Fall 2022 | Lecture 16

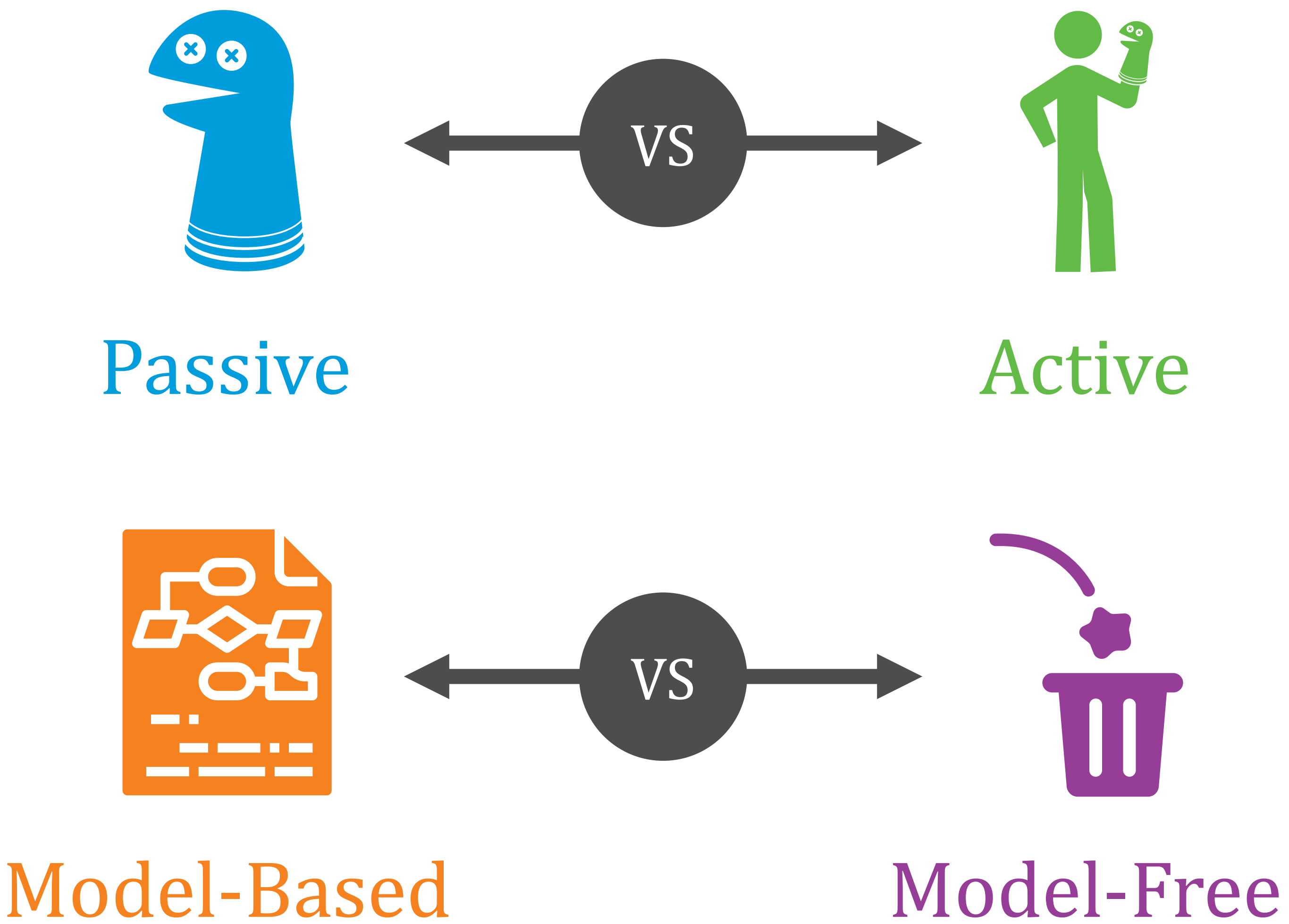
Reinforcement Learning

Ariel Procaccia | Harvard University

REINFORCEMENT LEARNING

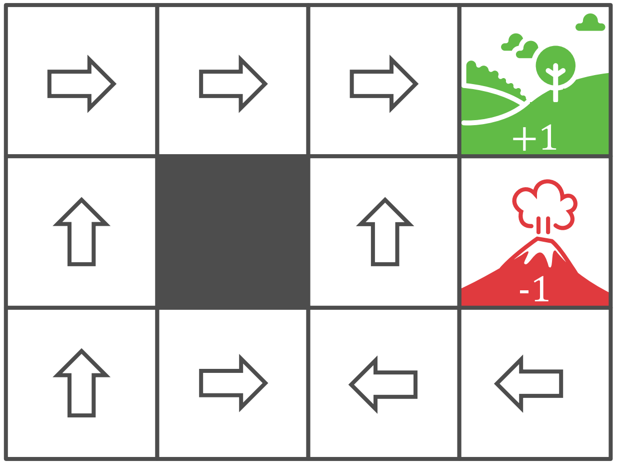
- Reinforcement refers to getting feedback through rewards
- Markov decision processes also have rewards, but the environment is known
- Today we focus on learning to act in an unknown environment, which is represented as an MDP with unknown transitions and rewards

RL DIMENSIONS

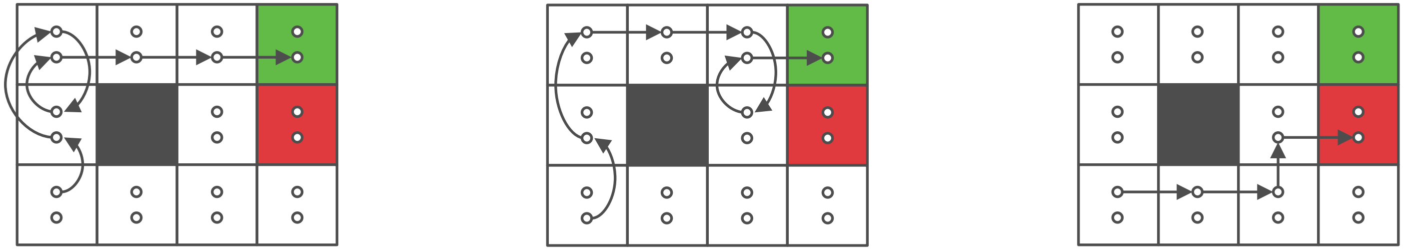


PASSIVE RL

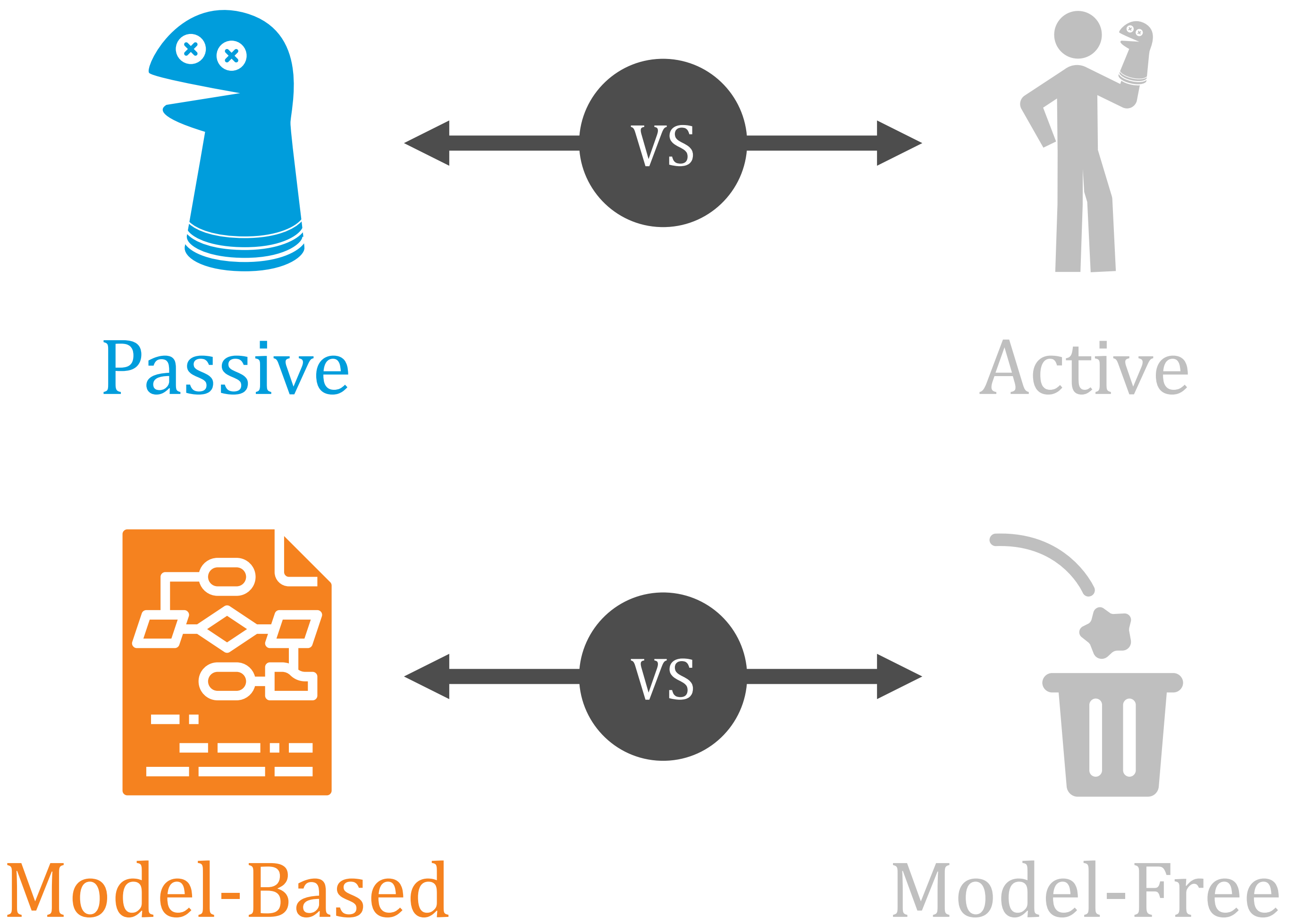
Fixed policy π



Sampled trajectories



RL DIMENSIONS



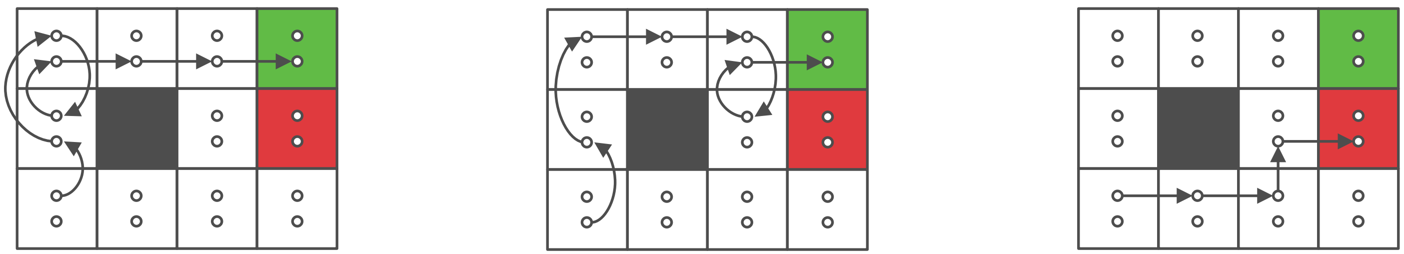
MONTE CARLO EVALUATION

- Goal: learn an estimate $\hat{U}(s)$ of the utility $U^{\pi}(s)$ of each state s for the fixed policy π
- We observe the reward $R(s)$ when we visit state s
- We estimate $\hat{P}(s'|\, s, \pi(s))$ by observing how many times s' was reached when taking action $\pi(s)$ in s and normalizing
- We can now compute $\hat{U}$ by solving the following equalities for all $s \in S$:

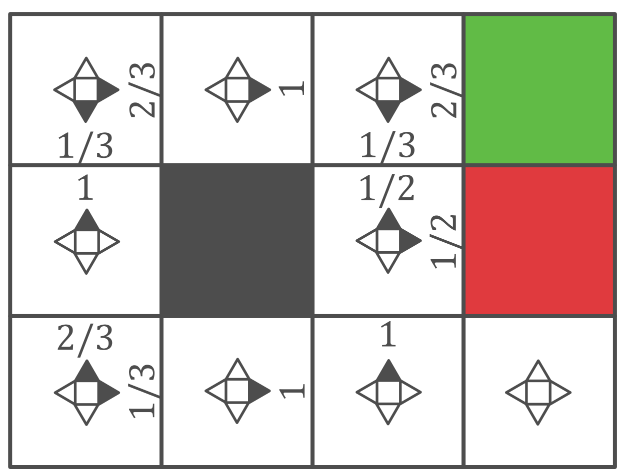
$$\hat{U}(s) = R(s) + \gamma \sum_{s' \in S} \hat{P}(s' \mid s, \pi(s)) \cdot \hat{U}(s')$$

MONTE CARLO: EXAMPLE

Sampled trajectories

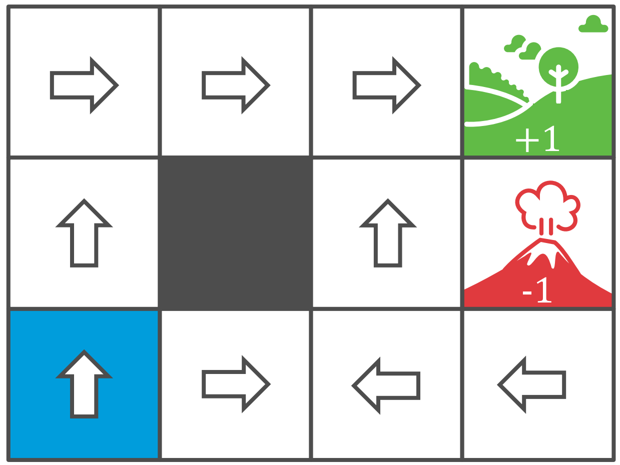


Estimated $\hat{P}$



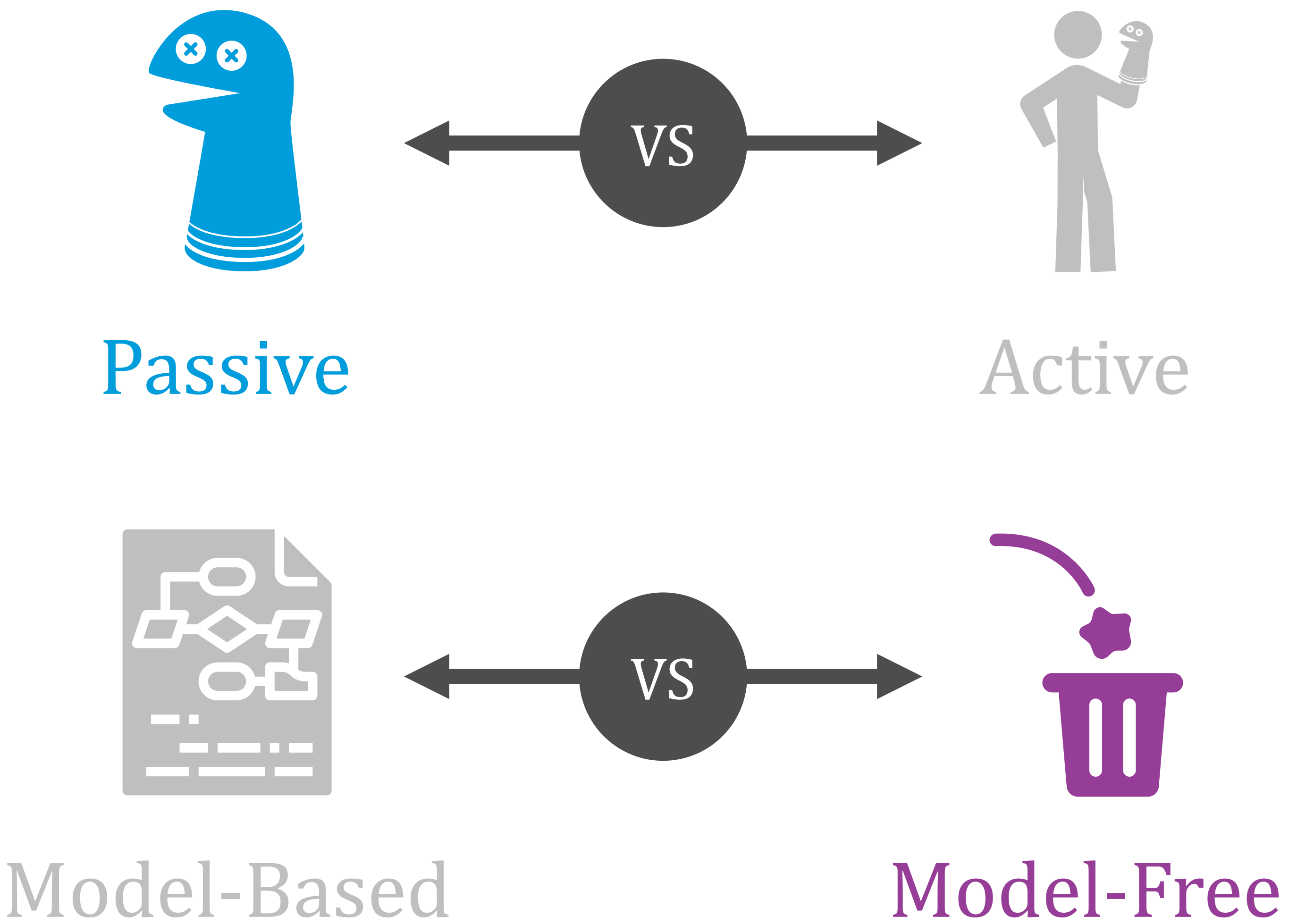
MONTE CARLO: EXAMPLE

Fixed policy π



Poll 1: Suppose trajectories starting at (1,1) are sampled. Is it the case that for every state (i, j), $\widehat{U}(i, j)$ converges to $U^{\pi}(i, j)$?

RL DIMENSIONS



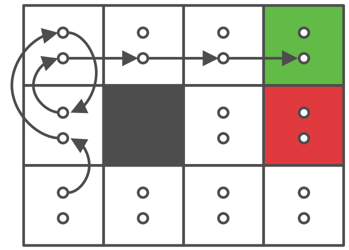
TEMPORAL-DIFFERENCE LEARNING

- Each time a transition from s to s' is encountered, update $\hat{U}(s)$ via

$$\hat{U}(s) \leftarrow (1-\alpha)\hat{U}(s) + \alpha\left(R(s) + \gamma\hat{U}(s')\right)$$

- The learning rate $\alpha = \alpha(k_s)$ depends on the number of times state s was visited, k_s
- If α decreases appropriately as k_s increases then $\hat{U}$ will converge to U^{π}

TD LEARNING: EXAMPLE



$$\hat{U}(s) \leftarrow (1-\alpha)\hat{U}(s) + \alpha\left(R(s) + \gamma\hat{U}(s')\right)$$

Estimated $\hat{U}$ for $\alpha = 0.5, \gamma = 1, R(s) = -0.1$

0	0	0	0
0		0	0
0	0	0	0

0	0	0	0
0		0	0
-0.05	0	0	0

0	0	0	0
-0.05		0	0
-0.05	0	0	0

-0.07	0	0	0
-0.05		0	0
-0.05	0	0	0

-0.07	0	0	0
-0.11		0	0
-0.05	0	0	0

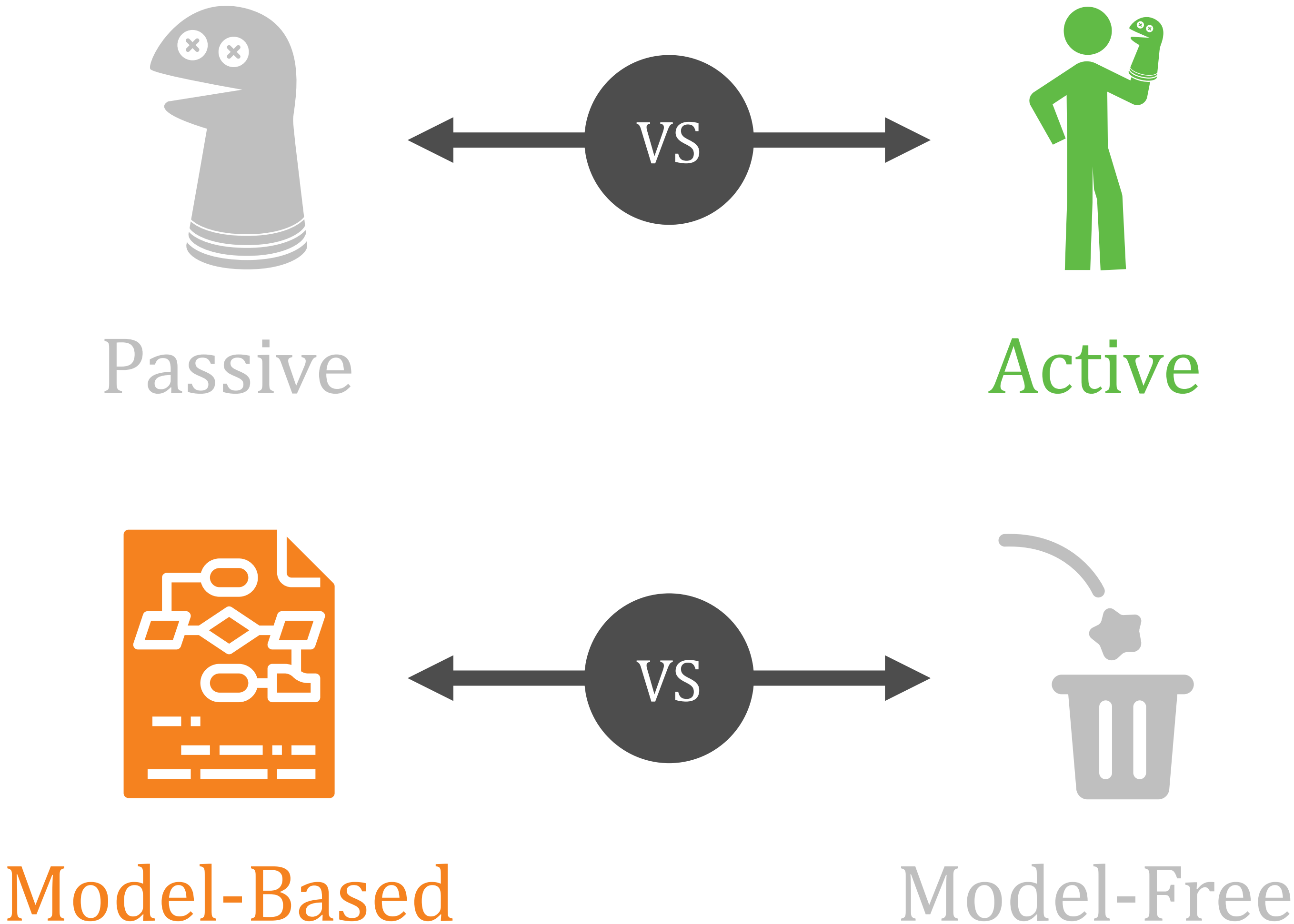
-0.08	0	0	0
-0.11		0	0
-0.05	0	0	0

-0.08	-0.05	0	0
-0.11		0	0
-0.05	0	0	0

-0.08	-0.05	-0.05	0
-0.11		0	0
-0.05	0	0	0

-0.08	-0.05	-0.05	0.5
-0.11		0	0
-0.05	0	0	0

RL DIMENSIONS



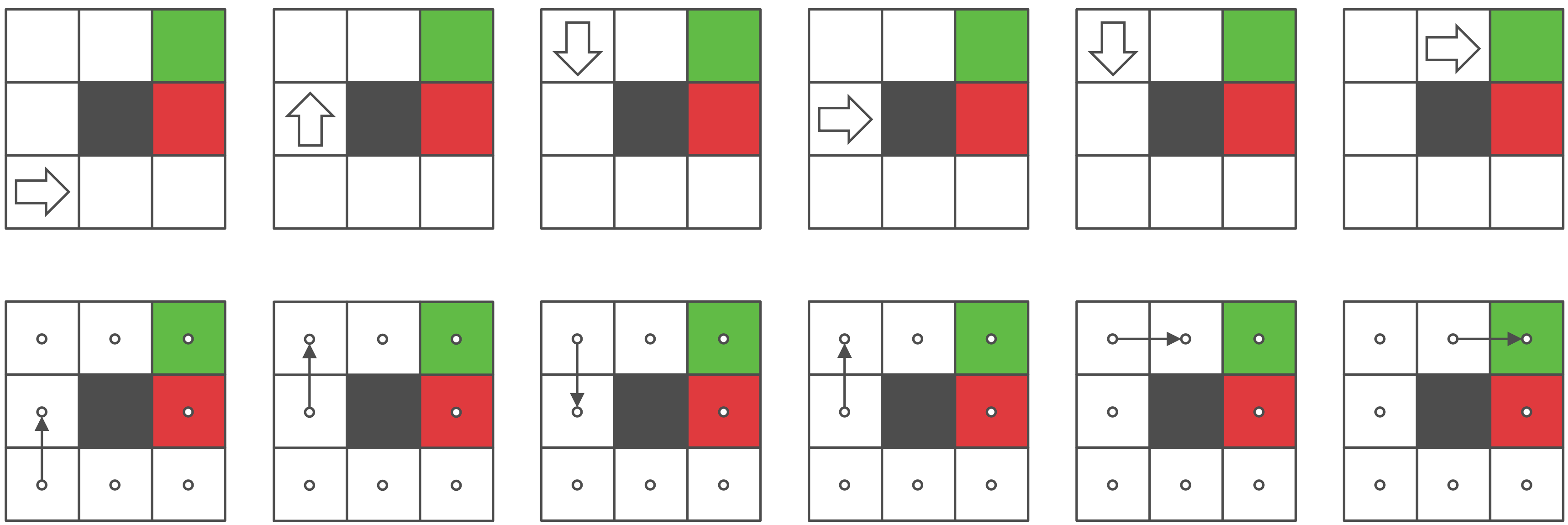
MONTE CARLO REDUX

- Our goal is to learn an approximately optimal policy
- Instead of following a fixed policy π, we take a random action at each step (for now)
- We estimate $\hat{P}(s' \mid s, a)$ by observing how many times s' was reached when taking action a in s and normalizing
- We can now solve the Bellman equations for all $s \in S$ and derive an optimal policy:

$$\hat{U}(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s' \in S} \hat{P}(s' \mid s, a) \cdot \hat{U}(s')$$

LEARNING THE MODEL: EXAMPLE

Random actions and transitions

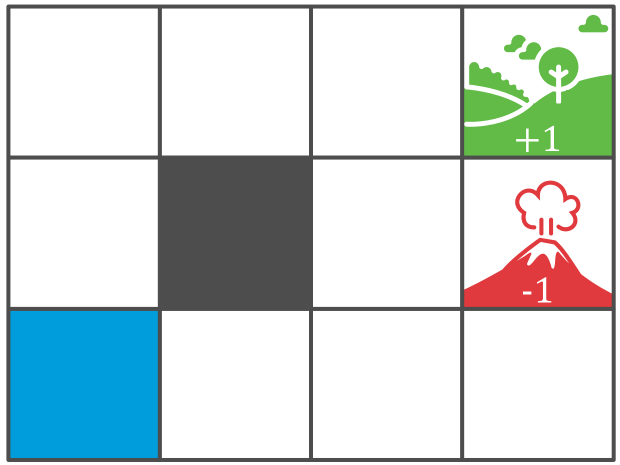


Transition table for (1,3):

	(1,1)	(1,2)	(2,1)	(2,3)	(3,1)	(3,2)	(3,3)
⇧	—	—	—	—	—	—	—
⇨	—	—	—	—	—	—	—
⇩	0	1	0	1	0	0	0
⇦	—	—	—	—	—	—	—

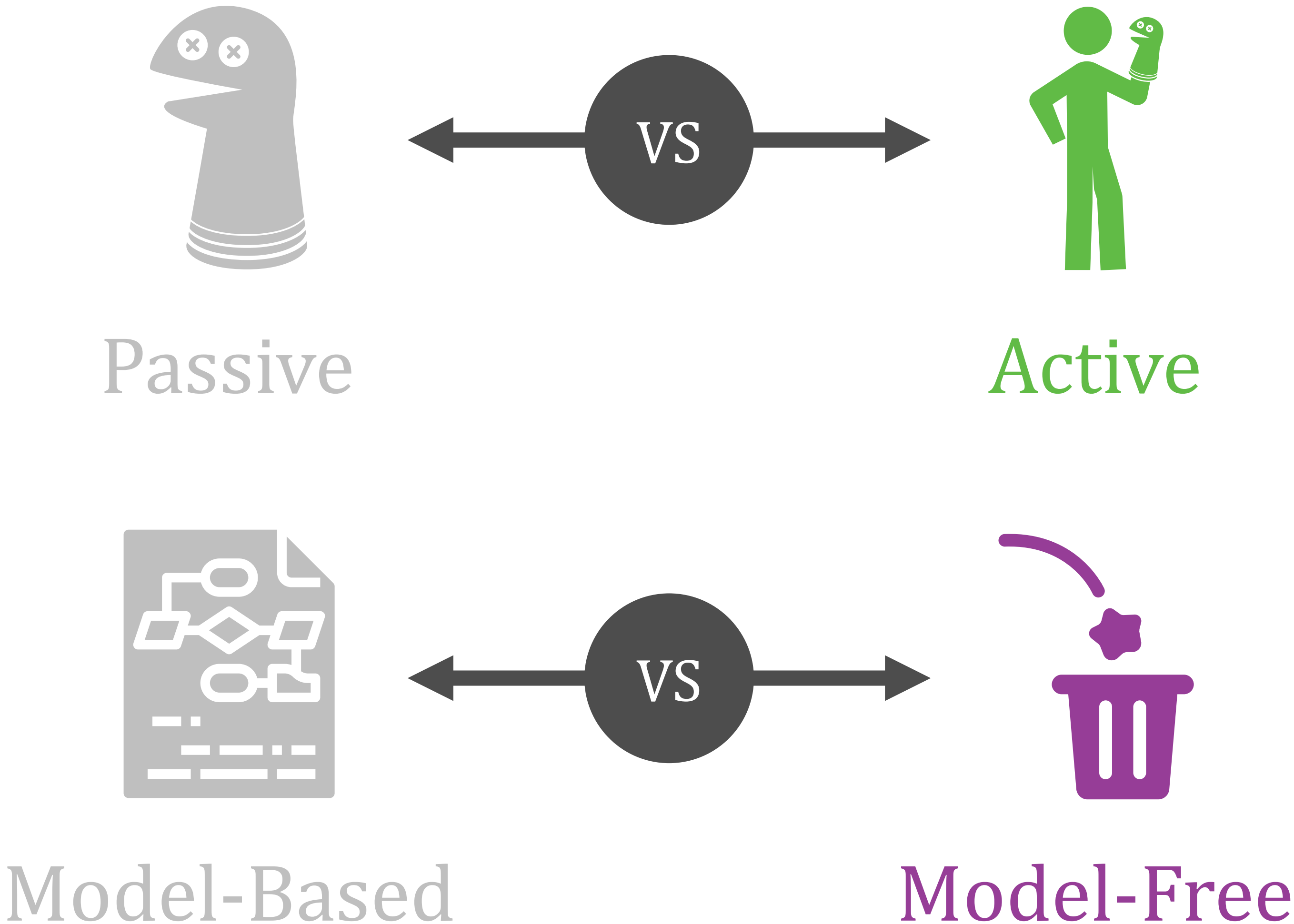
MONTE CARLO: EXAMPLE

Random exploration



Poll 2: Suppose trajectories starting at (1,1) are sampled. Is it the case that for every state (i, j), $\widehat{U}(i, j)$ converges to $U(i, j)$?

RL DIMENSIONS



TD-LEARNING REDUX

- Model-based passive RL extends to the active case; does the model-free TD algorithm extend too?
- TD learning simply estimates the utilities $\hat{U}$
- This doesn’t extend to a model-free active RL algorithm because the learned parameters don’t contain enough information to choose actions

Q-LEARNING

- Idea: If we had an estimate $Q(s, a)$ for each state-action pair then we could choose, for all $s \in S$,

$$\pi(s) \in \operatorname*{argmax}_{a \in A_s} Q(s, a)$$

- In equilibrium the optimal Q values satisfy Bellman-like equations for all $s \in S, a \in A_s$:

$$Q(s, a) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) \max_{a' \in A_{s'}} Q(s', a')$$

Q-LEARNING

- Instead of following a fixed policy π, we take a random action at each step (for now)
- Each time a transition from s to s' via action a is encountered, update $Q(s,a)$ via
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s',a')\right)$$
- As before, $\alpha = \alpha(k_{sa})$ is the learning rate which now depends on the number of times action a was taken in state s

EXPLORATION VS. EXPLOITATION

Exploitation

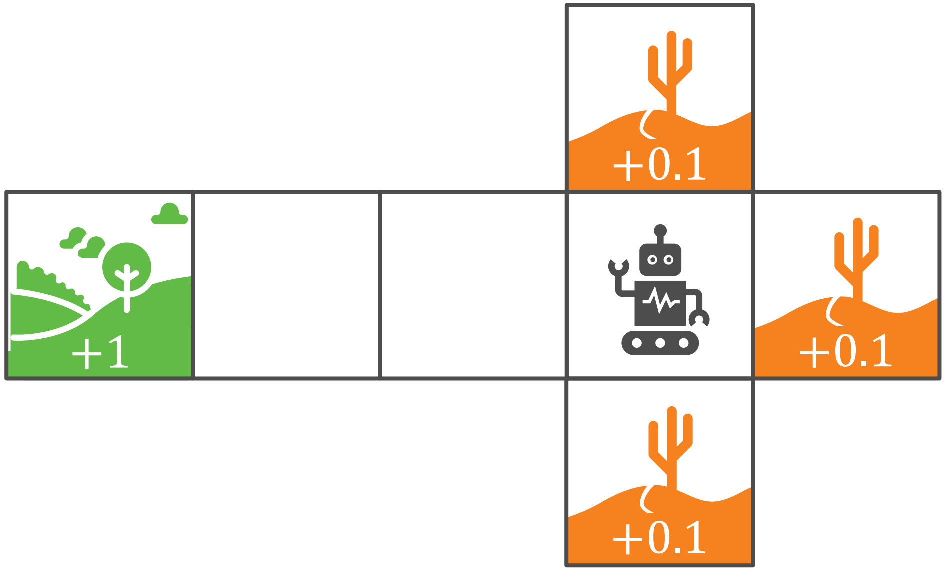
Obtain reward from a familiar option

Exploration

Risk reward to seek a better option

THE COST OF GREED

What would happen if a Q-learning agent always selected the action $\arg\max_a Q(s, a)$?

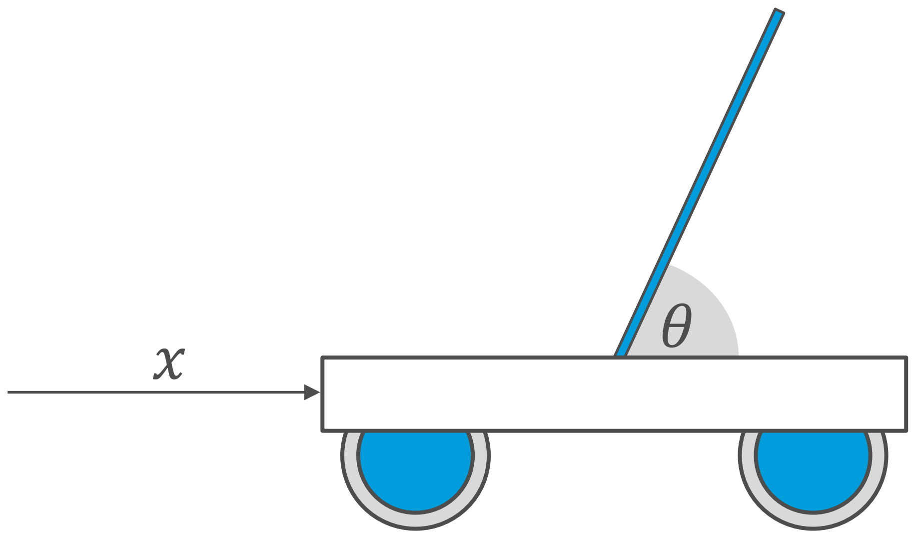


HOW TO EXPLORE

- **Theorem:** If all state action pairs are visited infinitely often, and $\alpha(k_{sa})$ goes to 0 at an appropriate rate, then Q-learning converges to an optimal policy
- To satisfy the exploration requirement:
  - **ϵ-exploration:** Use $\operatorname{argmax}_a Q(s,a)$ with probability $1-\epsilon$ and a random action with probability ϵ
  - **Softmax:** Choose each action with probability

$$\frac{e^{Q(s,a)/\theta}}{\sum_{a' \in A_s} e^{Q(s,a')/\theta}}$$

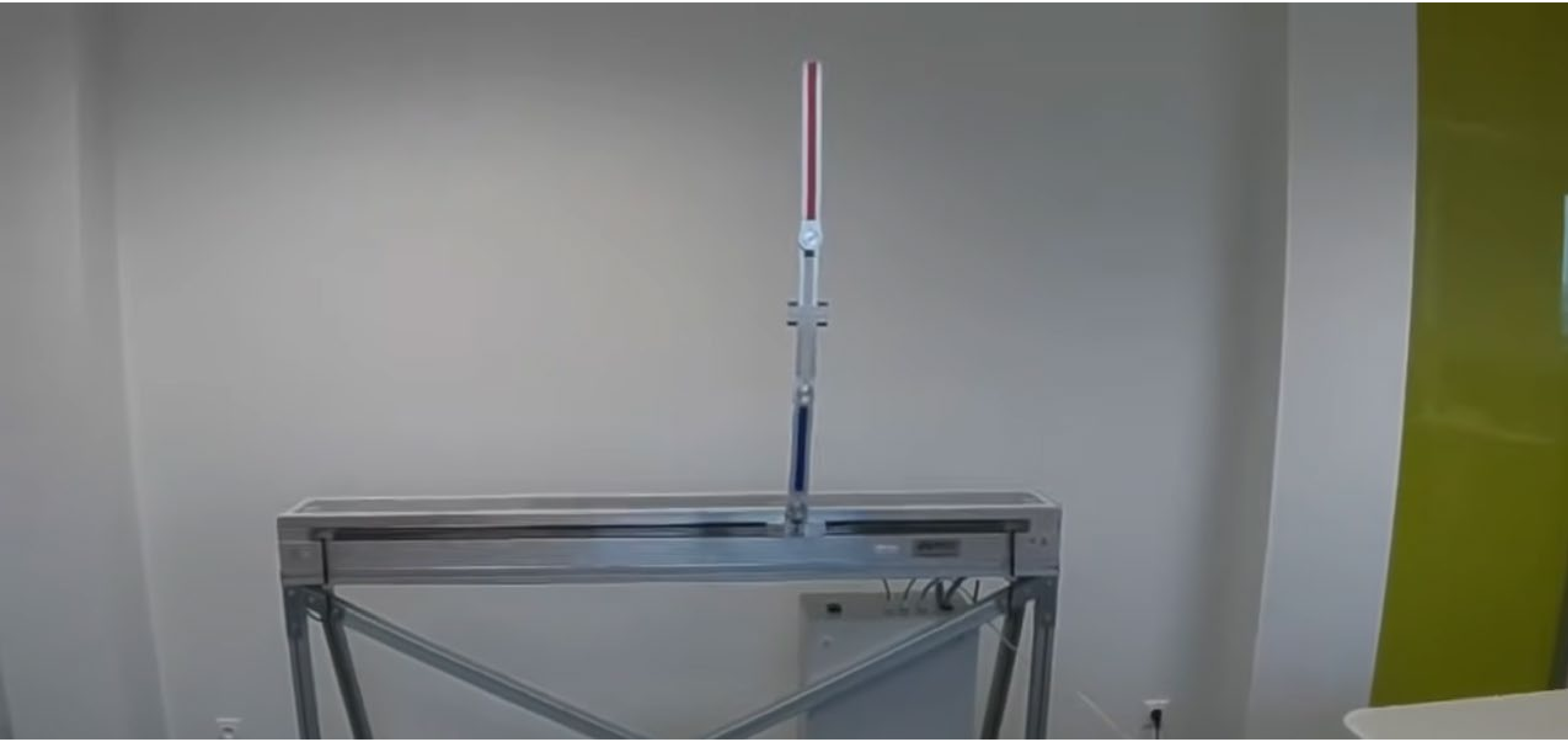
APPLICATION: ROBOT CONTROL



The cart-pole balancing problem

BOXES (1968) is a reinforcement learning algorithm that discretizes the parameter space into boxes and gives negative reward for failure. The action space is “jerk left” or “jerk right.”

APPLICATION: ROBOT CONTROL

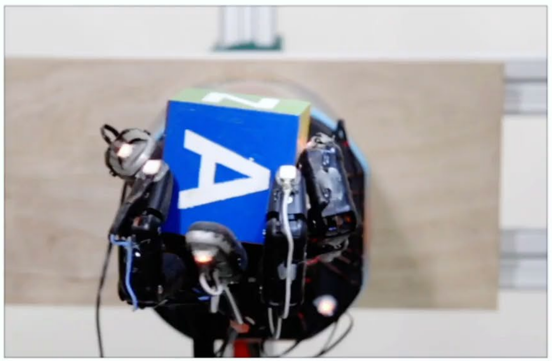


https://www.youtube.com/watch?v=meMWfva-Jio

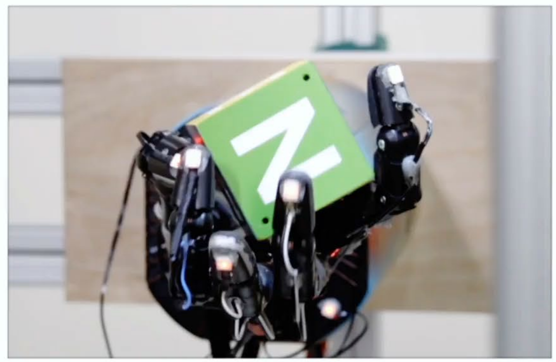
APPLICATION: ROBOT CONTROL

https://www.youtube.com/watch?v=VCdxqn0fcnE

APPLICATION: ROBOT CONTROL



Finger pivoting



Sliding

Finger gaiting

[Open AI et al., 2018]