# Announcements

Assignments:

- HW7 (online)
  - Due today, 10 pm
- HW8 (written)
  - Will be released after HW7 is due. Due 10/29 Tue, 10 pm
- P4
  - Due 10/31 Thu, 10 pm

Recitation worksheet for last week's material is available online

Piazza in-class post is ready to go

Piazza Poll: Don't worry too much if you attended a lecture and missed one take of a poll or you missed a lecture which had many polls ☺. We will take that into account.

# AI: Representation and Problem Solving

## Reinforcement Learning


Instructors: Fei Fang & Pat Virtue

Slide credits: CMU AI and http://ai.berkeley.edu

# Learning Objective (RL I&II)

- Describe the relationships and differences between
  - Markov Decision Processes (MDP) vs Reinforcement Learning (RL)
  - Model-based vs Model-free RL
  - Temporal-Difference Value Learning (TD Value Learning) vs Q-Learning
  - Passive vs Active RL
  - Off-policy vs On-policy Learning
  - Exploration vs Exploitation
- Describe and implement
  - TD (Value) Learning
  - Q-Learning
  - $\epsilon$-Greedy algorithm
  - Approximate Q-learning (Feature-based)
- Derive weight update for Approximate Q-learning

Next Lecture

# MDP/RL Notation

Standard expectimax: $V(s) = \max_a \sum_{s'} P(s'|s,a)V(s')$

Bellman equations: $V(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$

Value iteration: $V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')], \quad \forall\, s$

Q-iteration: $Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')], \quad \forall\, s,a$

Policy extraction: $\pi_V(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')], \quad \forall\, s$

Policy evaluation: $V^{\pi}_{k+1}(s) = \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s') + \gamma V^{\pi}_k(s')], \quad \forall\, s$

Policy improvement: $\pi_{new}(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^{\pi_{old}}(s')], \quad \forall\, s$

TD (value) learning: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha\,[r + \gamma\, V^{\pi}(s') - V^{\pi}(s)]$

Q-learning: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

# Piazza Poll 1

Rewards may depend on any combination of *state*, *action*, *next state*.

Which of the following are valid formulations of the Bellman equations?

A. $V(s) = \max_{a} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$

B. $V(s) = R(s) + \gamma \max_{a} \sum_{s'} P(s'|s,a) V(s')$

C. $V(s) = \max_{a} [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s')$

D. $Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$

# Piazza Poll 1

Rewards may depend on any combination of *state*, *action*, *next state*.

Which of the following are valid formulations of the Bellman equations?

✓ *A.* $V(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$

✓ *B.* $V(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V(s')$

✓ *C.* $V(s) = \max_a [R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s')$

✓ *D.* $Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$

# Recall

## Which of the following are used in policy iteration?

Value iteration: $V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')]$, $\forall\, s$

Q-iteration: $Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$, $\forall\, s,a$

Policy extraction: $\pi_V(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$, $\forall\, s$

Policy evaluation: $V_{k+1}^{\pi}(s) = \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]$, $\forall\, s$

Policy improvement: $\pi_{new}(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^{\pi_{old}}(s')]$, $\forall\, s$

# Recall

## Which of the following are used in policy iteration?

Value iteration: $V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')]\,, \quad \forall\, s$

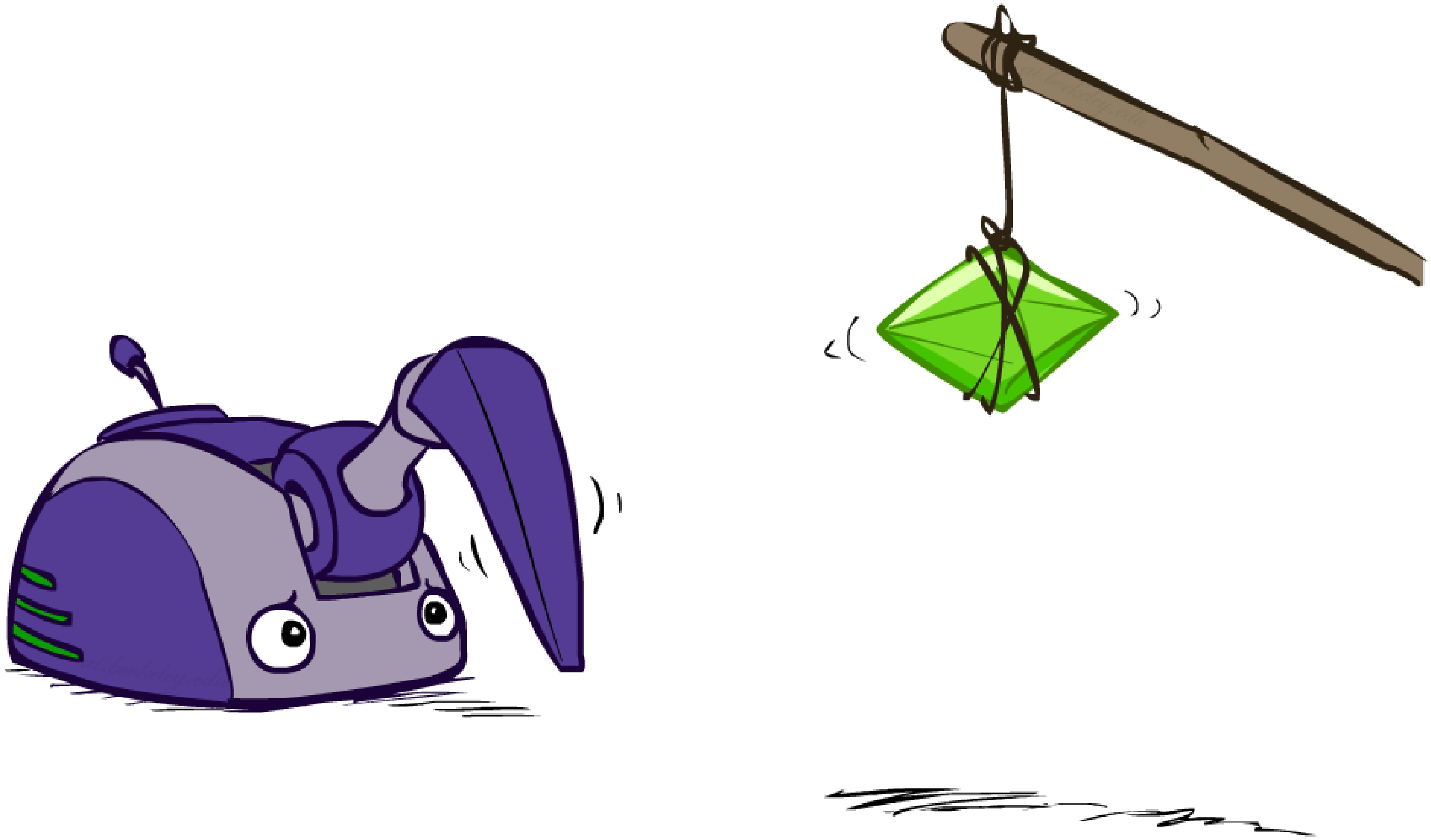
Q-iteration: $Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]\,, \quad \forall\, s,a$

Policy extraction: $\pi_V(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]\,, \quad \forall\, s$

✓ Policy evaluation: $V_{k+1}^{\pi}(s) = \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]\,, \quad \forall\, s$

✓ Policy improvement: $\pi_{new}(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^{\pi_{old}}(s')]\,, \quad \forall\, s$

# Reinforcement Learning



# Reinforcement learning

## What if we didn't know $P(s'|s,a)$ and $R(s,a,s')$?

Value iteration: $V_{k+1}(s) = \max_a \sum_{s'} \cancel{P(s'|s,a)}[\cancel{R(s,a,s')} + \gamma V_k(s')]\,, \quad \forall\, s$

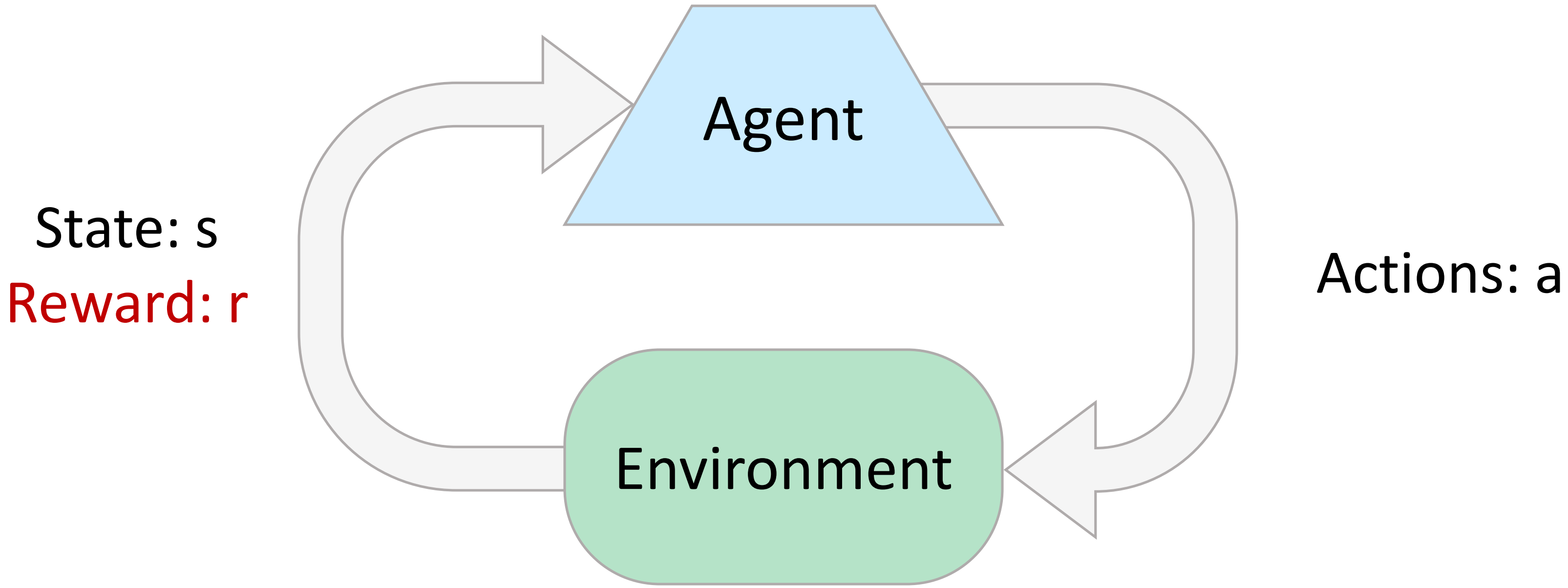
Q-iteration: $Q_{k+1}(s,a) = \sum_{s'} \cancel{P(s'|s,a)}[\cancel{R(s,a,s')} + \gamma \max_{a'} Q_k(s',a')]\,, \quad \forall\, s,a$

Policy extraction: $\pi_V(s) = \operatorname{argmax}_a \sum_{s'} \cancel{P(s'|s,a)}[\cancel{R(s,a,s')} + \gamma V(s')]\,, \quad \forall\, s$

Policy evaluation: $V^{\pi}_{k+1}(s) = \sum_{s'} \cancel{P(s'|s,\pi(s))}[\cancel{R(s,\pi(s),s')} + \gamma V^{\pi}_k(s')]\,, \quad \forall\, s$

Policy improvement: $\pi_{new}(s) = \operatorname{argmax}_a \sum_{s'} \cancel{P(s'|s,a)}[\cancel{R(s,a,s')} + \gamma V^{\pi_{old}}(s')]\,, \quad \forall\, s$

# 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!

# Example: Learning to Walk

Initial

A Learning Trial

After Learning [1K Trials]

[Kohl and Stone, ICRA 2004]

# Example: Learning to Walk

Initial

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – initial]

# Example: Learning to Walk

Training

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – training]

# Example: Learning to Walk

Finished

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – finished]

# Example: Sidewinding



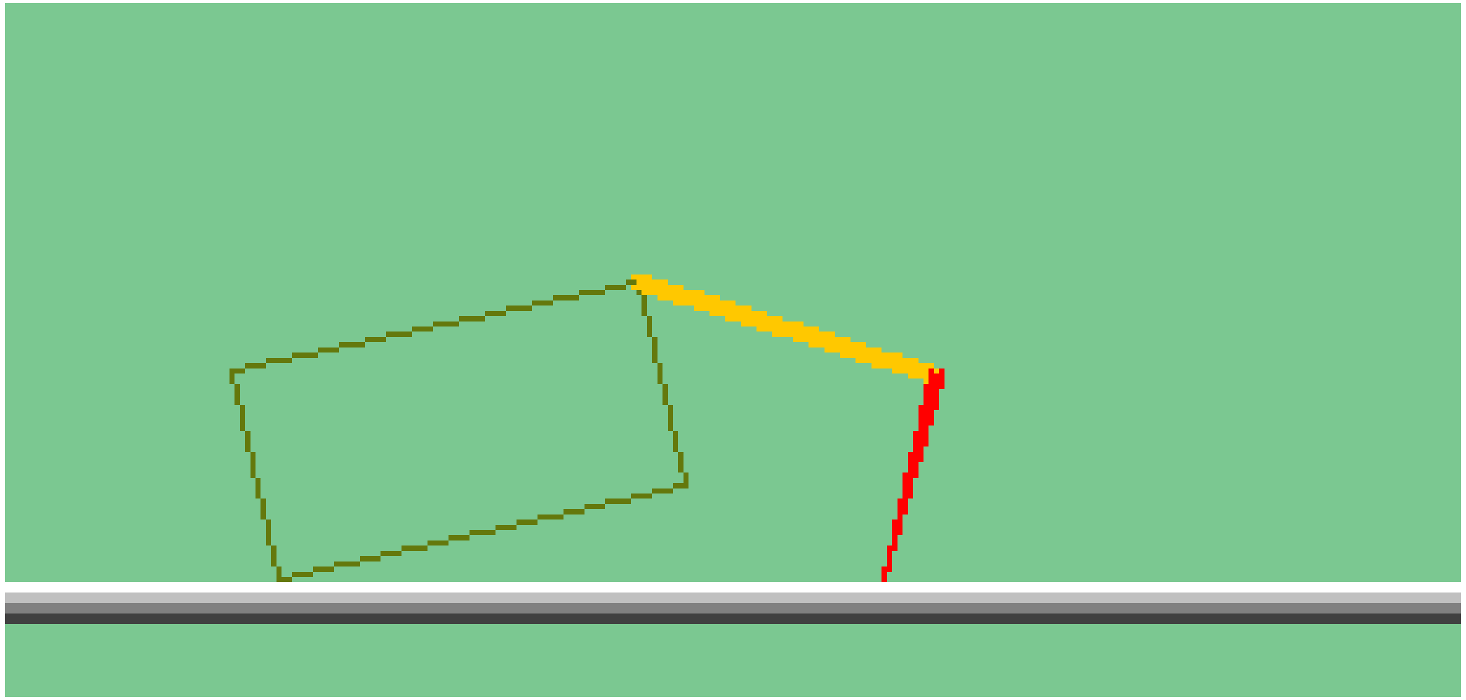
[Video: SNAKE – climbStep+sidewinding]

# Example: Toddler Robot

[Tedrake, Zhang and Seung, 2005]

[Video: TODDLER – 40s]

# The Crawler!



[Demo: Crawler Bot (L10D1)] [You, in Project 3]

# Demo Crawler Bot

# Reinforcement Learning

## Still assume a Markov decision process (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. we don’t know which states are good or what the actions do
- Must actually try actions and states out to learn

# Reinforcement Learning

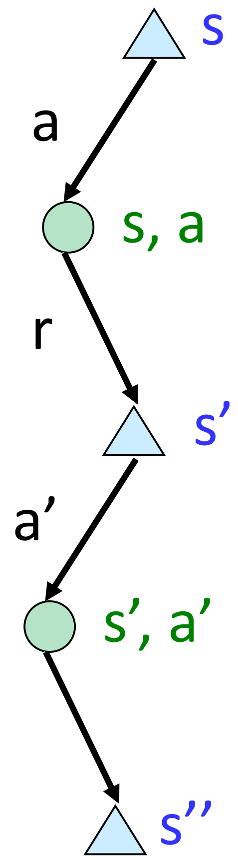
- Experience world through episodes

$$(s, a, r, s', a', r', s'', a'', r'', s'''' \ldots)$$

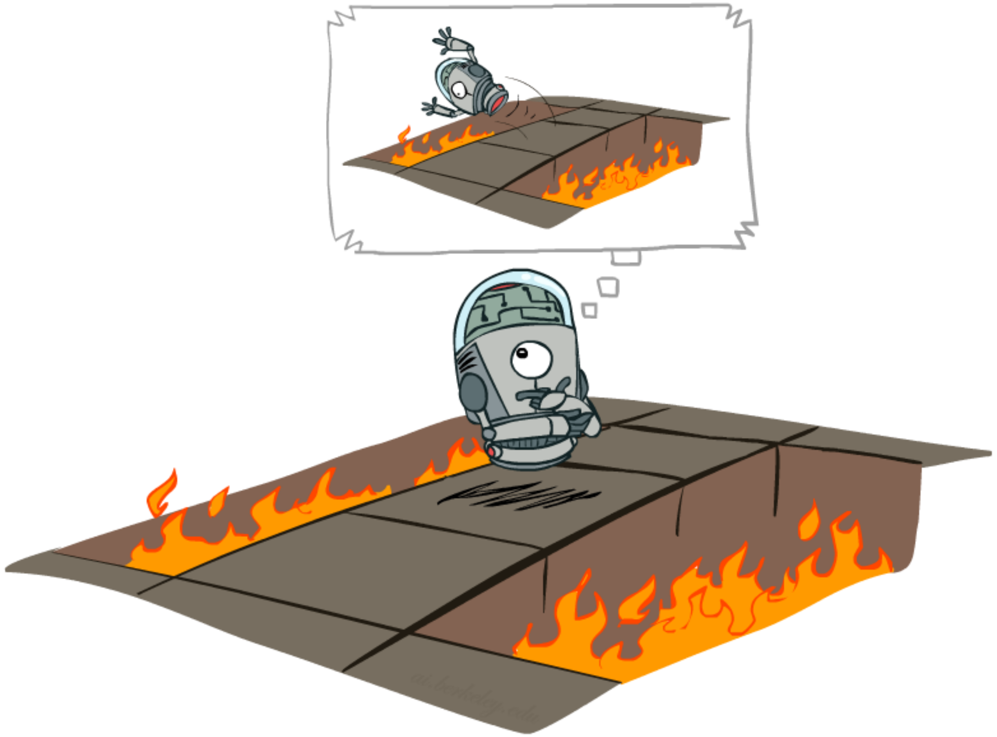
- You need many episodes ☺
- Learn from your experience

Key questions:

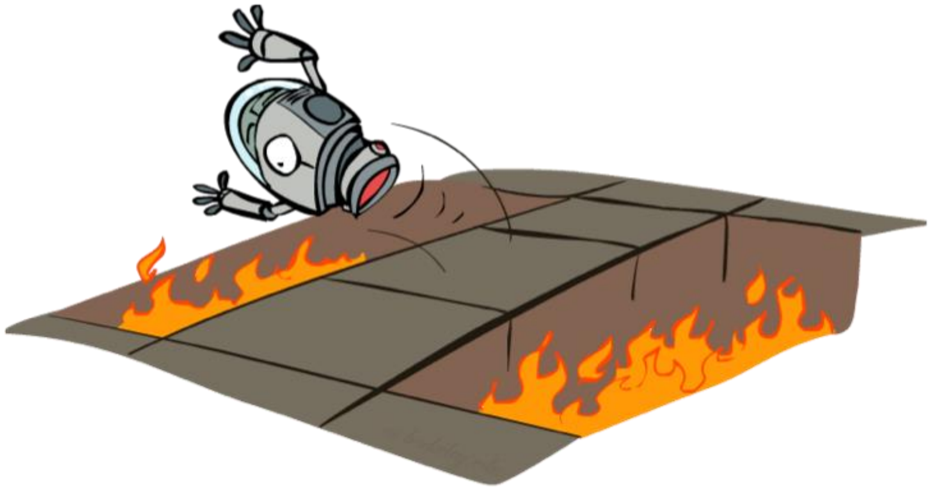
- When experiencing the world, how to take the actions?
- Given the experience, how to learn from it?



# Offline (MDPs) vs. Online (RL)

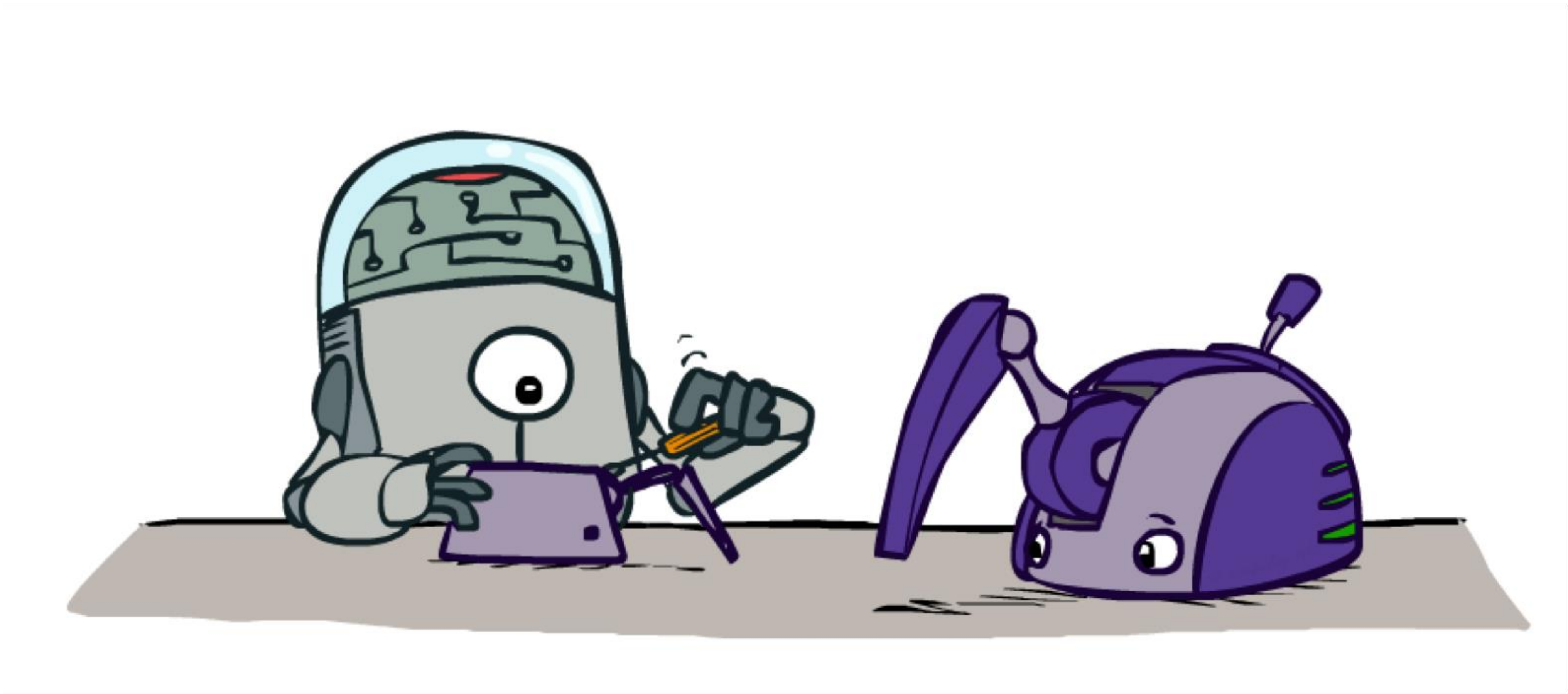


Planning offline



Learning to play online
(Trial and error)

# Model-Based Learning



# Model-Based Learning

## Model-Based Idea:

- Learn an approximate model based on experiences
- Solve for values as if the learned model were correct

## Step 1: Learn empirical MDP model

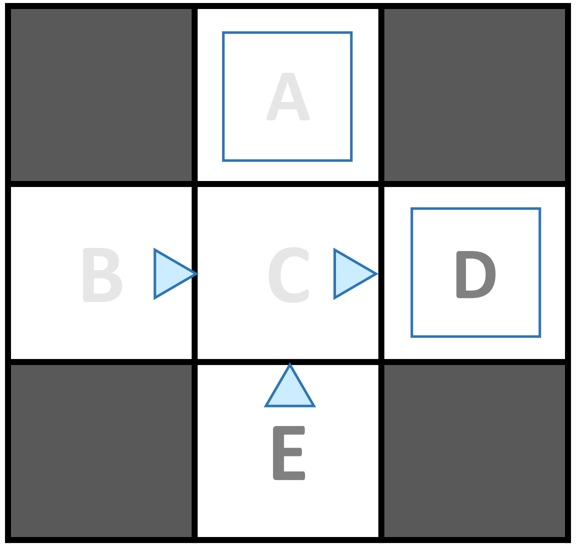
- Count outcomes s' for each s, a
- Normalize to give an estimate of $\widehat{T}(s, a, s')$
- Discover each $\widehat{R}(s, a, s')$ when we experience (s, a, s')

## Step 2: Solve the learned MDP

- For example, use value iteration, as before

# Example: Model-Based Learning

## A 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

## Learned Model

$$\hat{T}(s, a, s')$$

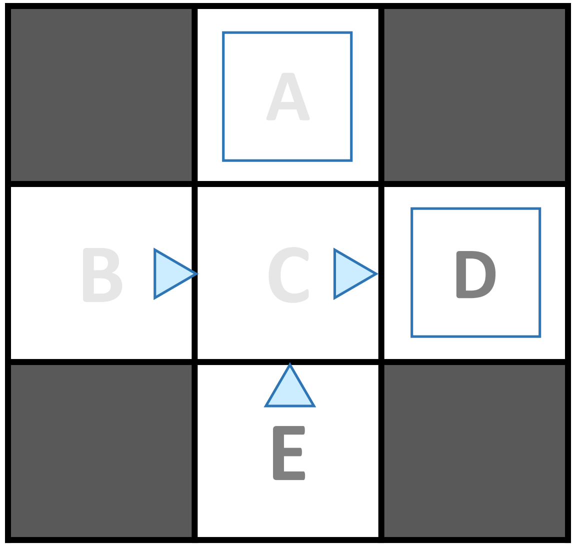
T(B, east, C) =
T(C, east, D) =
T(C, east, A) =
…

$$\hat{R}(s, a, s')$$

R(B, east, C) =
R(C, east, D) =
R(D, exit, x) =
…

# Example: Model-Based Learning

## A 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

## Learned Model

$$\hat{T}(s, a, s')$$

T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
…

$$\hat{R}(s, a, s')$$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
…

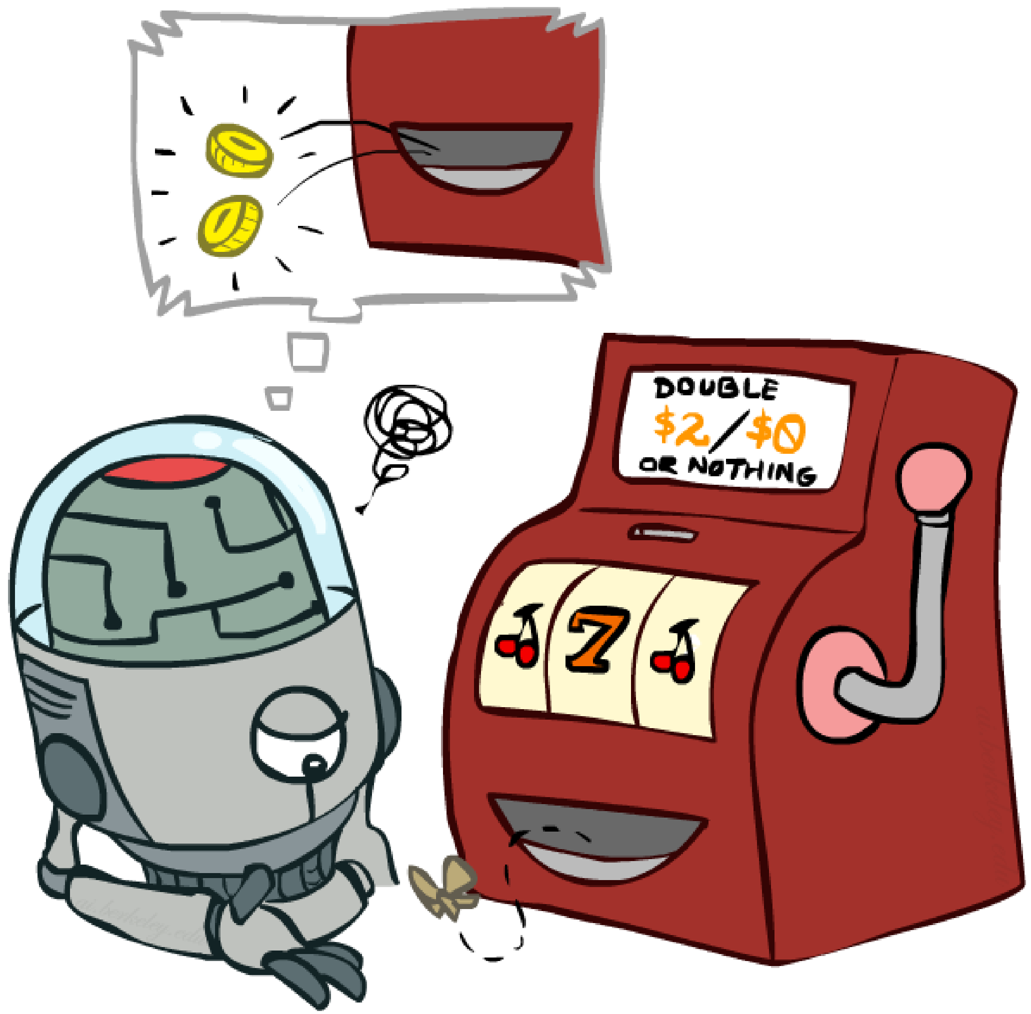
Any requirement for $\pi$ to learn a reasonable $\hat{T}$ and $\hat{R}$?

# Mid-Semester Feedback

5-min break

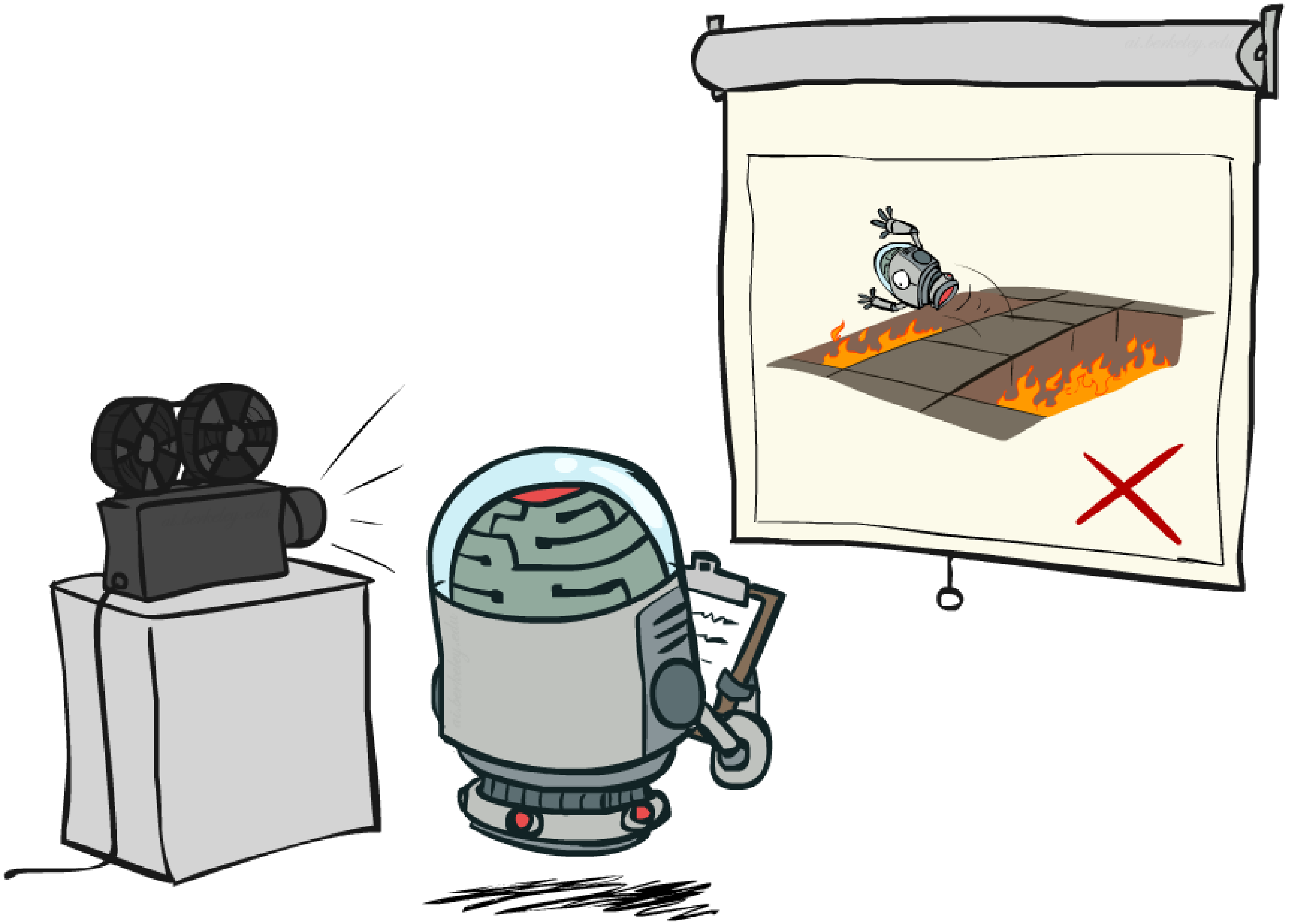
We would like to encourage you to fill the 15-281 mid-semester feedback forms online (links on Piazza, one for the course and one for the Tas)

# Model-Free Learning




How can we find the optimal policy without building an explicit MDP model (R and T)?

# Passive Reinforcement Learning



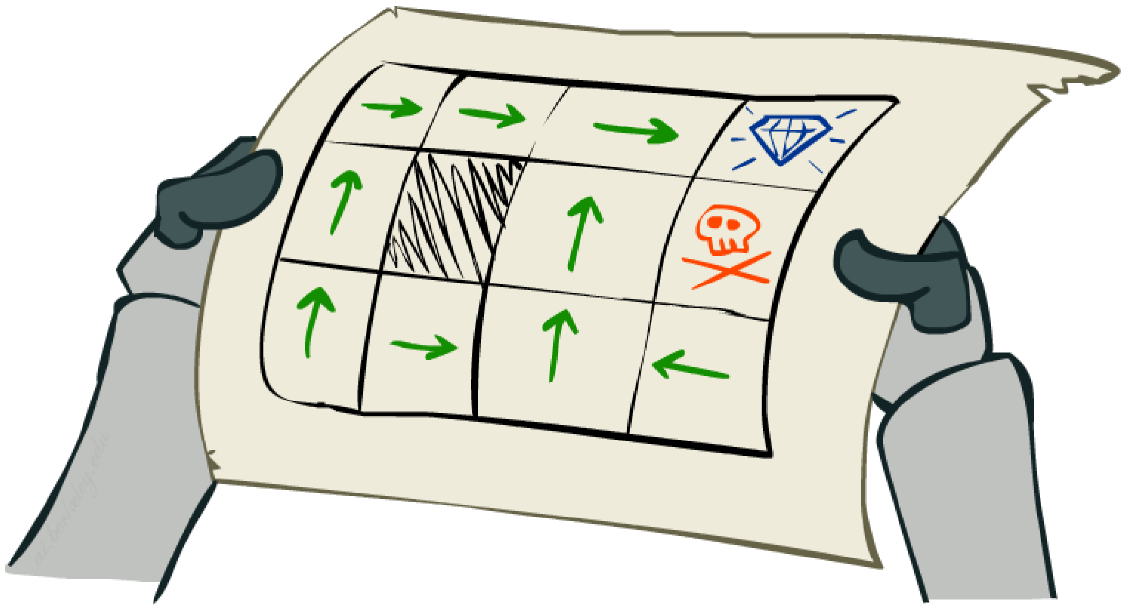
Given a policy $\pi$, learn how good it is.

"Passive" in the sense that the agent does not "choose" action itself.

# Passive Reinforcement Learning

## Simplified task: policy evaluation

- Input: a fixed policy $\pi(s)$
- You don't know the transitions T(s,a,s')
- You don't know the rewards R(s,a,s')
- Goal: learn the state values



## In this case:

- Learner is "along for the ride"
- No choice about what actions to take
- Just execute the policy and learn from experience
- This is NOT offline planning!  You actually take actions in the world.

# Direct Evaluation

Goal: Compute values for each state under $\pi$

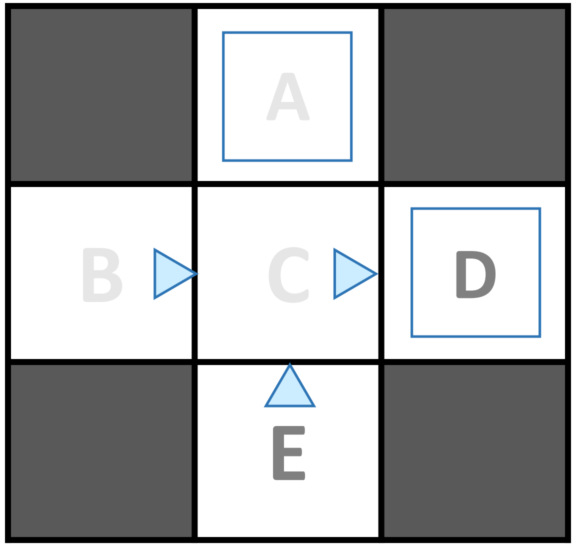
Idea: Average together observed sample values

- Act according to $\pi$
- 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

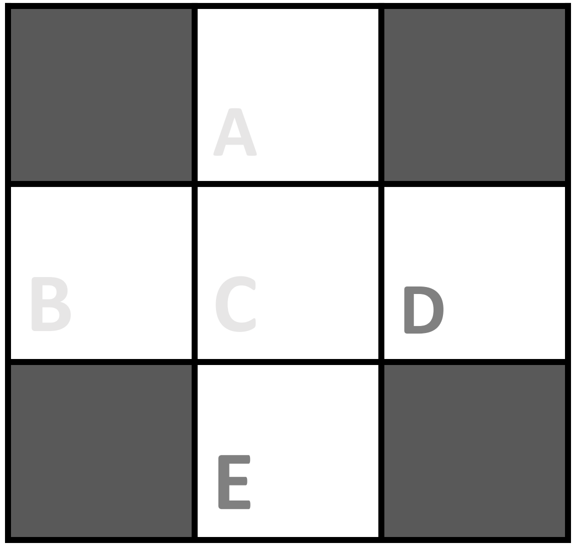
**Episode 3**

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

**Episode 4**

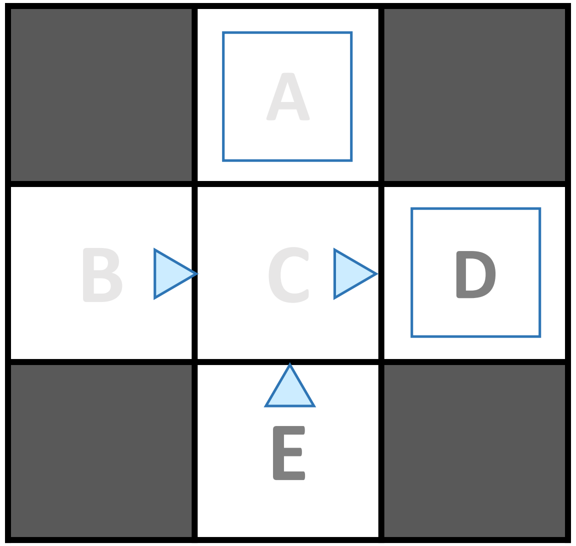
E, north, C, -1
C, east, A, -1
A, exit, x, -10

**Output Values**



# 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

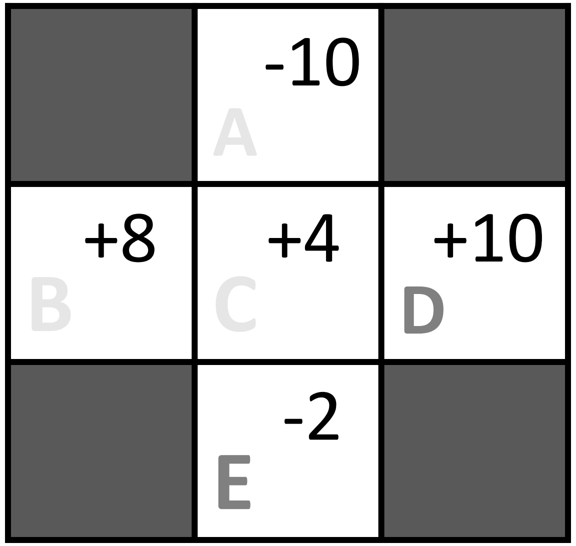
**Episode 3**

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

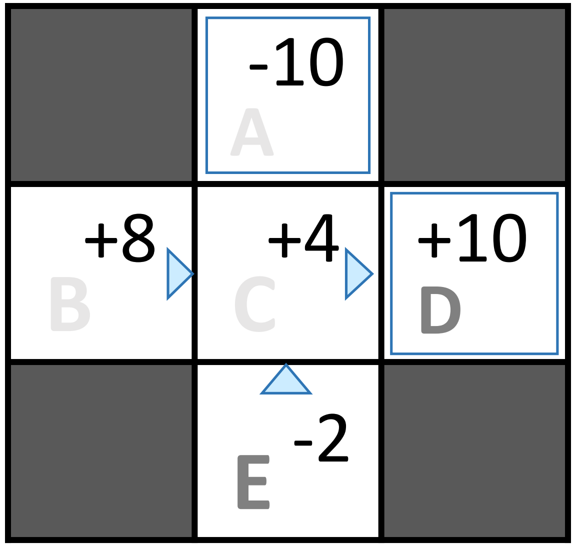
## What's good about direct evaluation?

- 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 bad about it?

- It wastes information about state connections (Markov property)
- Each state must be learned separately
- So, it takes a long time to learn

## Output Values




*If B and E both go to C under this policy, how can their values be different?*

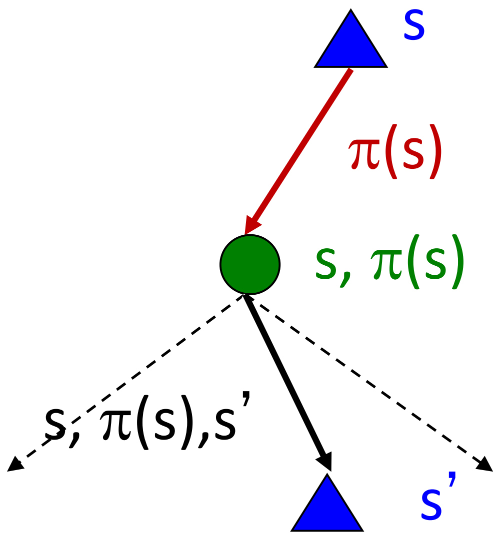
# Can We Use Policy Evaluation?

## Simplified Bellman updates calculate V for a fixed policy $\pi$:

- Each round, replace V with a one-step-look-ahead layer over V

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

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




- This approach fully exploited the connections between the states
- Unfortunately, we need T and R to do it!
- Luckily, you have access to the environment and you can try it out

## Key question: how can we do this update to V without knowing T and R?

# Can We Use Policy Evaluation?

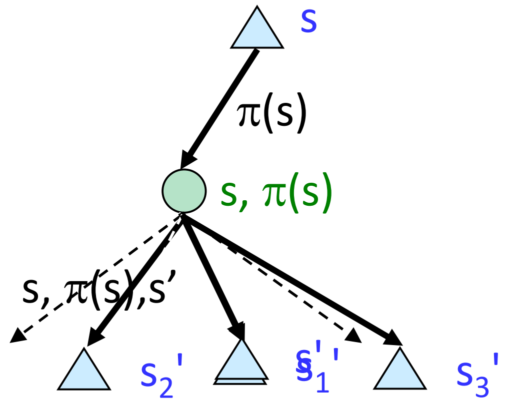
$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')], \forall s$$

How will you evaluate a biased coin / average age of students in 15-281?

First idea: Take samples of outcomes s' (by taking the action!) and average

$$sample_1 = R(s,\pi(s),s_1') + \gamma V_k^{\pi}(s_1')$$
$$sample_2 = R(s,\pi(s),s_2') + \gamma V_k^{\pi}(s_2')$$
$$\ldots$$
$$sample_n = R(s,\pi(s),s_n') + \gamma V_k^{\pi}(s_n')$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n}\sum_i sample_i$$




*Almost! But we can't rewind time to get sample after sample from state s.*

# Can We Use Policy Evaluation?

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

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$$

In the extreme case, we can just take one sample ($n = 1$)

$$V_{k+1}^{\pi}(s) \leftarrow sample$$

But this is very high variance!

Second idea: Make use of the value of $V_k^{\pi}(s)$. Use running average.

$$V_{k+1}^{\pi}(s) \leftarrow (1 - \alpha) V_k^{\pi}(s) + \alpha \times sample$$

# Exponential Moving Average

## Exponential moving average

- The running interpolation update: $\bar{x}_n = (1-\alpha)\cdot\bar{x}_{n-1} + \alpha\cdot x_n$
- Makes recent samples more important:

$$\bar{x}_n = \frac{x_n + (1-\alpha)\cdot x_{n-1} + (1-\alpha)^2\cdot x_{n-2} + \dots}{1 + (1-\alpha) + (1-\alpha)^2 + \dots}$$

- Forgets about the past (distant past values were wrong anyway)

## Decreasing learning rate (alpha) can give converging averages

# Can We Use Policy Evaluation?

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

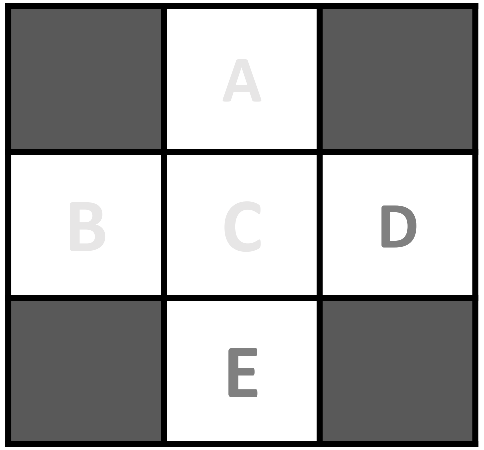
You only have access to a stochastic environment.

You cannot fully control which state you will be at and directly jump to each of the states one by one to update $V^{\pi}$.

Third idea: Only update one state $s$ at a time (the state you are in) as you try out the policy in the environment

B, east, C, -2

$$sample = R(s,\pi(s),s') + \gamma V_k^{\pi}(s') = -2 + 1 * V^{\pi}(C) = -2$$



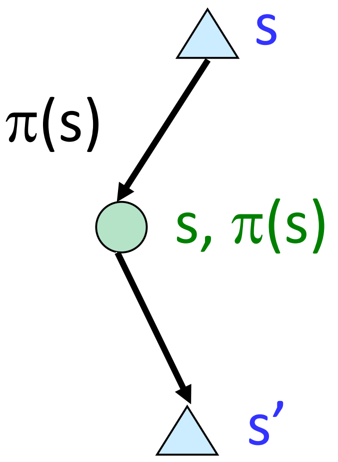
# Temporal Difference (Value) Learning

Putting the ideas together: Temporal Difference (Value) Learning!

- Task: Given policy $\pi$, learn state value $V^{\pi}$

Learn from every experience

- Update $V^{\pi}(s)$ each time we experience a transition $(s, a, s', r)$
- Likely outcomes $s'$ will contribute updates more often
- Move values toward latest sample (running average)




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

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

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