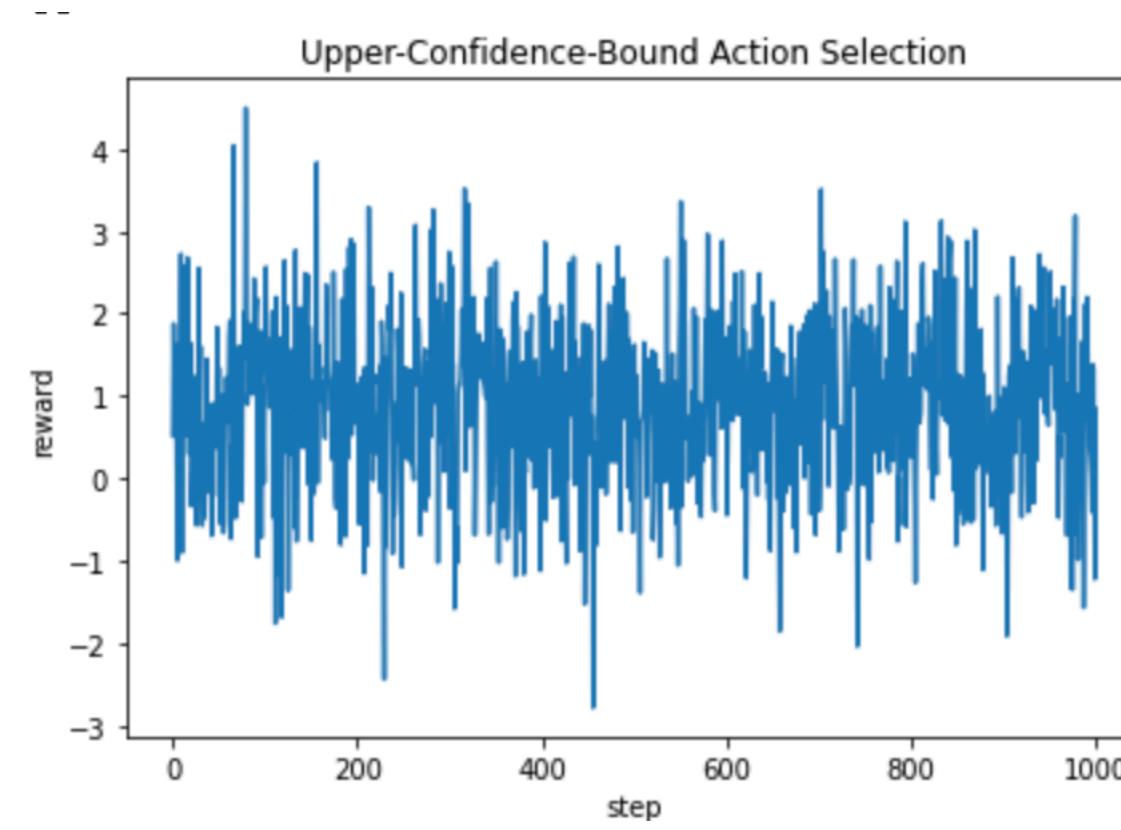
$$P(a_t = a) = \operatorname{argmax}_a [V_t(a) + c \cdot \sqrt{\frac{\ln t}{N_t(a)}}]$$



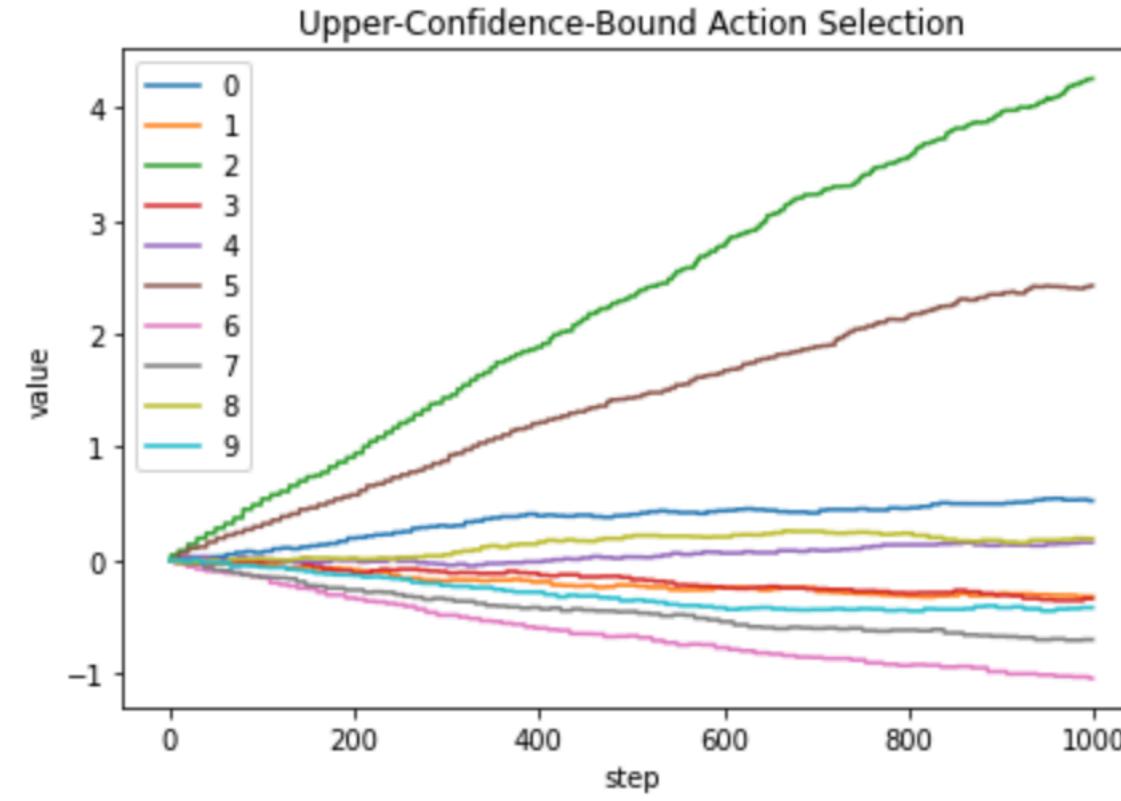
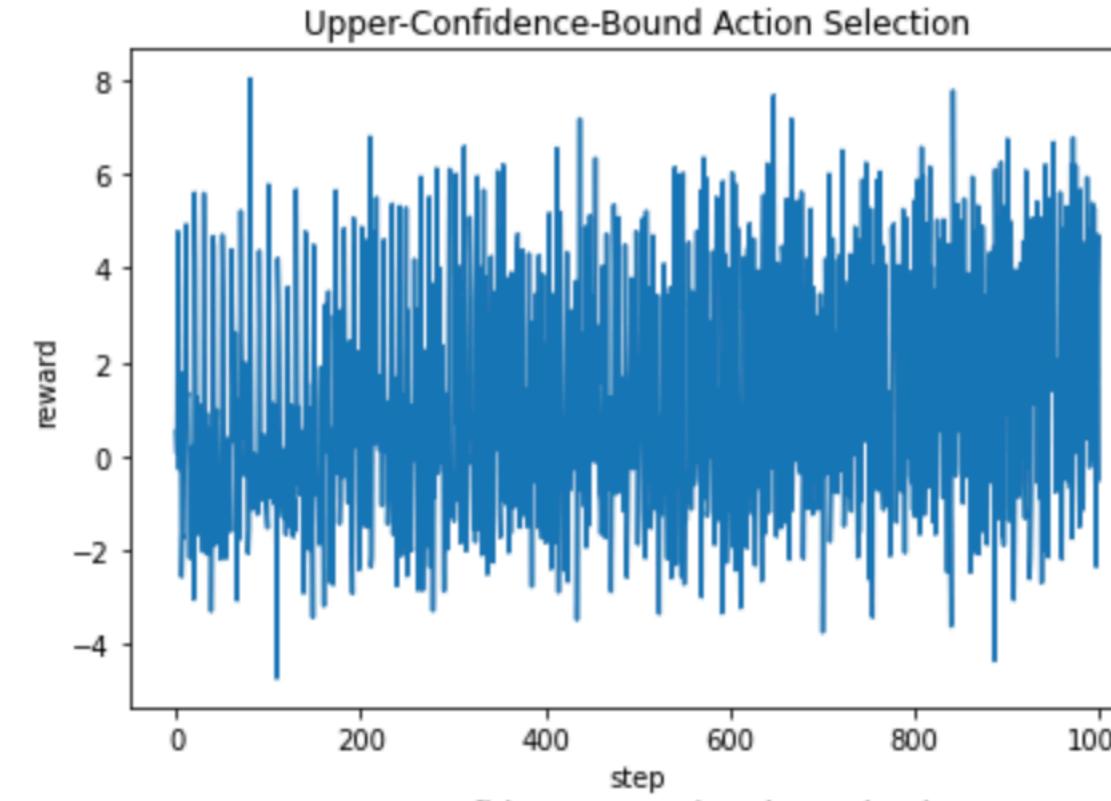
How will your choice of c affect these plots? (Here: $c = 5$)

Performance comparison

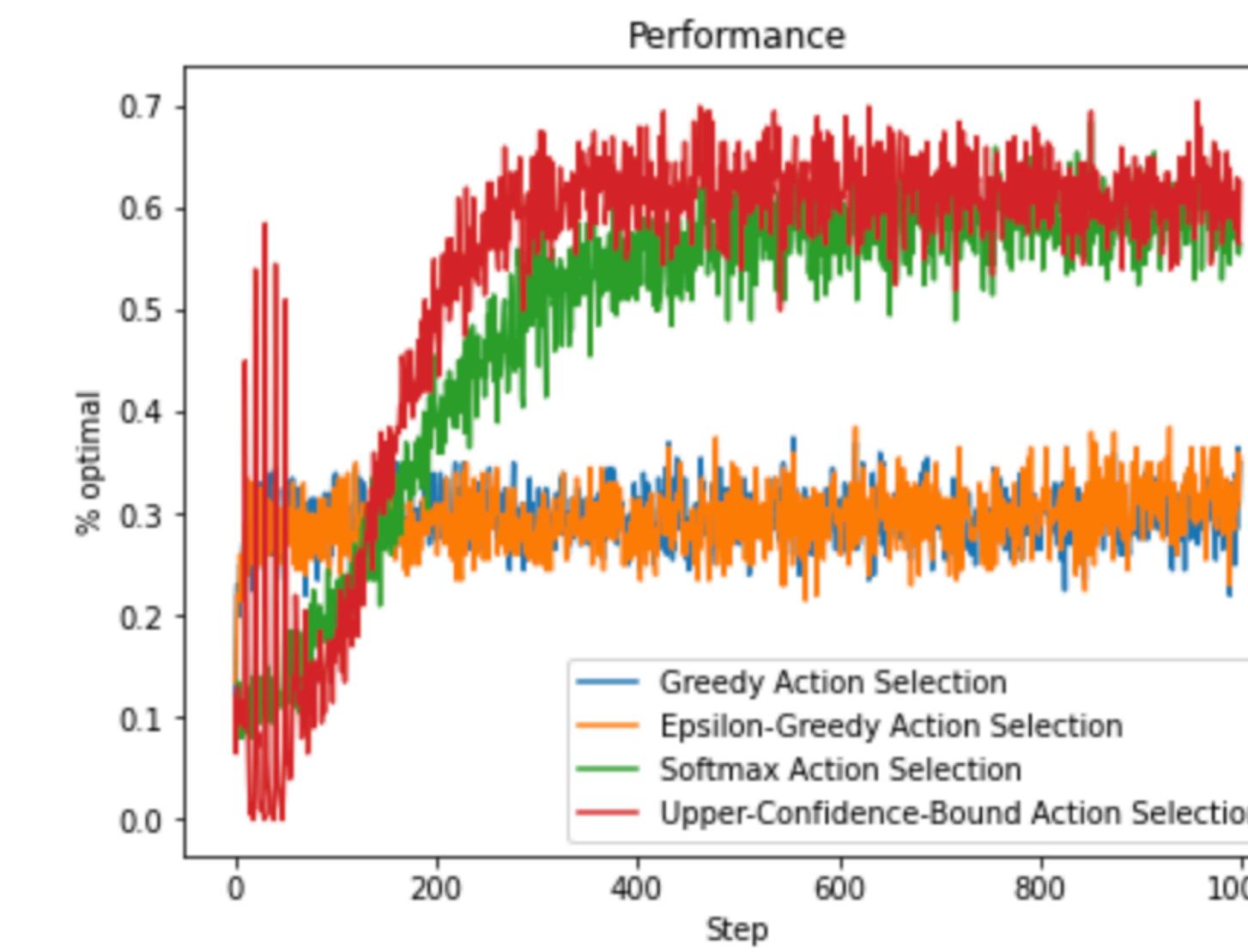
$c = 0$



$c = 50$



Perfomance comparison



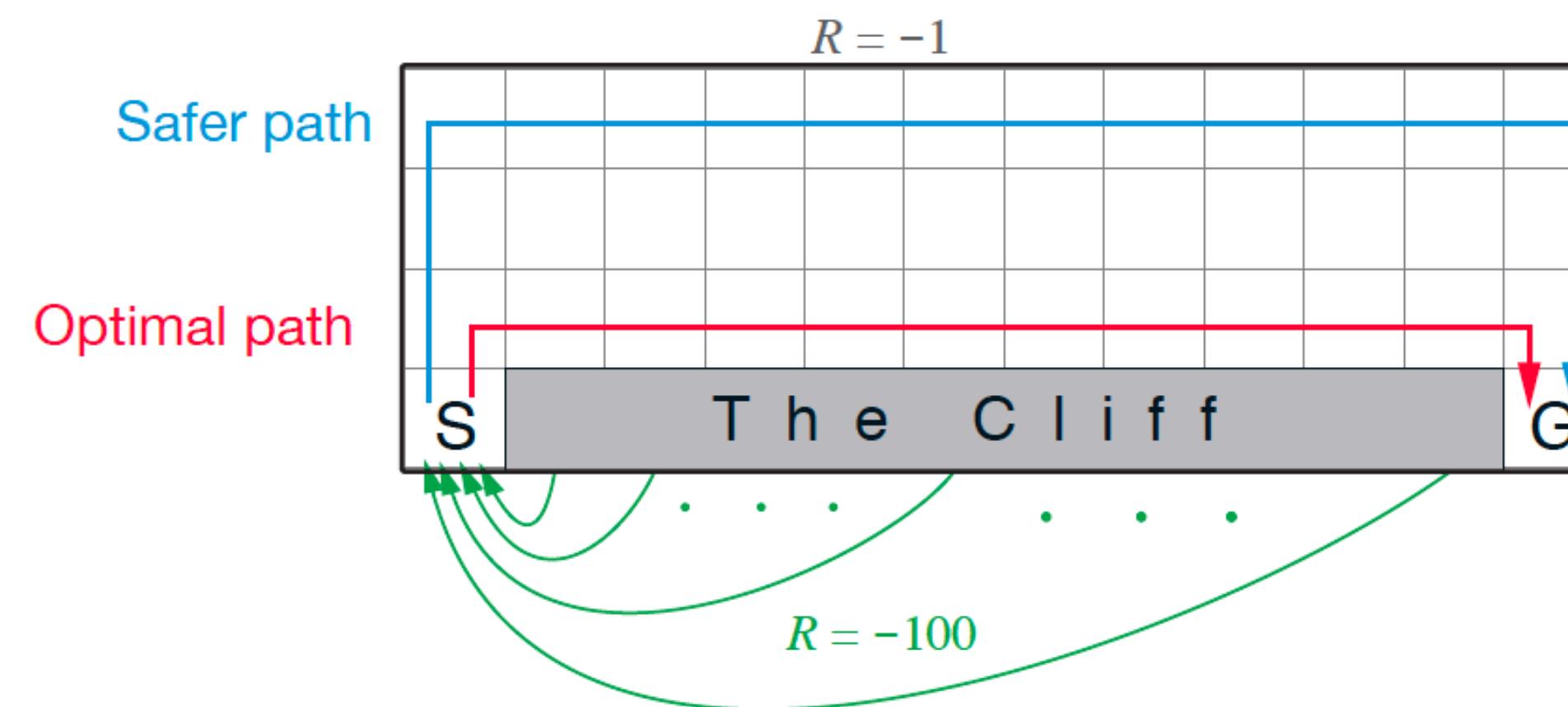
Q-Learning

Limitation of multi-armed bandit problems

Your current action does not influence what happens next!!

How can we solve sequential problems?

The textbook problem:
'Cliff-World'



The rules:

- Agent has to move from start (S) to goal (G)
- Reaching the goal results in a positive reward of +10
- Falling off the cliff results in a negative reward of -100
- Any other state results in a negative reward of -1

What's the problem the agent has to solve here??

Note the subtle introduction of the concept of '**transition probabilities**' here
- implicit, later: explicit

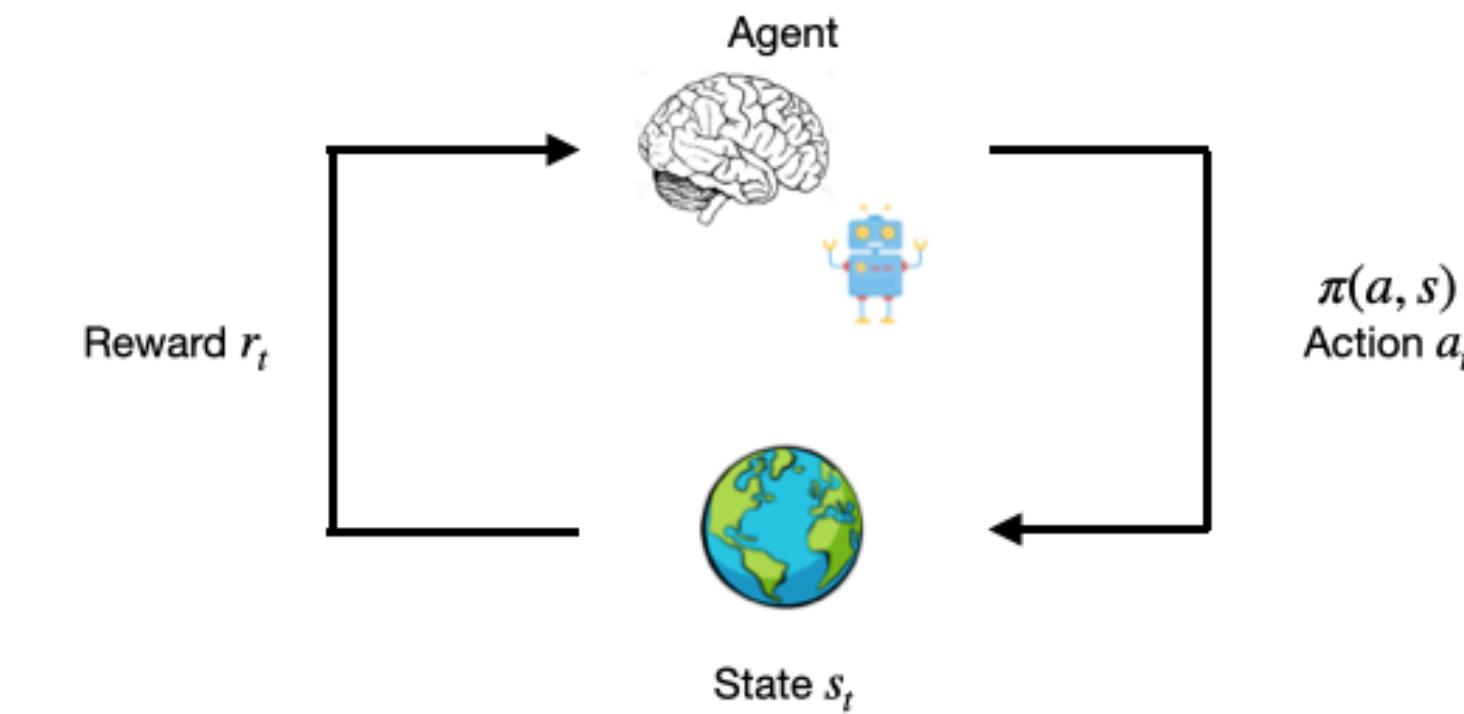
From classical to instrumental learning

TD Learning:

$$V(s_t) \leftarrow V(s_t) + \alpha \cdot (r + \gamma \cdot V(s_{t+1}) - V(s_t))$$

↑ ↑
Learning rate Discount rate

Prediction error



Q-Learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r + \gamma \cdot \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

↑ ↑
Learning rate Discount rate

Prediction error

What's the difference between $V(s_t)$ and $Q(s_t, a_t)$?

What's $\max_a Q(s_t, a_t)$ doing?

Note that this is just an update rule - doesn't tell us how to select an action!

Coding: Q-Learning

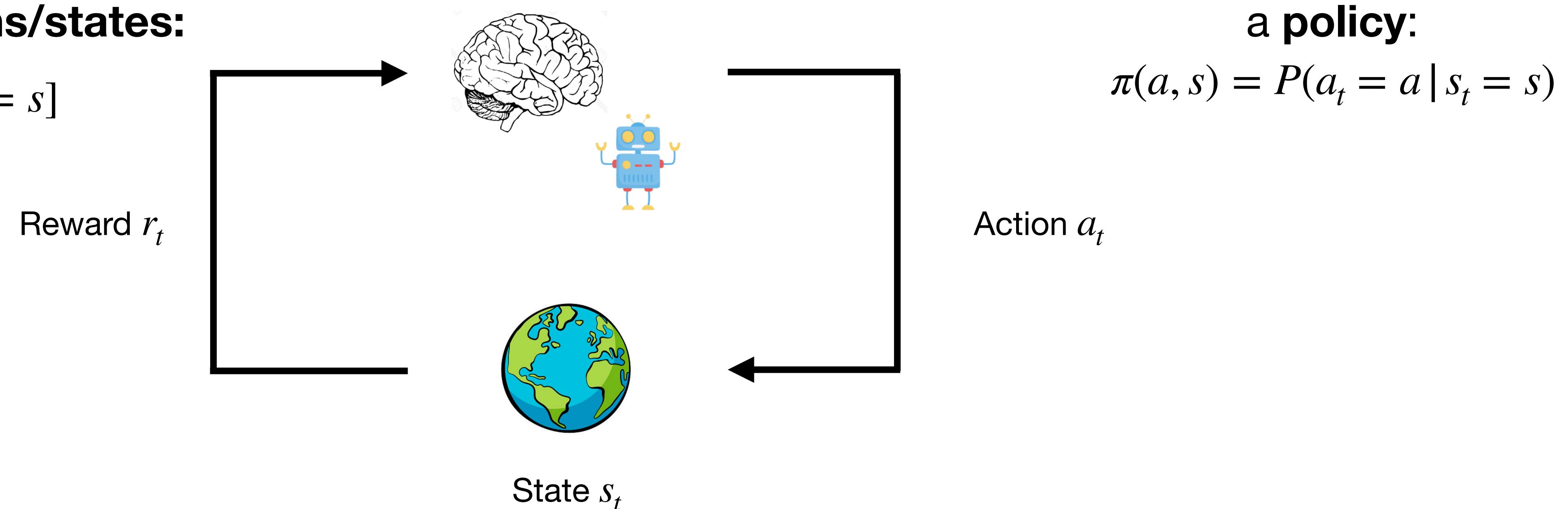
https://github.com/schwartenbeckph/RL-Course/tree/main/2022_06_28

MDPs

Basic setup: how to agents learn to act?

Based on a reward signal, agents learn **values of actions/states**:

$$V_{\pi}(s) = \mathbb{E}_{\pi}[R | s_0 = s]$$

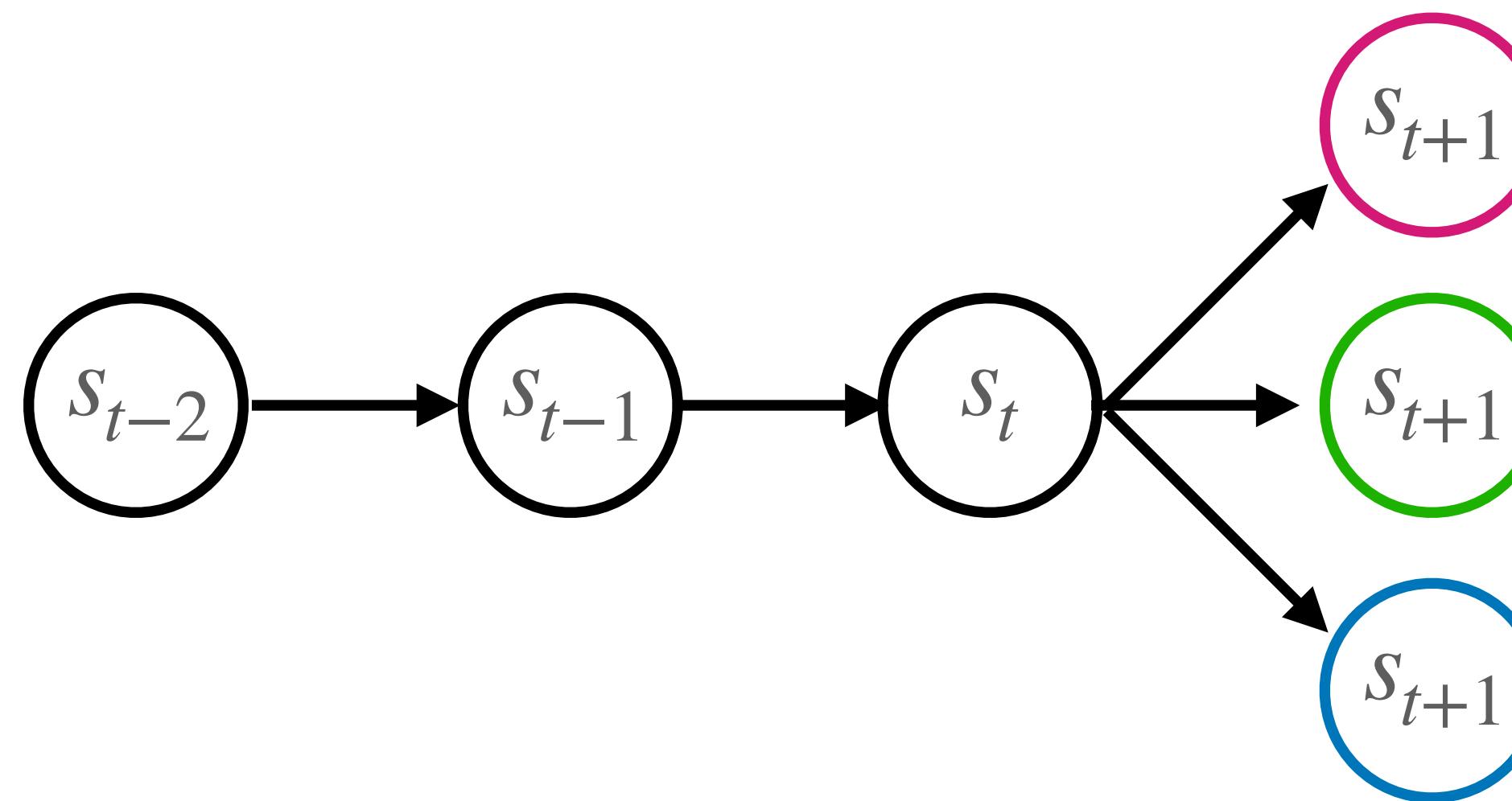


Action is governed by a **policy**:

$$\pi(a, s) = P(a_t = a | s_t = s)$$

Markov Process

Most RL problems are problems where agents face sequences of states:



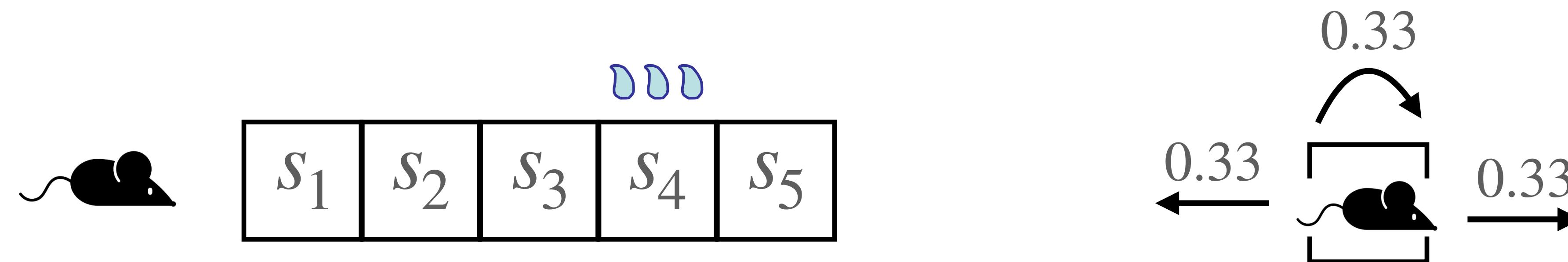
Fundamental property: **Markov property**

$$P(s_{t+1} = s | s_t, s_{t-1}, s_{t-2}, \dots) = P(s_{t+1} = s | s_t)$$

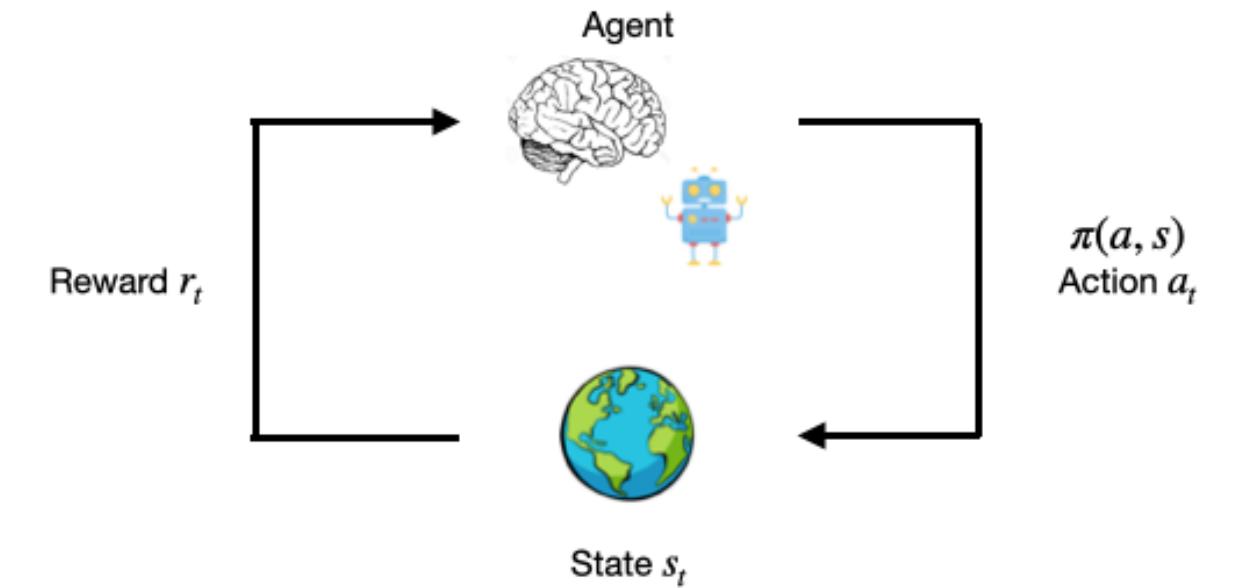
“The future is independent of the past given the present”

Markov Process

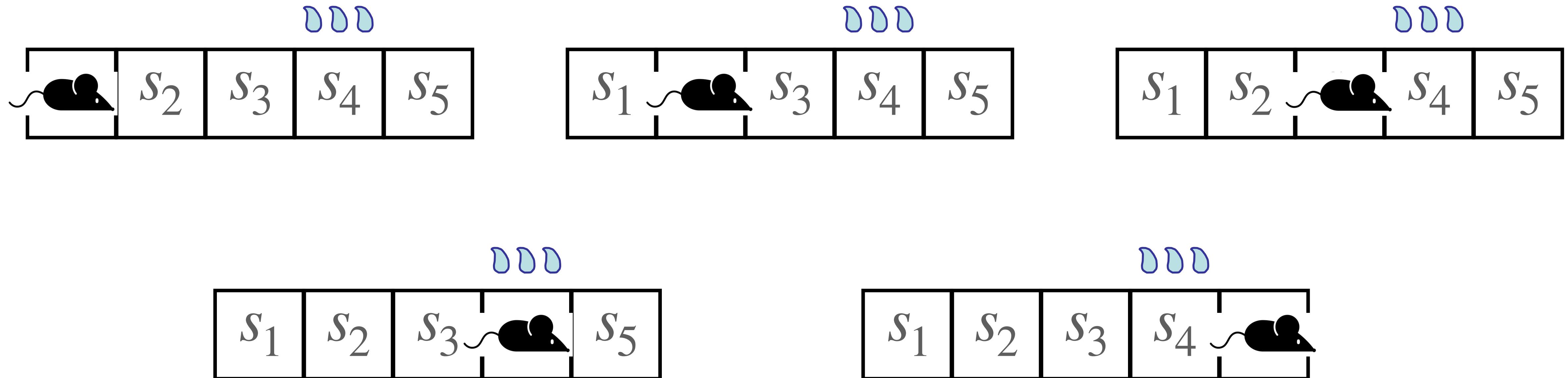
Let's assume a super simple problem:



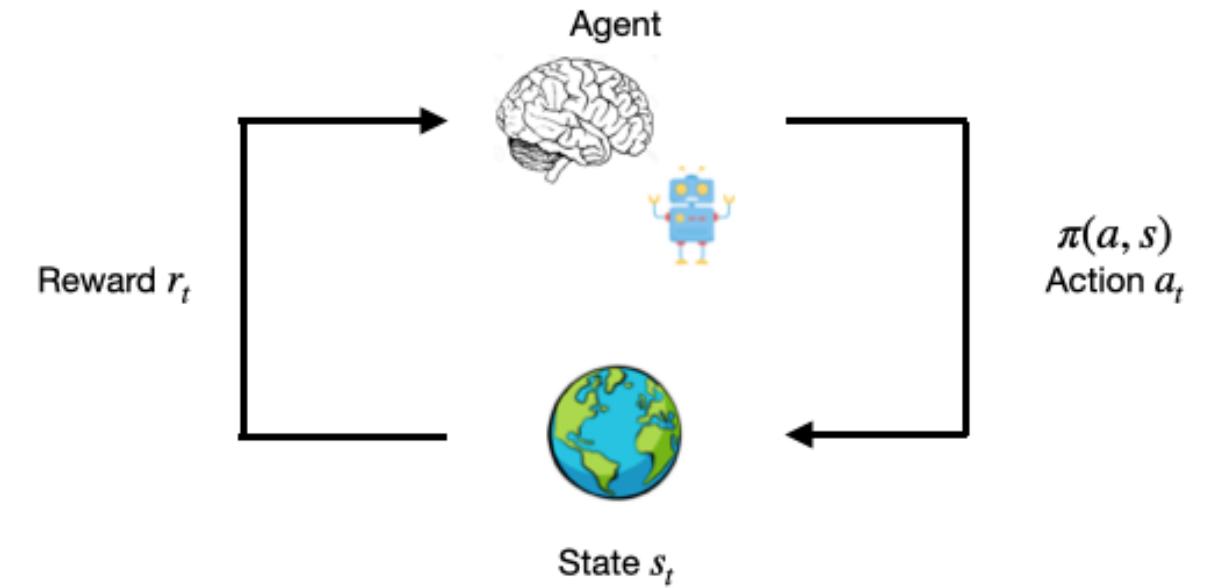
Markov Process



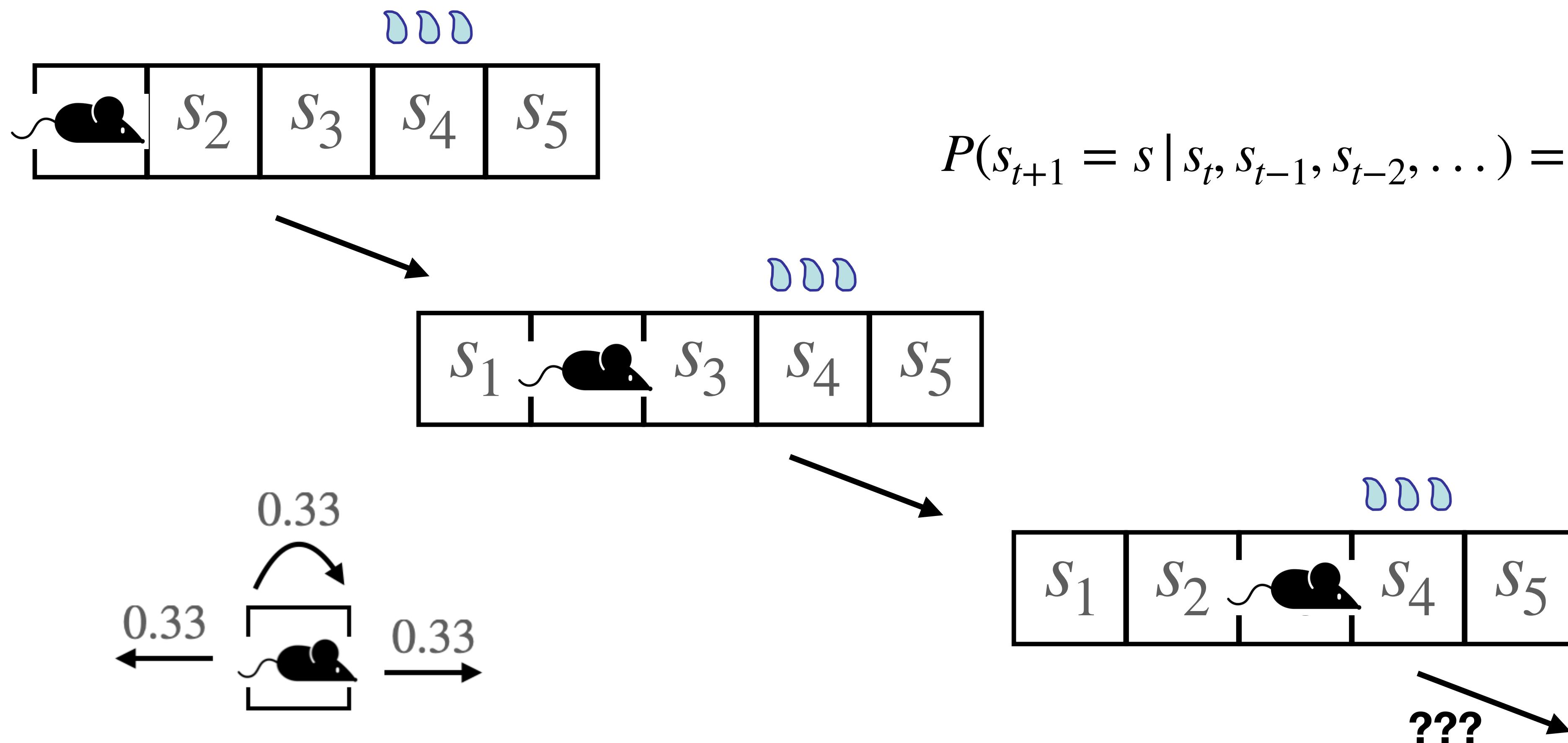
We can now define a **state space** S :



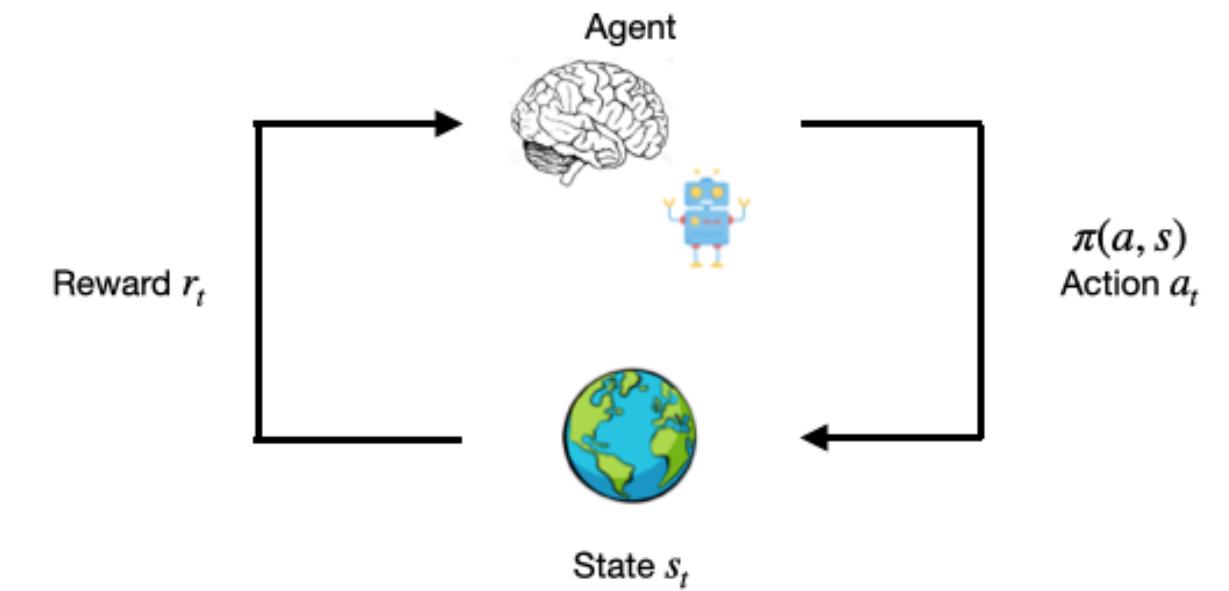
Markov Process



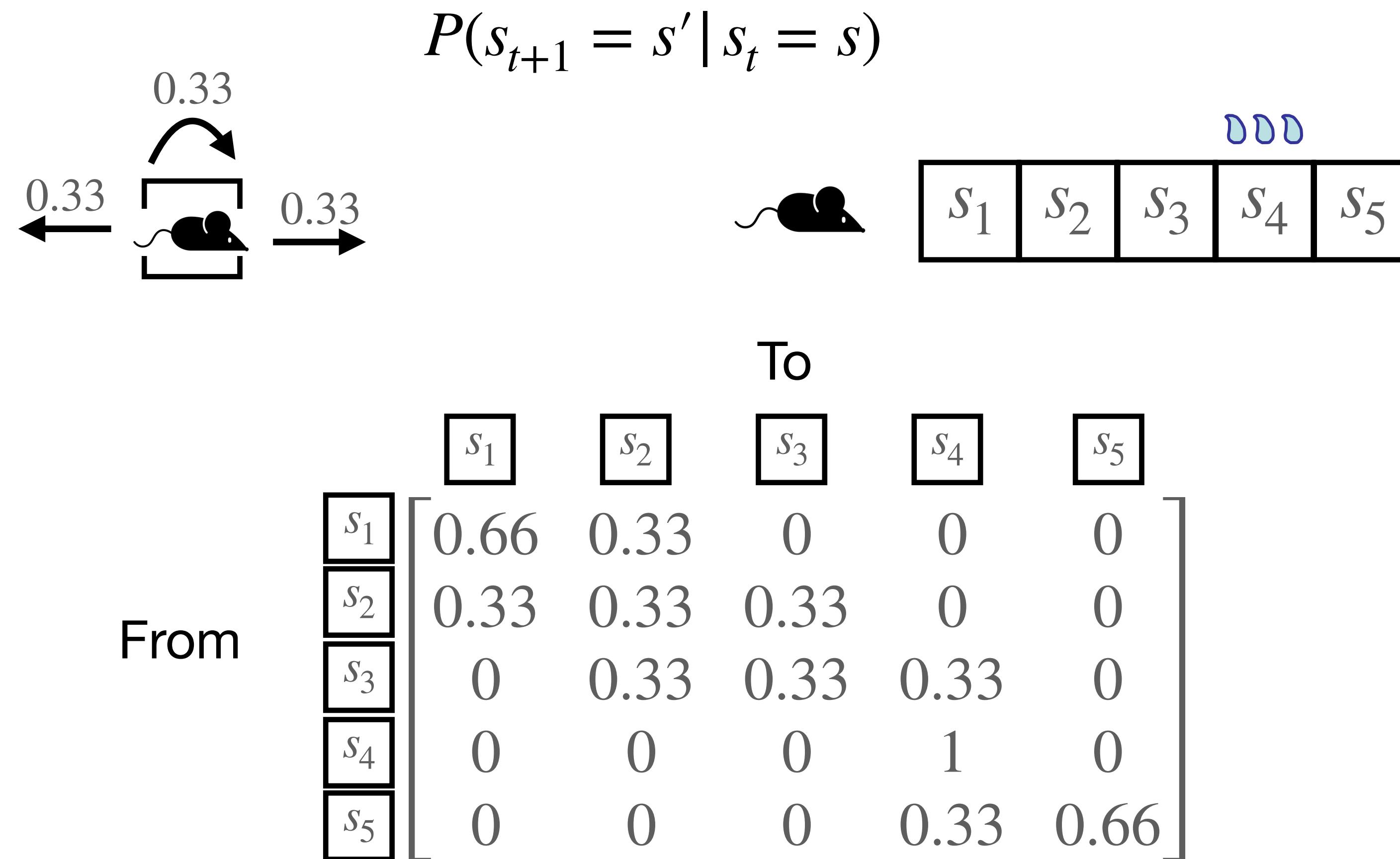
Does this problem have the **Markov Property**?



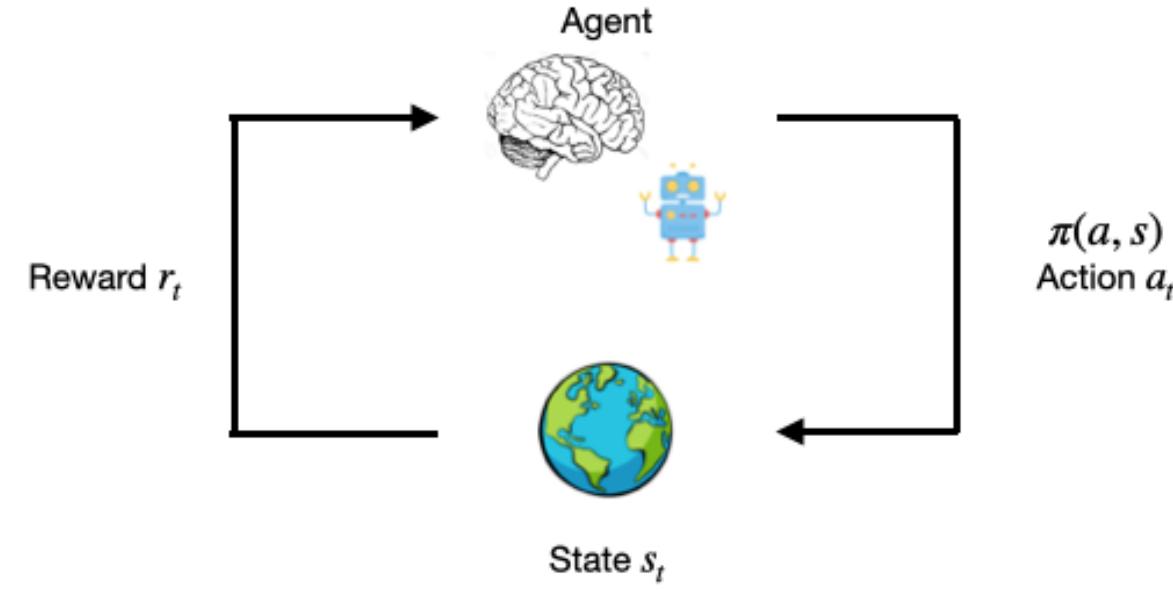
Markov Process



This allows us to define **transition probabilities** P :



Markov Reward Process

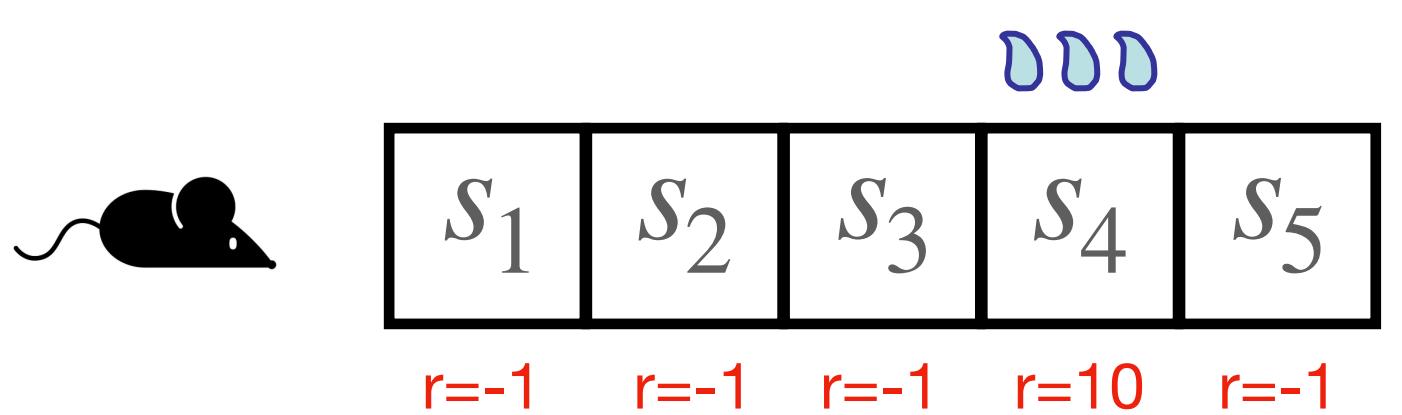


A **Markov Process** is defined based on

- A **State Space** S
- **Transition Probabilities** P
$$P(s_{t+1} = s' | s_t = s)$$

A **Markov Reward Process** is defined based on

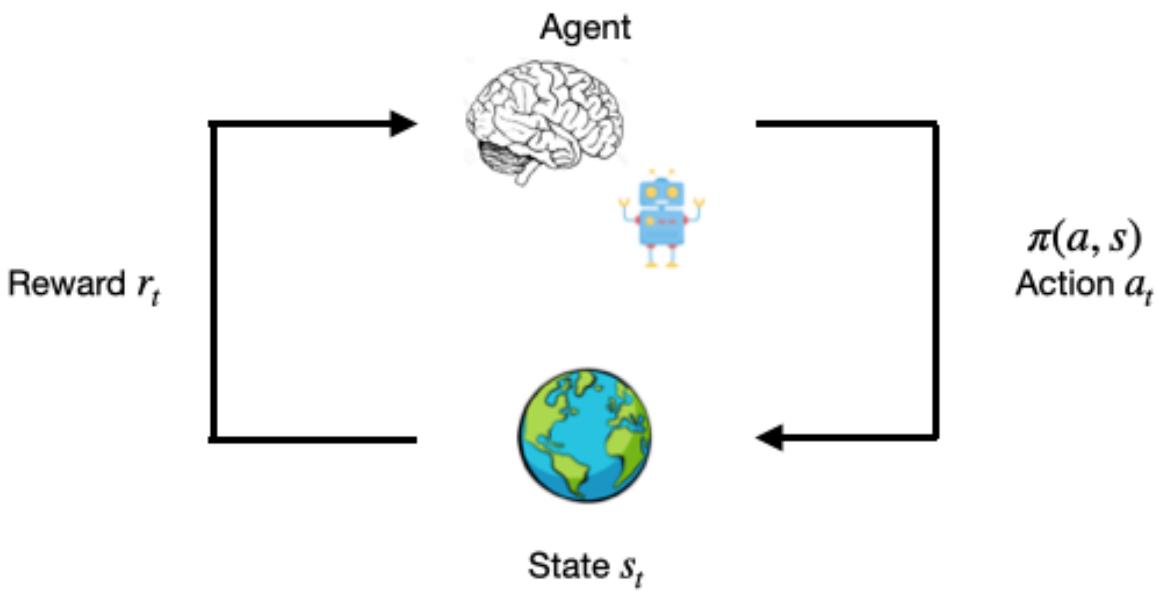
- A State Space S
- Transition Probabilities P
- A **Reward Function** $R_s = \mathbb{E}[r_{t+1} | s_t = s]$
- A **Discount Factor** $\gamma \in [0, 1]$



Allows to define **Return**

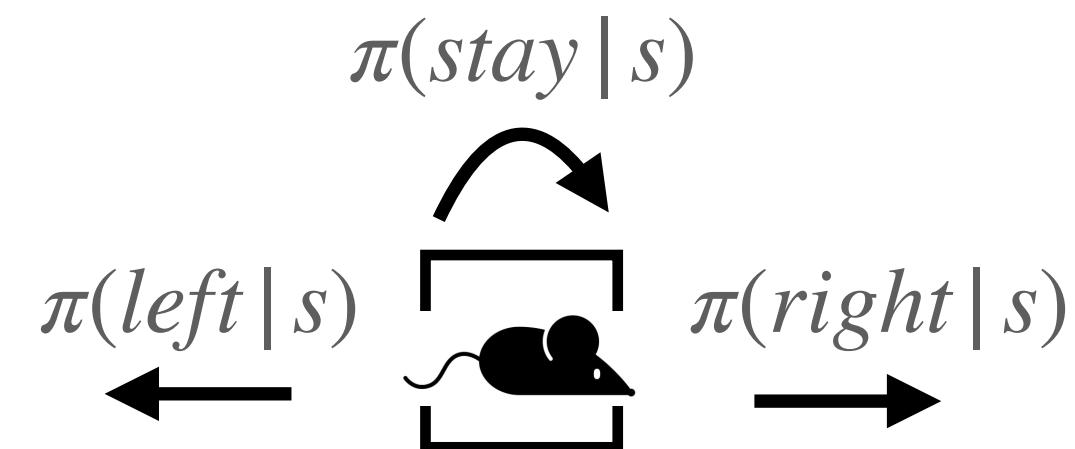
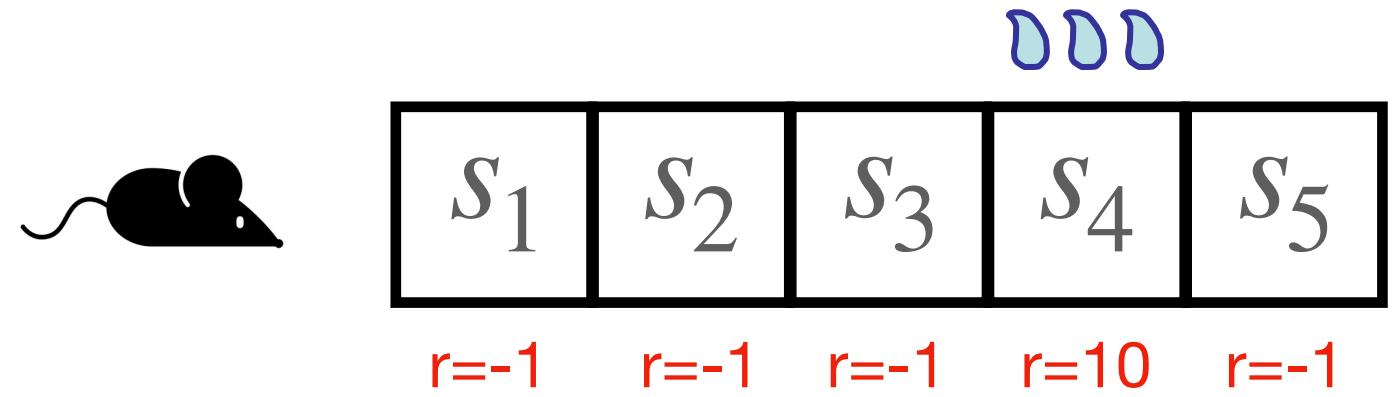
$$G_t = R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1}$$

Markov Decision Process



A **Markov Decision Process** is defined based on

- A State Space S
- An **Action Space** A
- Transition Probabilities P
- A Reward Function $R_s = \mathbb{E}[r_{t+1} | s_t = s]$
- A Discount Factor $\gamma \in [0,1]$

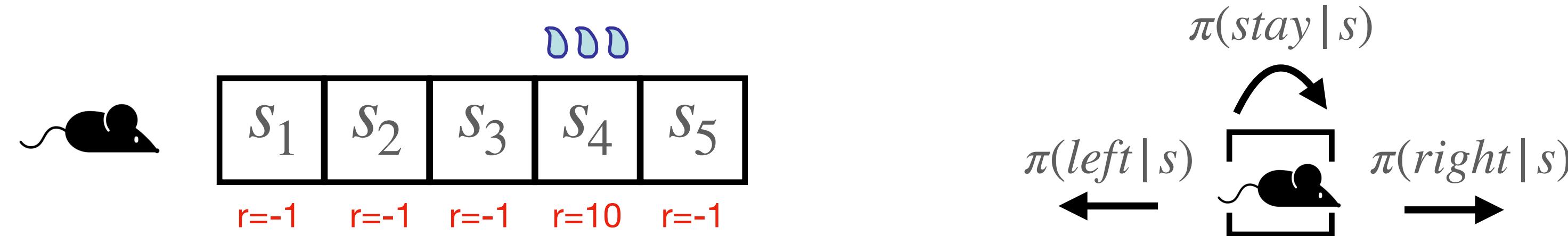


Actions are governed via a **policy**: $\pi(a, s) = P(a_t = a | s_t = s)$

MDPs basis for model-based RL

Allows to specify all environment dynamics for RL problem:

$$P(s', r | s, a) = P(s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a)$$

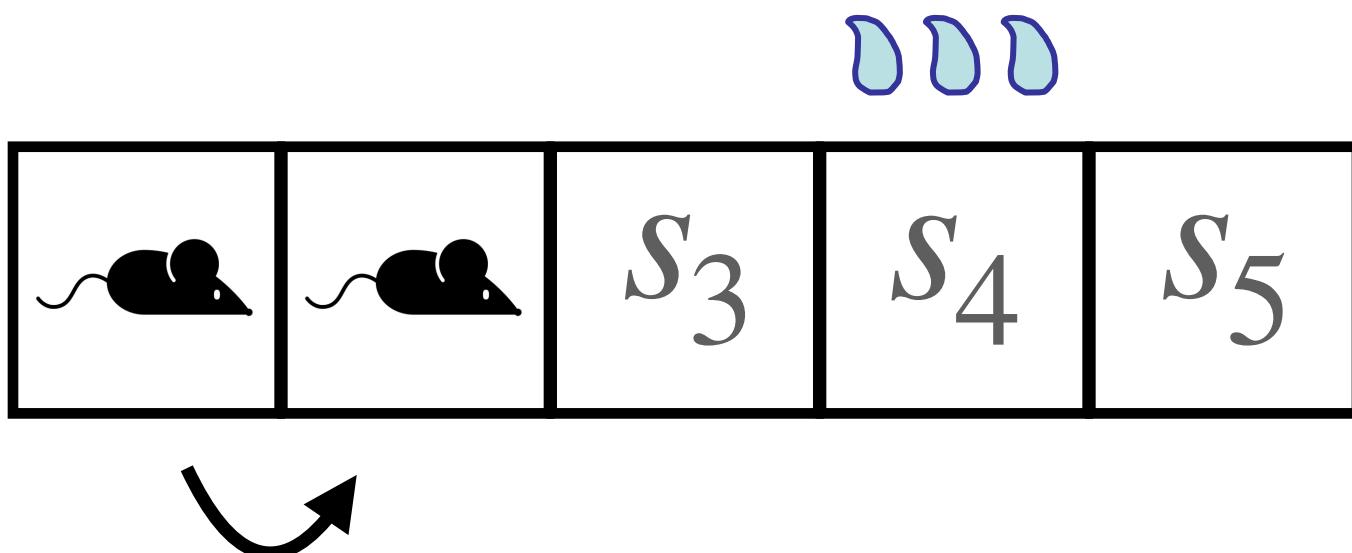


MDPs basis for model-based RL

$$P(s', r | s, a) = P(s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a)$$

Allows to specify useful thinks like **state-transition probabilities** (often T):

$$P(s' | s, a) = P(s_{t+1} = s' | s_t = s, a_t = a) = \sum_r P(s', r | s, a)$$



$$P(s' | s, right) = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$