Reinforcement Learning

Russell and Norvig: Chapter 21 (21.1-21.4)
CSE 240: Winter 2023
Lecture 17 (or 18?)

Announcements

- Assignment 4 is due on Friday at 5pm
- Assignment 5 posted and it is due on Wednesday: 03/22 of finals week
- Last Quiz due *next* Friday: 03/17 at 5pm
  - We will drop the lowest quiz
- Please do the course evaluations
  - We will award 1 point of extra credit if over > 80% of the course do the evaluations.
  - The evaluations matter.
- ~~Guest speakers~~

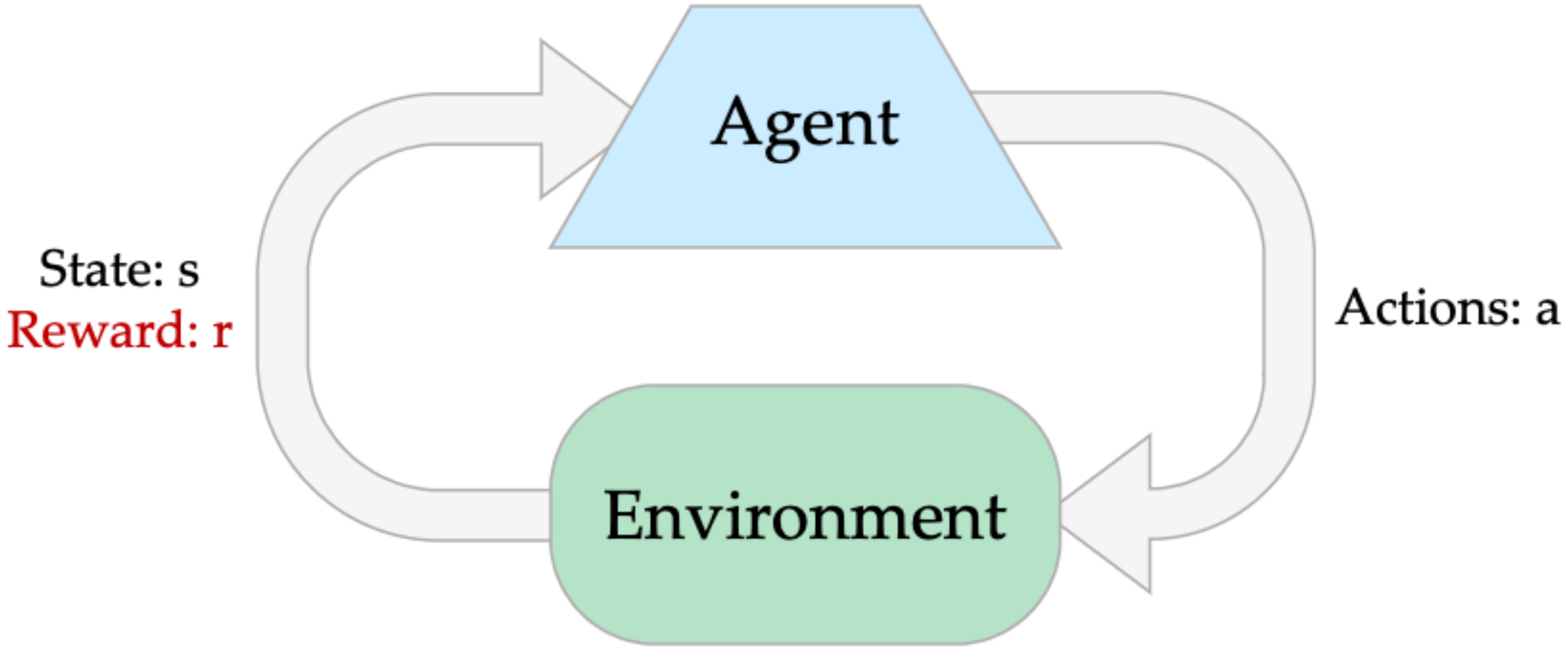
Agenda and Topics

- Reinforcement Learning
  - Motivation
  - Model-base Learning
  - Model-free Learning
    - Passive RL
    - Direct Evaluation
    - Temporal Difference Learning
    - Q-Learning (if time)

Reinforcement Learning

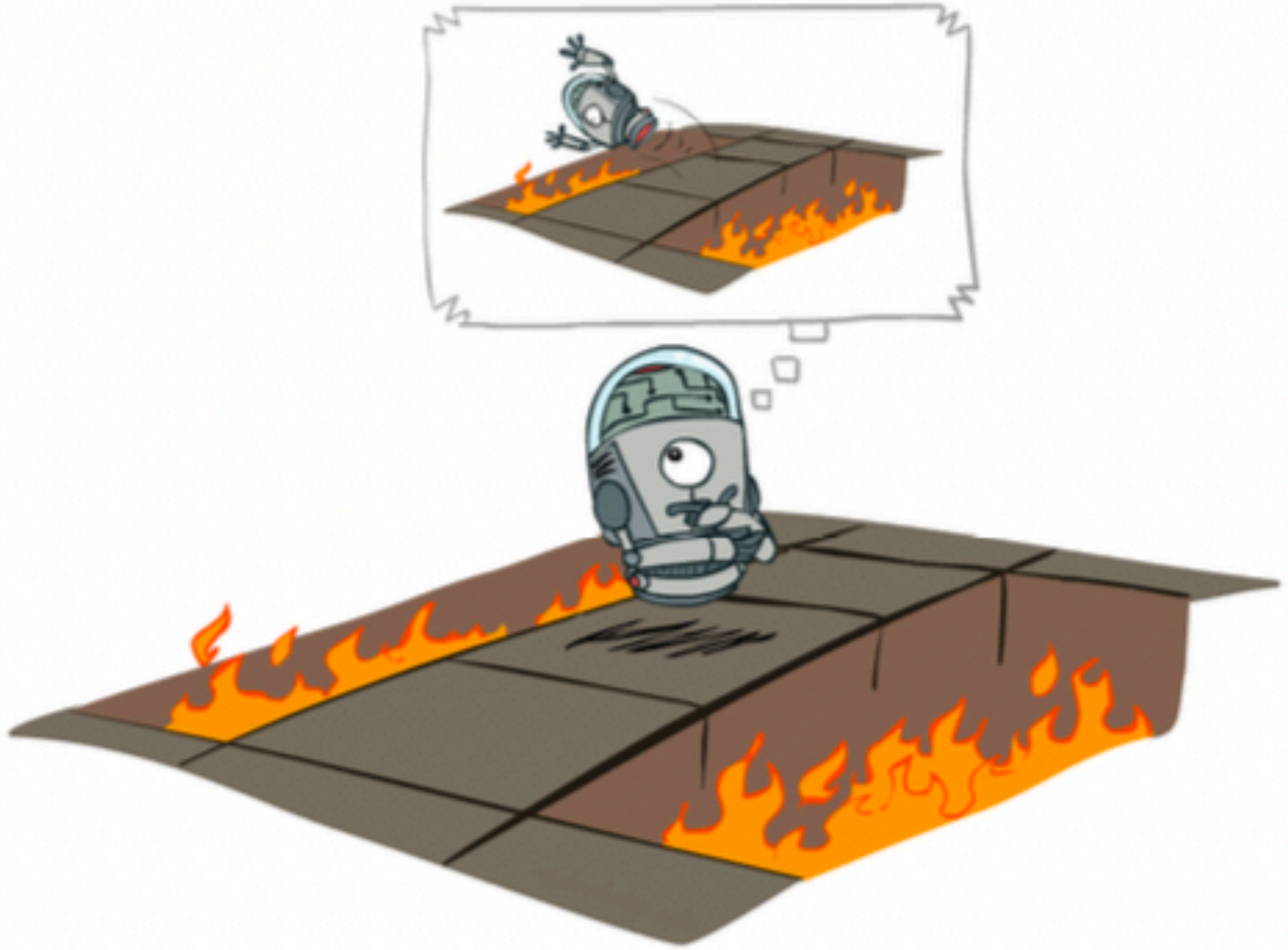
- Reinforcement learning:
  - Still assume an MDP:
    - A set of states $s \in S$
    - A set of actions (per state) A
    - A model $T(s,a,s')$
    - A reward function $R(s,a,s')$
  - Still looking for a policy $\pi(s)$
  - New twist: don't know T or R
    - i.e. don't know which states are good or what the actions do
    - Must actually try actions and states out to learn

Reinforcement Learning

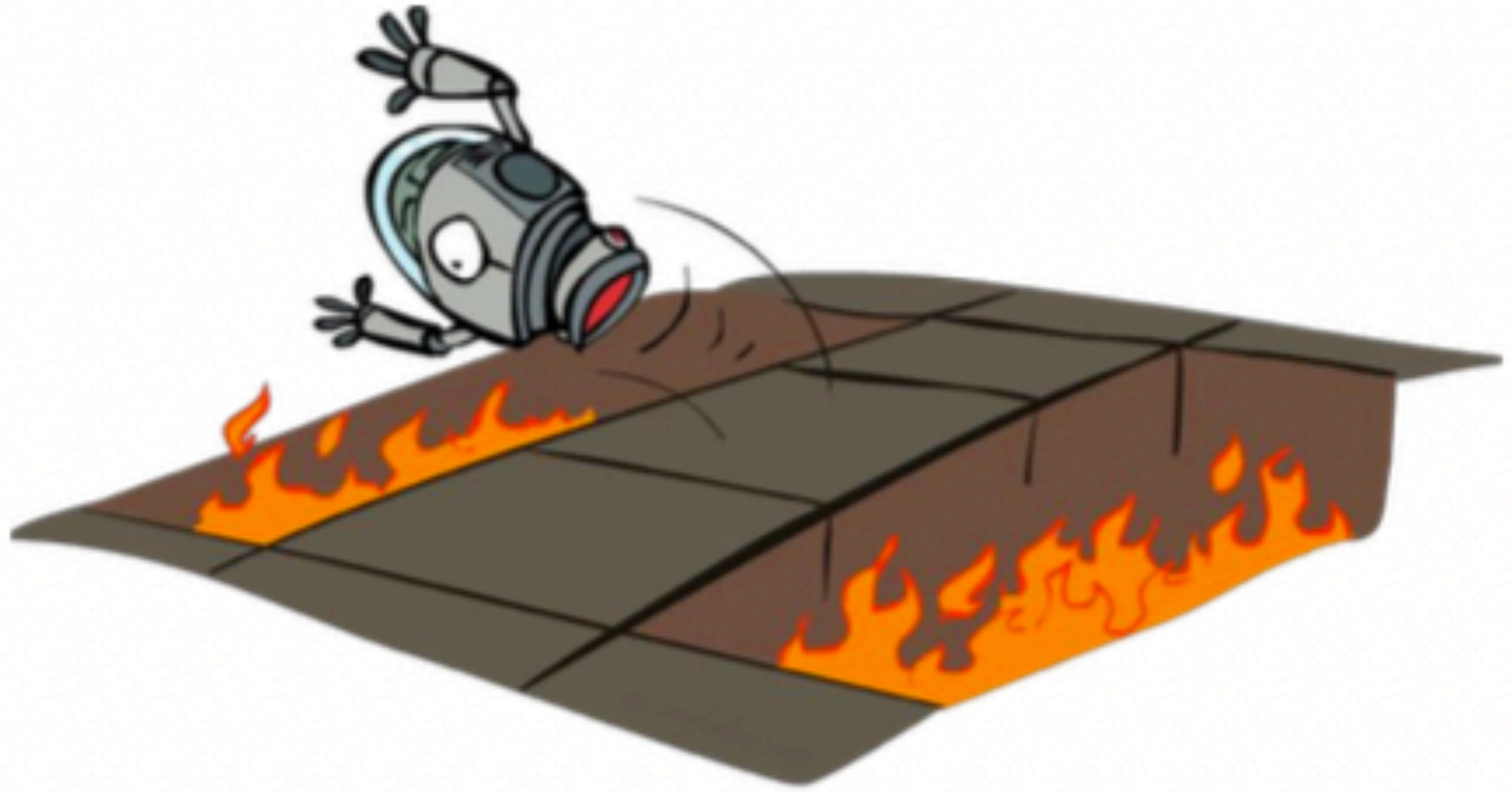


- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must (learn to) act so as to maximize expected rewards
  - All learning is based on observed samples of outcomes!

Offline (MDPs) vs. Online RL



Offline Solution



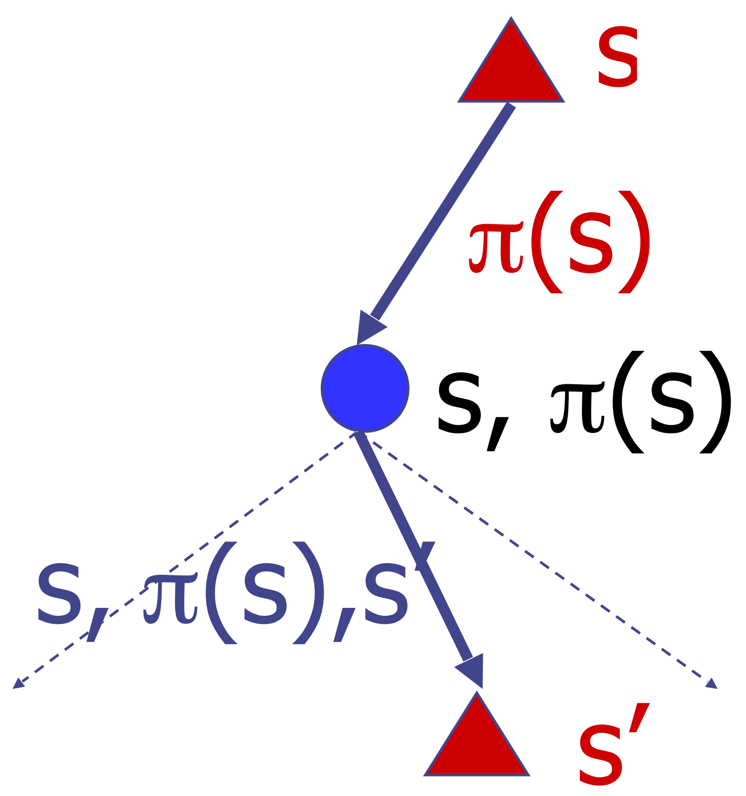
Online Learning

Model-Based Learning

Model-Based Learning

- Model-Based Idea:
  - Learn the model empirically through experience
  - Solve for values as if the learned model were correct

- Step 1: Learn empirical model learning
  - Count outcomes for each s,a
  - Normalize to give estimate of $\hat{T}(s, a, s')$
  - Discover $\hat{R}(s, a, s')$ when we experience (s,a,s')



- Solving the MDP with the learned model
  - Iterative policy evaluation, for example

$$V_{i+1}^{\pi}(s) \leftarrow \sum_{s'} \hat{T}(s, \pi(s), s')[\hat{R}(s, \pi(s), s') + \gamma V_i^{\pi}(s')]$$

Analogy: Expected Age

Goal: Compute expected age of students

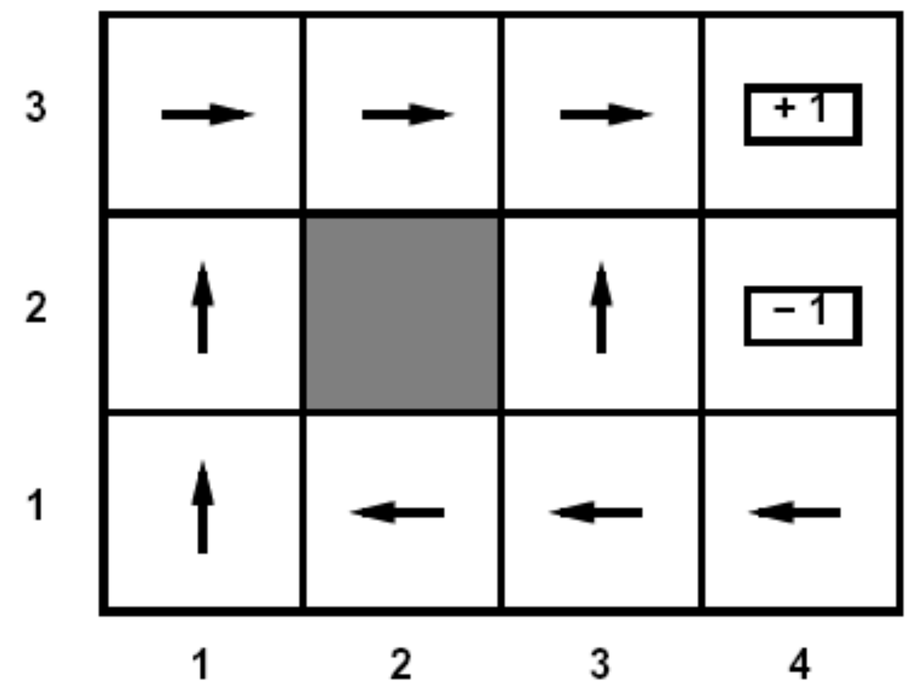
Known P(A)

$$E[A] = \sum_a P(a) \cdot a \qquad = 0.35 \times 20 + \ldots$$

Model-Free Learning

Passive Learning

- Simplified task
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - You are given a policy π(s)
  - Goal: learn the state values
  - … what policy evaluation did



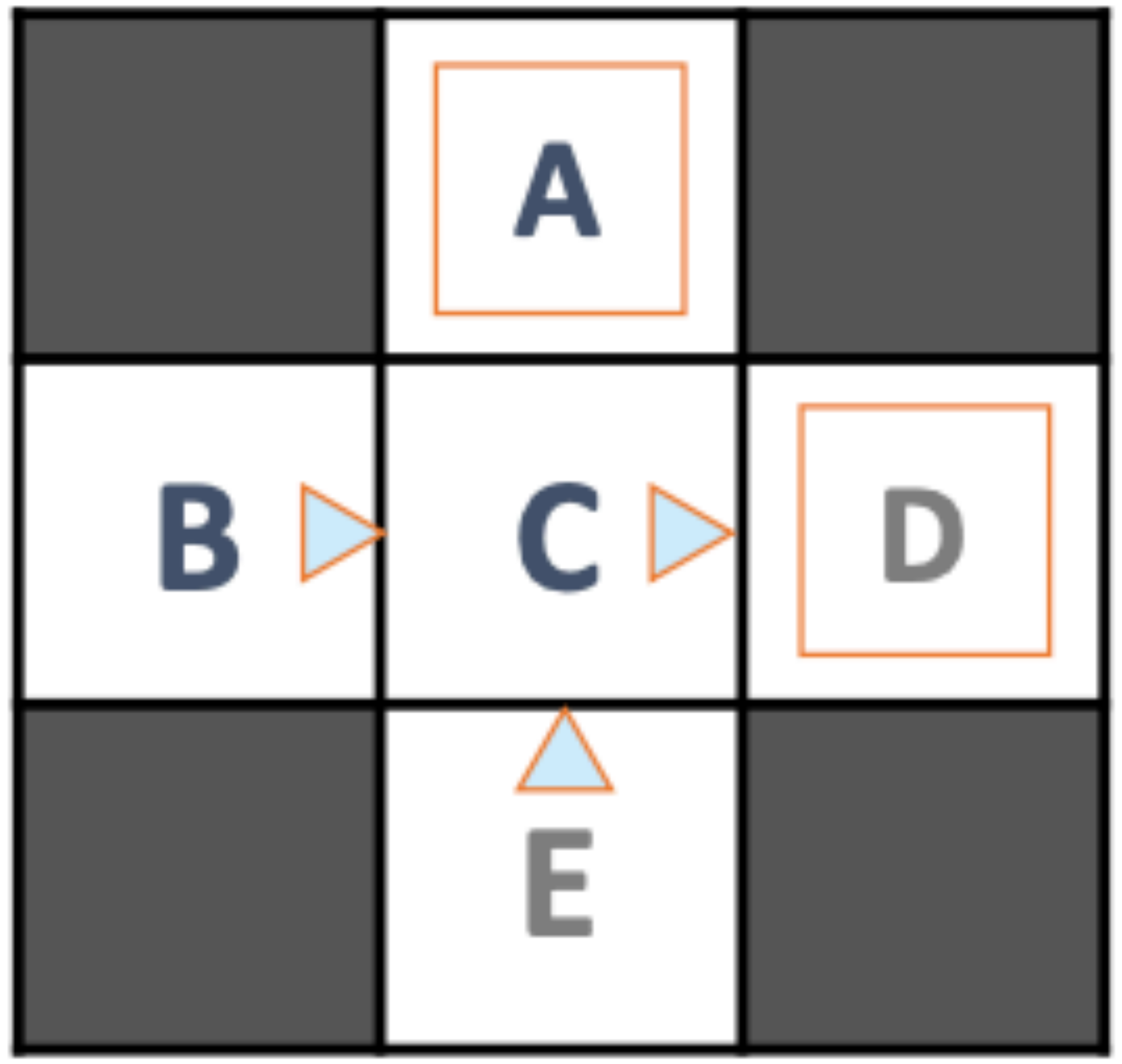
- In this case:
  - Learner "along for the ride"
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - We'll get to the active case soon
  - This is NOT offline planning! You actually take actions in the world and see what happens…

Direct Evaluation

- Goal: Compute values for each state under π.
- Idea: Average together observed sample values:
  - Act according to π
  - Every time you visit a state, write down what the sum of discounted rewards turned out to be.
  - Average those samples
- This is called direct evaluation

Example: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

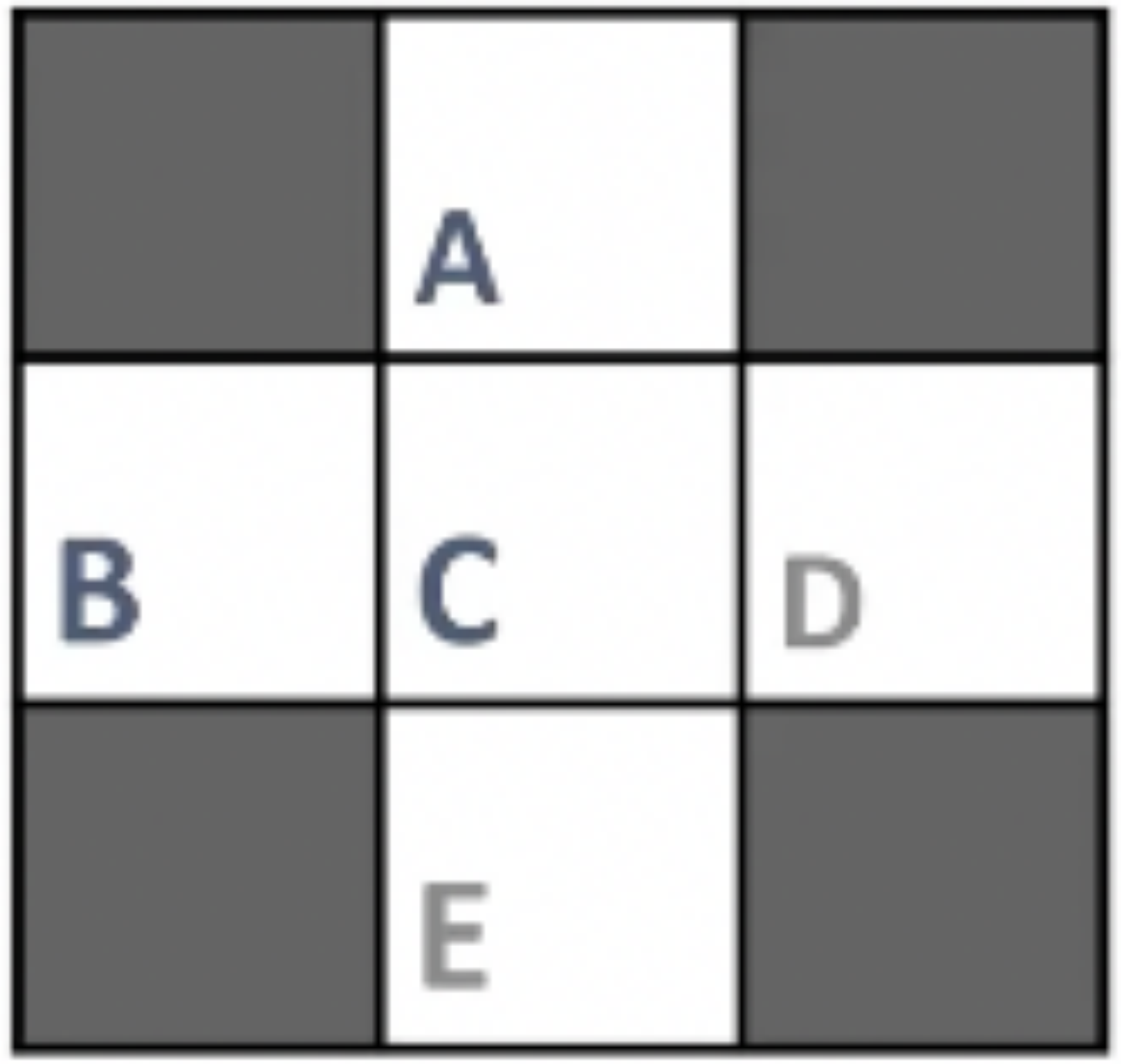
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

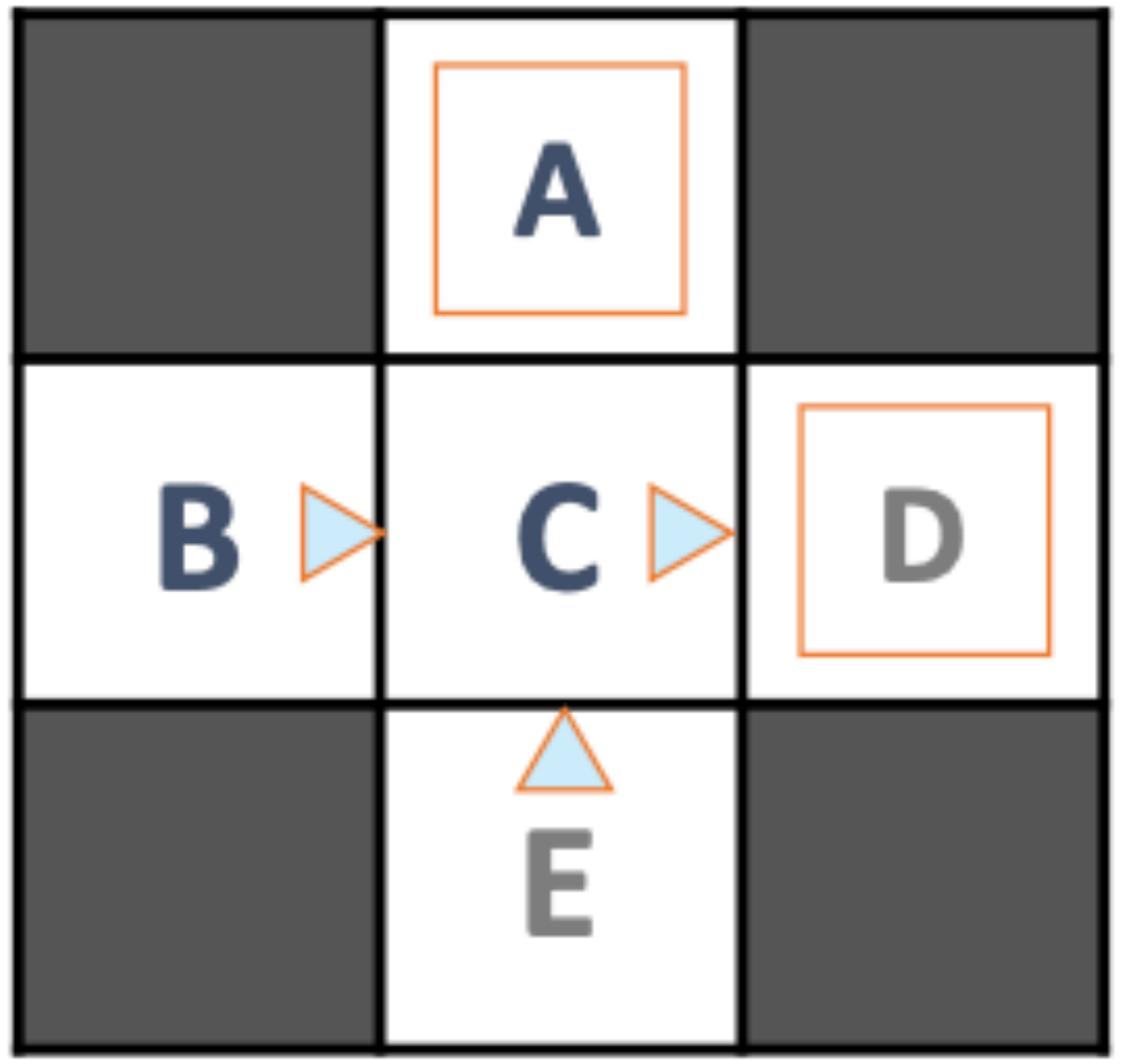
Output Values



CE 18: Calculate Output Values

For A, C, D

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

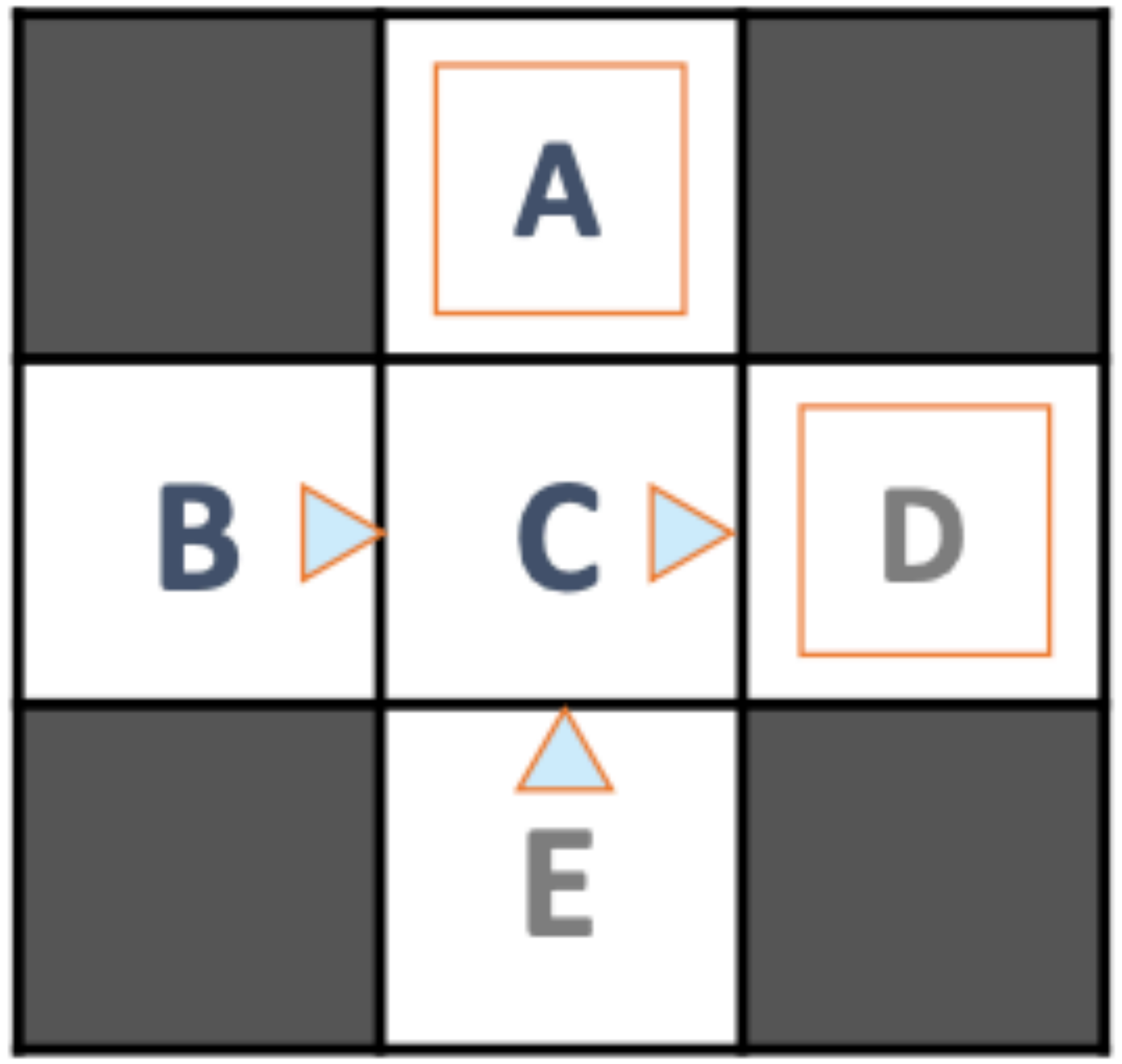
Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values

Example: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

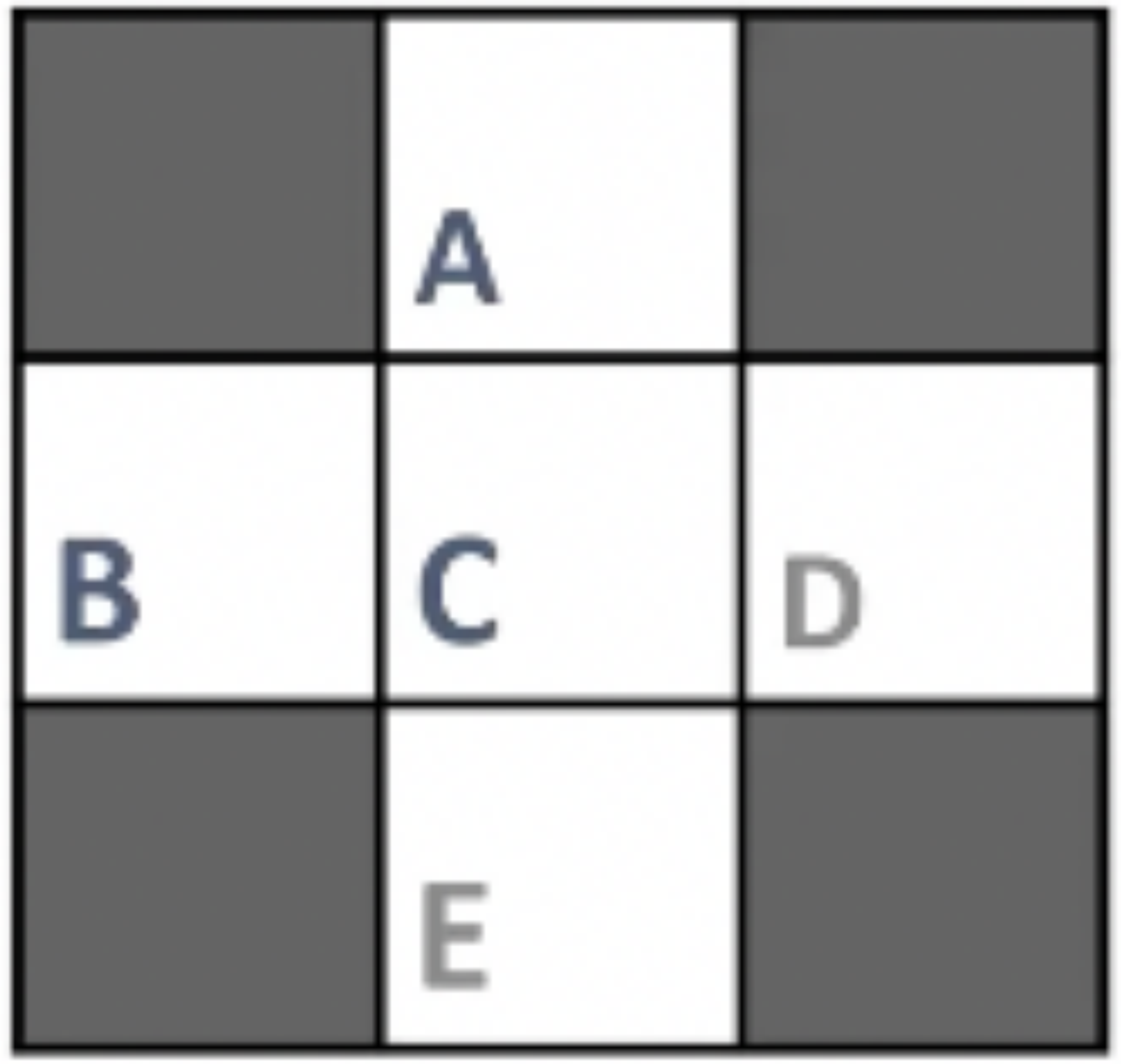
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

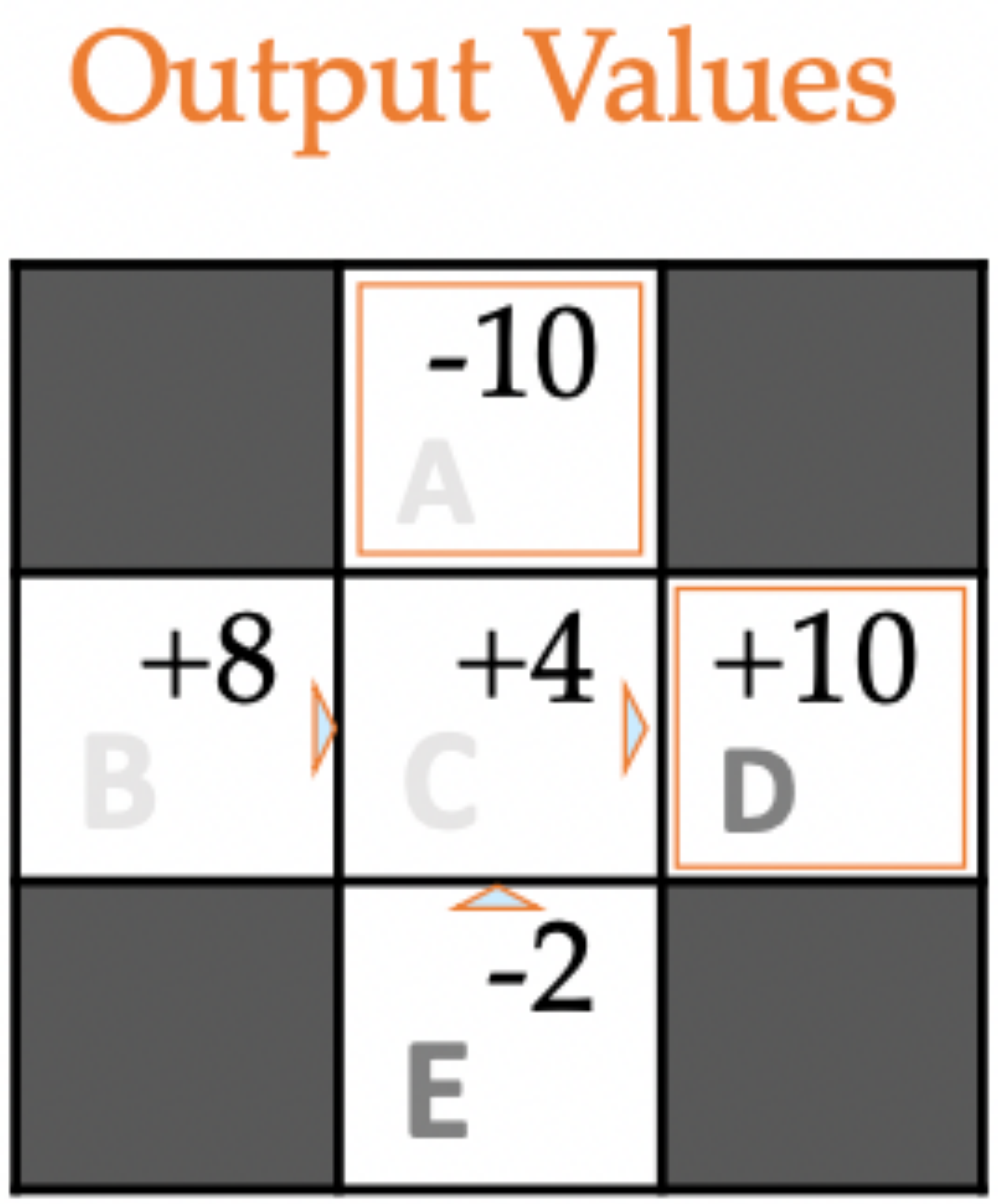
E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values



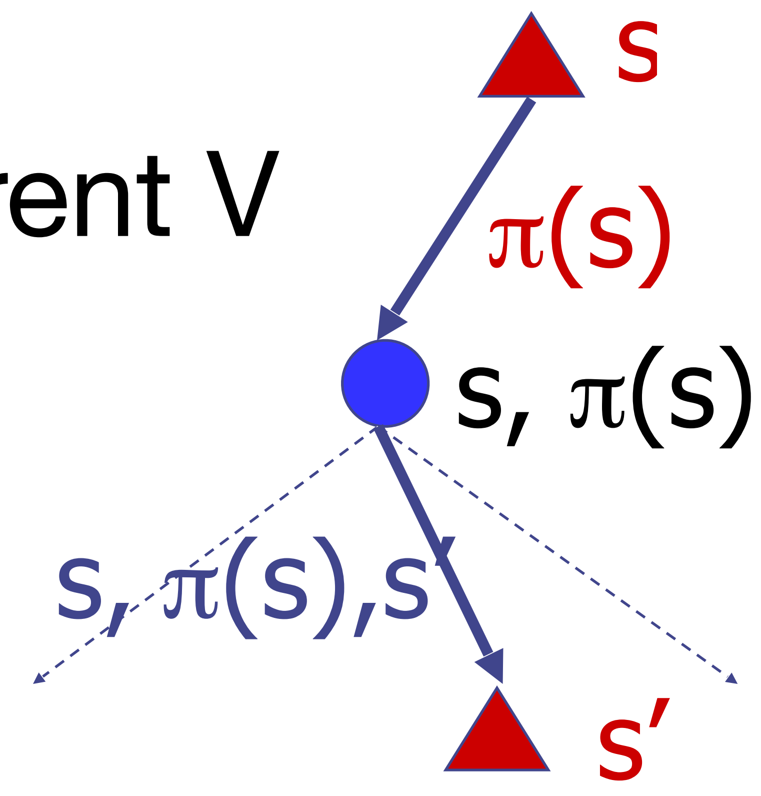
Problems with Direct Evaluation

- What's good about direct evaluation?
  - It's easy to understand
  - It doesn't require any knowledge of T, R
  - It eventually computes the correct average values using just sample transitions
- What is bad about it?
  - It wastes information about state connections
  - Each state must be learned separately
  - So, it takes a long time to learn



Why Not Use Policy Evaluation?

- Simplified Bellman updates to calculate V for a fixed policy:
  - New V is expected one-step-look-ahead using current V
  - Unfortunately, need T and R



$$V_0^\pi(s) = 0$$

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i^\pi(s')]$$

- Key question: how can we do this update to V without knowing T and R?
  - In other words, how to take a weighted average without knowing the weights?

Sample-Based Policy Evaluation?

- We want to improve our estimate of V by computing these averages:

$$V^{\pi}_{k+1}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')\left[R(s, \pi(s), s') + \gamma V^{\pi}_{k}(s')\right]$$

- Idea: Take samples of outcomes s' (by doing the action!) and average

$$sample_1 = R(s, \pi(s), s'_1 + \gamma V^{\pi}_{k}(s'_1)$$

$$sample_2 = R(s, \pi(s), s'_2 + \gamma V^{\pi}_{k}(s'_2)$$

$$\ldots$$

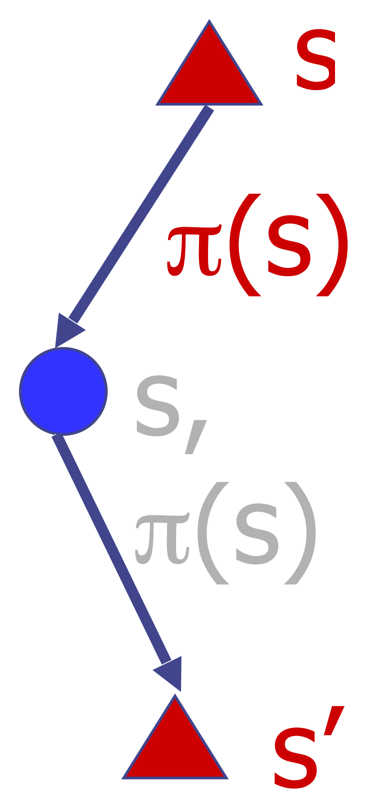
$$sample_n = R(s, \pi(s), s'_n + \gamma V^{\pi}_{k}(s'_n)$$

$$V^{\pi}_{k+1}(s) \leftarrow \frac{1}{n} \sum_{i} sample_i$$

Temporal Difference Learning

Temporal-Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience (s,a,s’,r)
  - Likely s’ will contribute updates more often



- Temporal difference learning
  - Policy still fixed!
  - Move values toward value of whatever successor occurs: running average!

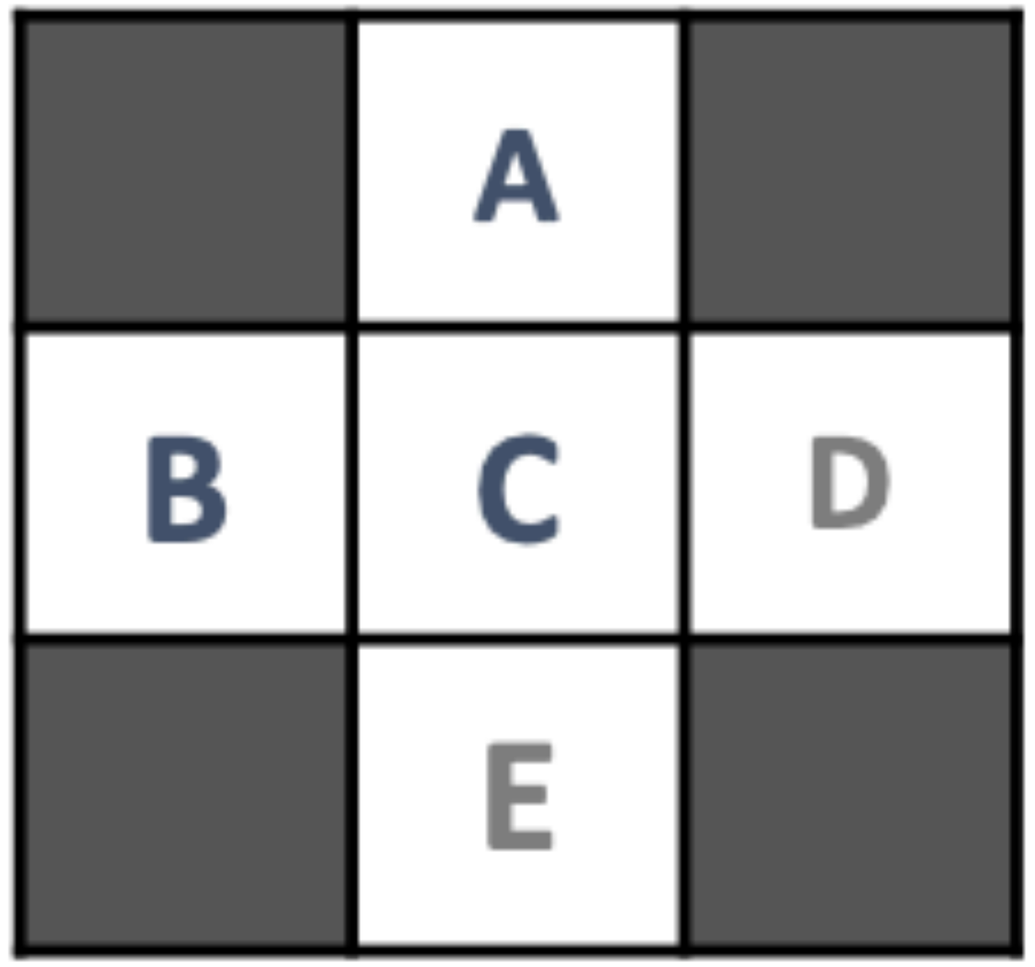
Sample of V(s): $$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

Update to V(s): $$V^{\pi}(s) \leftarrow (1 - \alpha)V^{\pi}(s) + (\alpha) sample$$

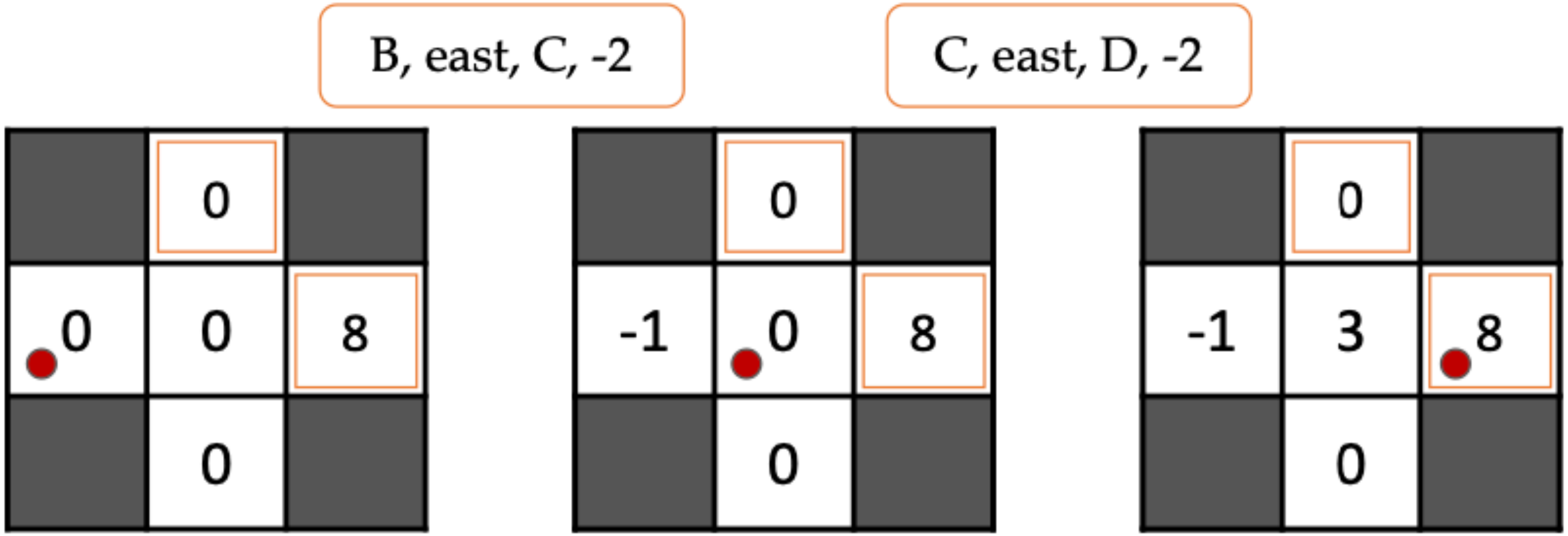
Same update: $$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$$

Example: Temporal Difference Learning

States



Observed Transitions

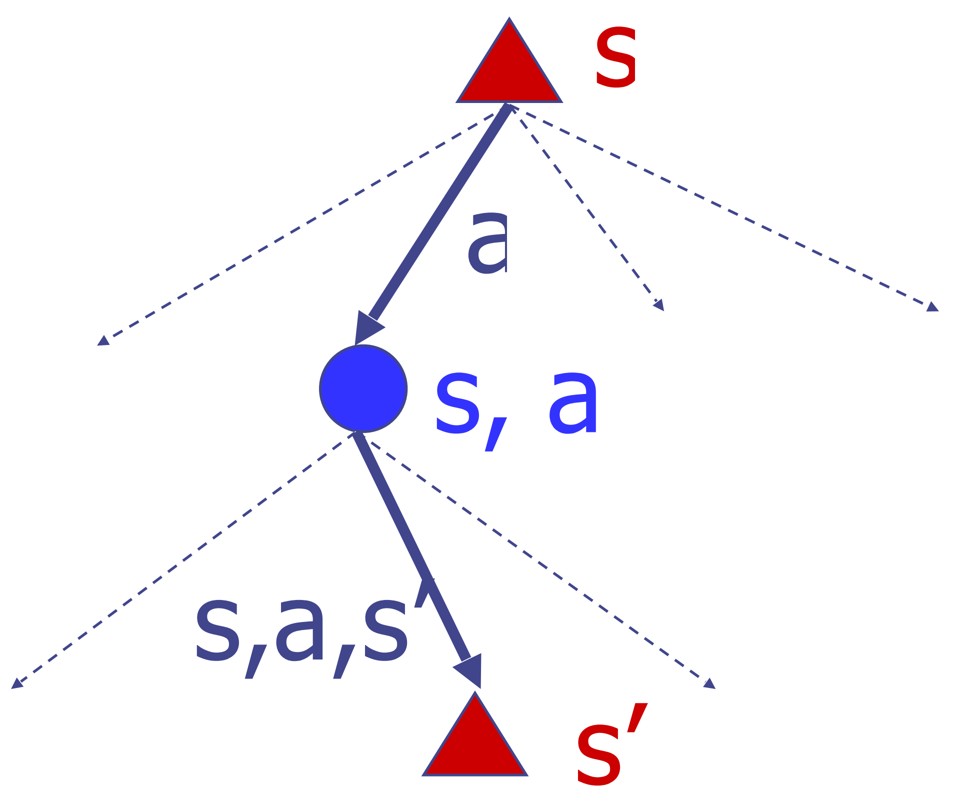


Assume: $\gamma = 1$, $\alpha = 1/2$

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

Problems with TD Value Learning

- TD value learning is a model-free way to do policy evaluation
- However, if we want to turn values into a (new) policy, we're stuck:



$$\pi(s) = \arg\max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

- Idea: learn Q-values directly
- Makes action selection model-free too!

Q-Learning

- Q-Learning: sample-based Q-value iteration
- Learn Q*(s,a) values
  - Receive a sample (s,a,s',r)
  - Consider your old estimate: $Q(s,a)$
  - Consider your new sample estimate:

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right]$$

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

  - Incorporate the new estimate into a running average:

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + (\alpha) \left[sample \right]$$

Summary

- Today: Reinforcement Learning
  - Motivation
  - Model-base Learning
  - Model-free Learning
    - Passive RL
    - Direct Evaluation
    - Temporal Difference Learning
    - Q-Learning (if time)
- Next week
  - Q-Learning
  - Deep RL
  - Neural networks