Lecture 16

Instructor: Amit Kumar Das

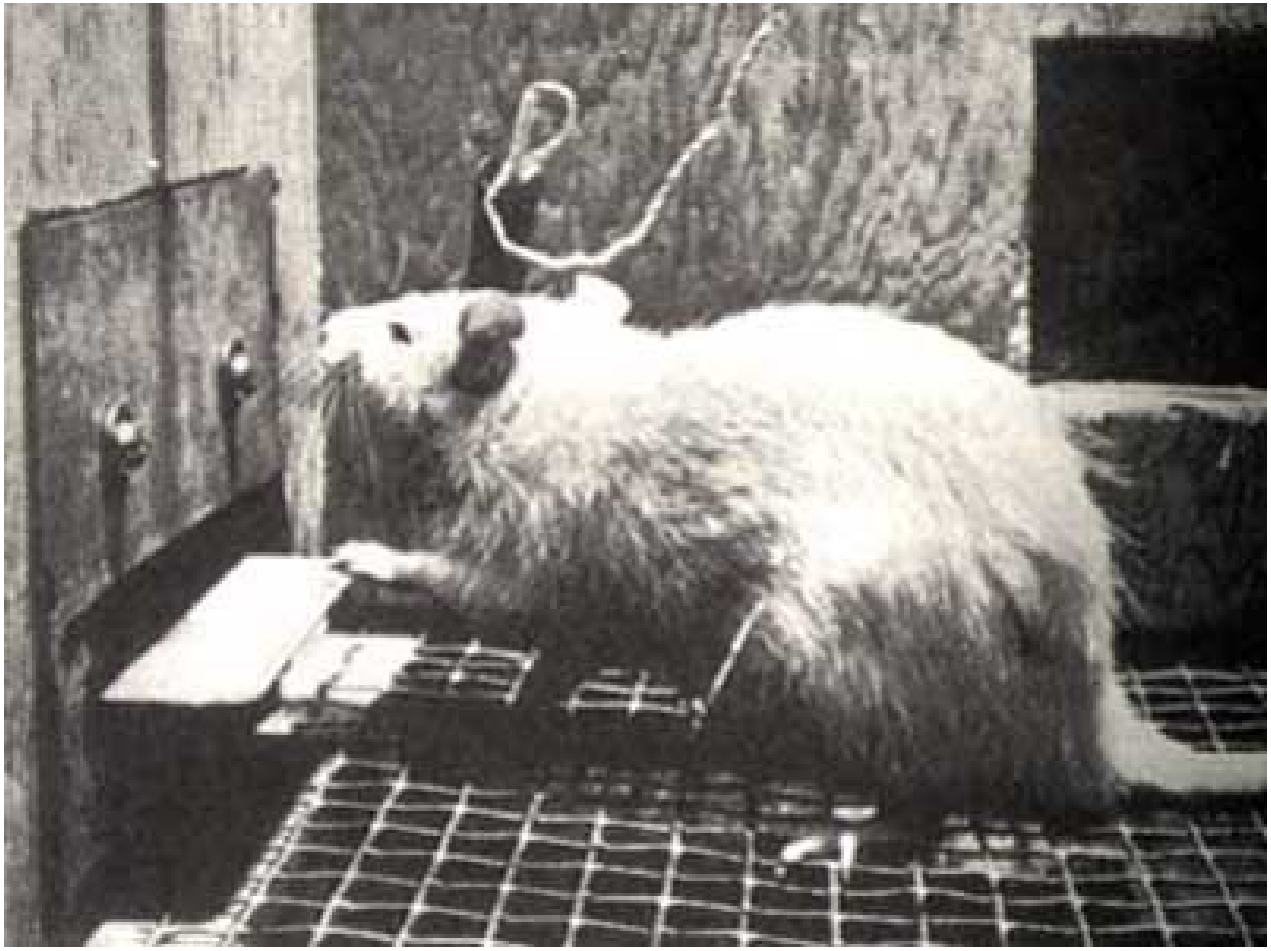
Senior Lecturer,

Department of Computer Science & Engineering,

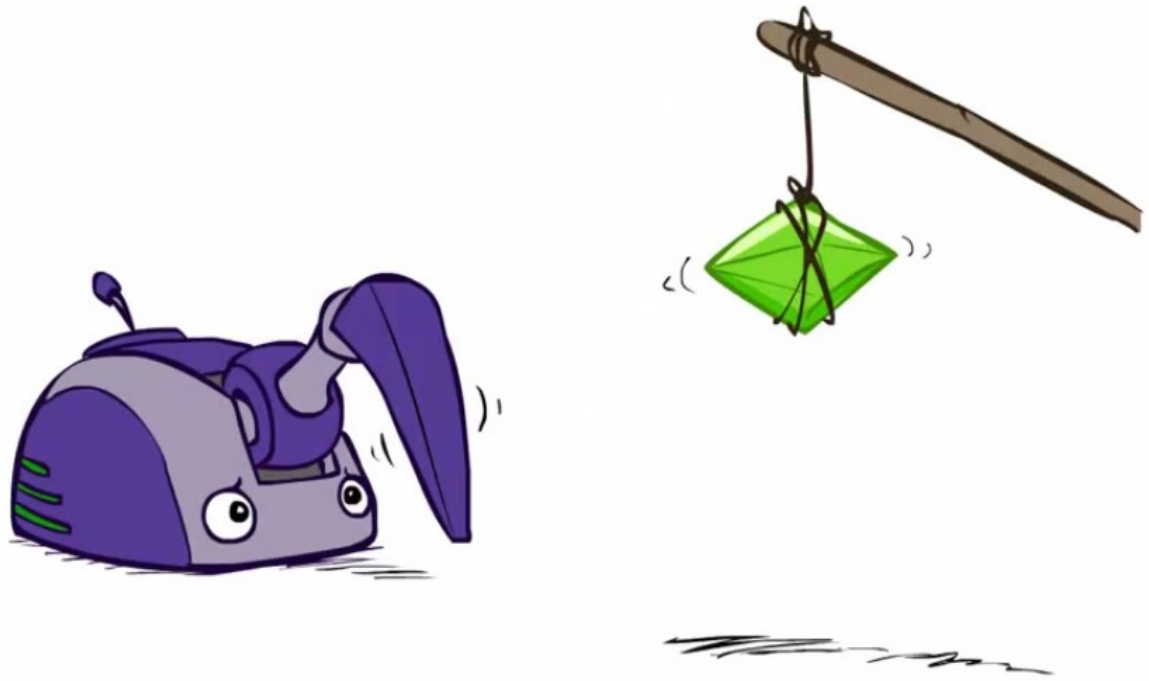
East West University

Dhaka, Bangladesh.

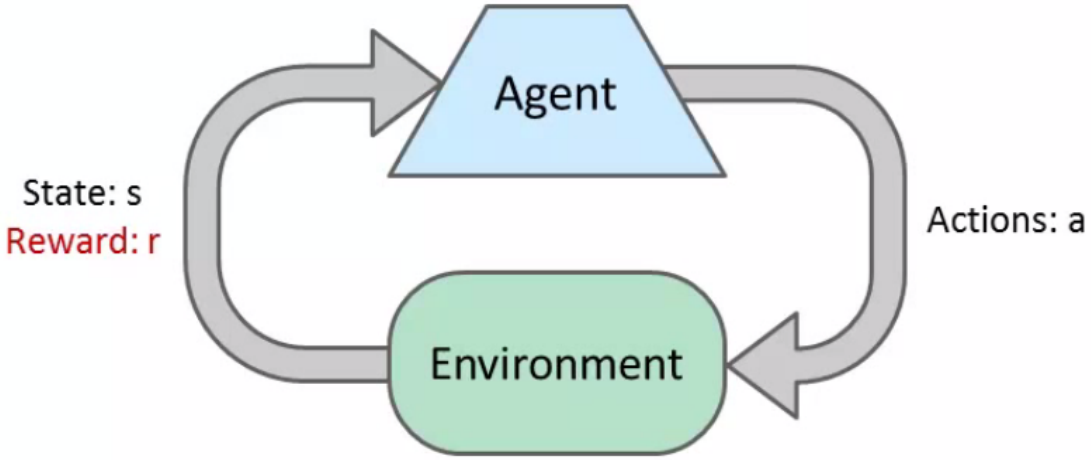
Reinforcement learning (Chapter 21)



Reinforcement Learning



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!

Reinforcement learning

- **Regular MDP**
  - Given:
    - Transition model $P(s' \mid s, a)$
    - Reward function $R(s)$
  - Find:
    - Policy $\pi(s)$

- **Reinforcement learning**
  - Transition model and reward function initially unknown
  - Still need to find the right policy
  - "Learn by doing"

Reinforcement learning: Basic scheme

- In each time step:
  - Take some action
  - Observe the outcome of the action: successor state and reward
  - Update some internal representation of the environment and policy
  - If you reach a terminal state, just start over (each pass through the environment is called a *trial*)

- Why is this called reinforcement learning?

Applications of reinforcement learning

- Backgammon

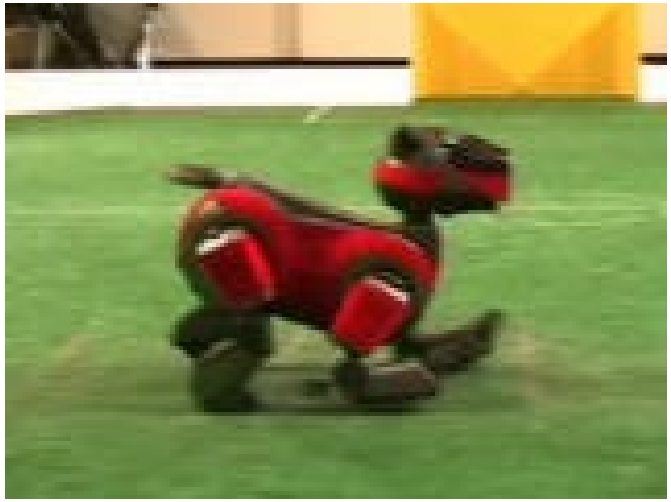
http://www.research.ibm.com/massive/tdl.html

http://en.wikipedia.org/wiki/TD-Gammon

Applications of reinforcement learning

- Learning a fast gait for Aibos

Initial gait



Learned gait

Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion
Nate Kohl and Peter Stone.
IEEE International Conference on Robotics and Automation, 2004.

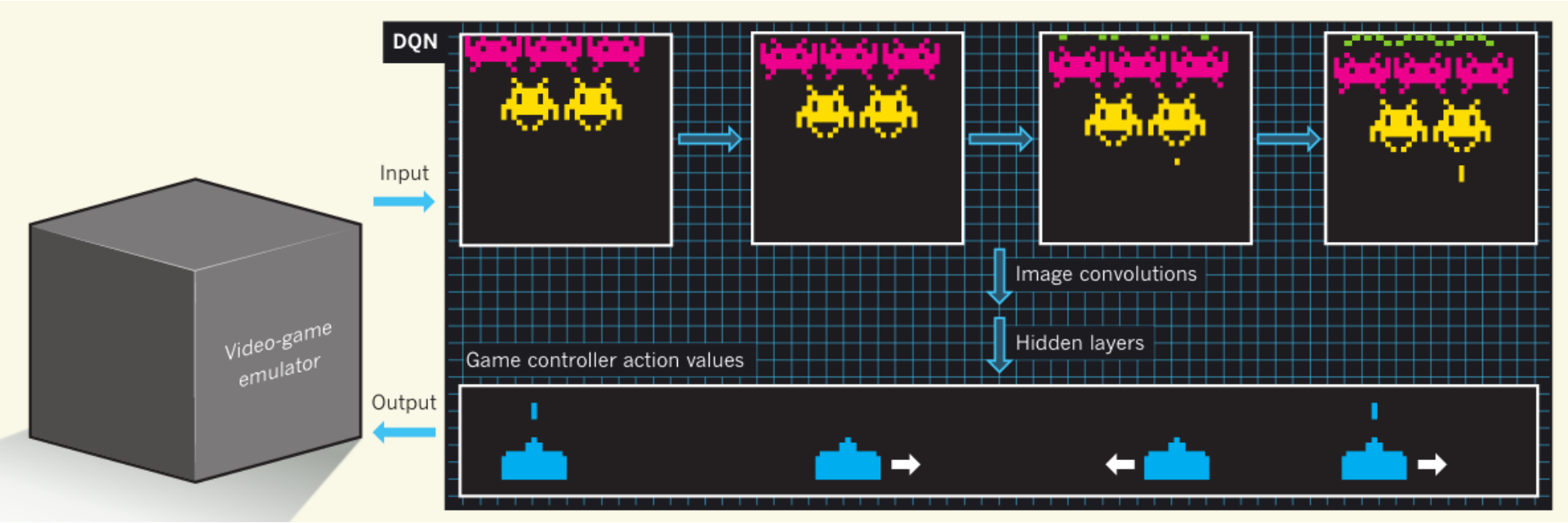
Applications of reinforcement learning

- Stanford autonomous helicopter

Pieter Abbeel et al.

Applications of reinforcement learning

- Playing Atari with deep reinforcement learning



Video

V. Mnih et al., *Nature*, February 2015

Applications of reinforcement learning

- End-to-end training of deep visuomotor policies

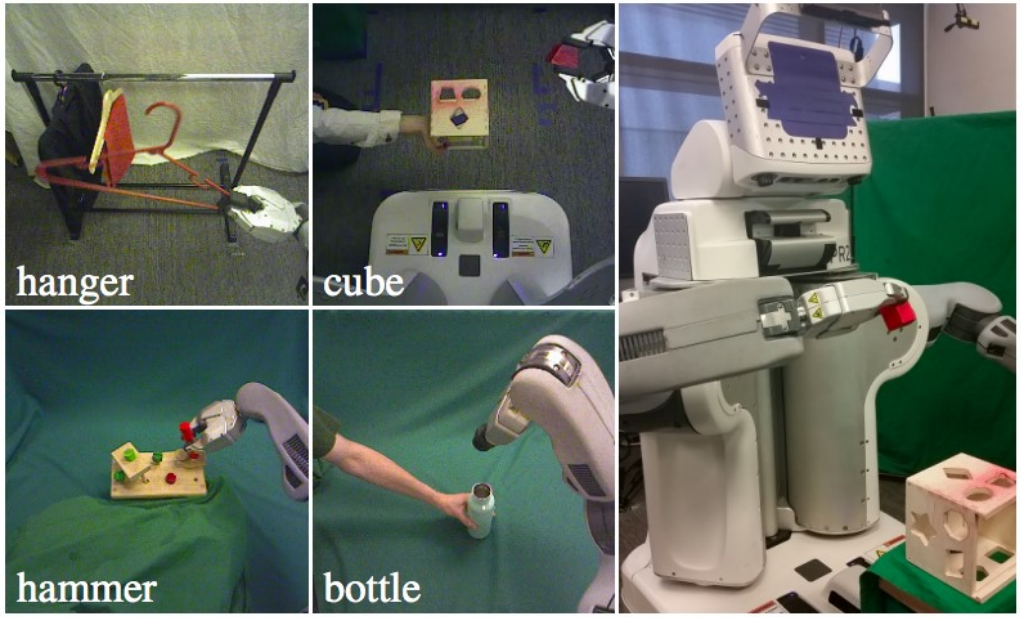


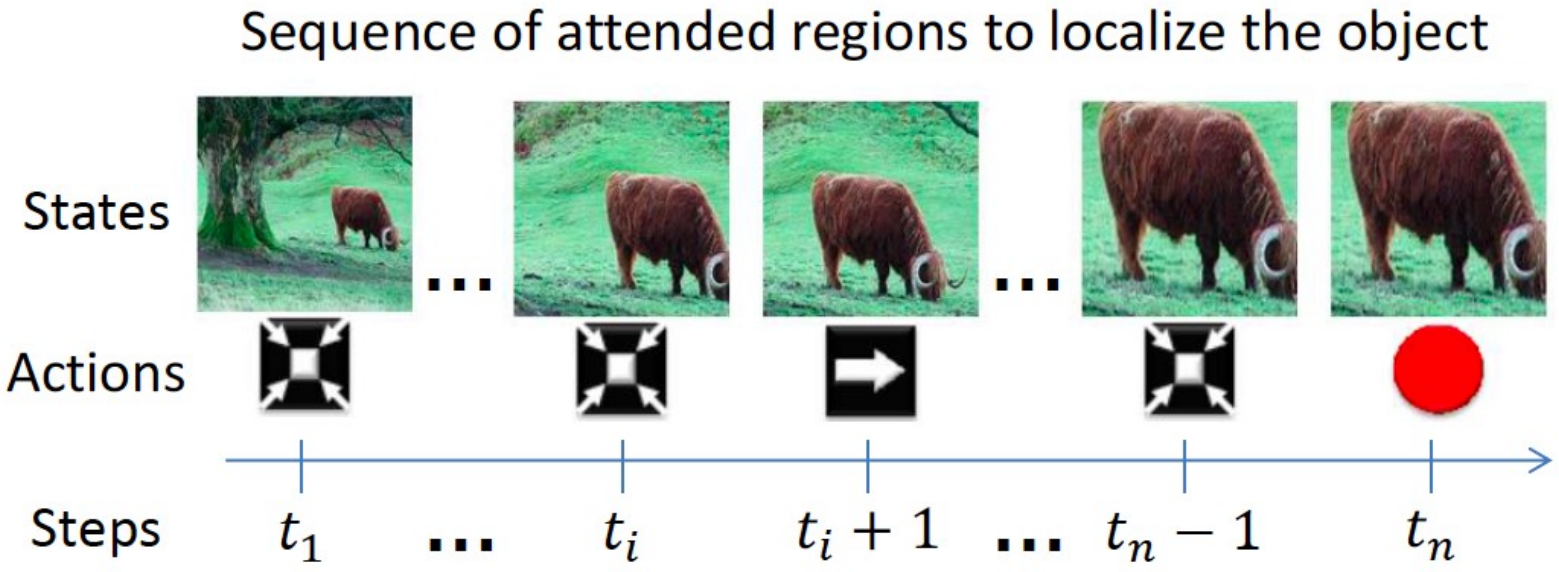
Fig. 1: Our method learns visuomotor policies that directly use camera image observations (left) to set motor torques on a PR2 robot (right).

Video

Sergey Levine et al., Berkeley

Applications of reinforcement learning

- Active object localization with deep reinforcement learning



J. Caicedo and S. Lazebnik, ICCV 2015, to appear

Example: Learning to Walk

Before Learning

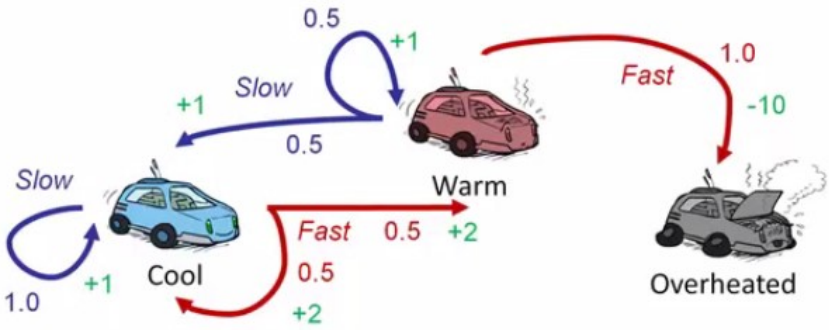
A Learning Trial

After Learning [1K Trials]

[Kohl and Stone, ICRA 2004]

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 $\pi(s)$

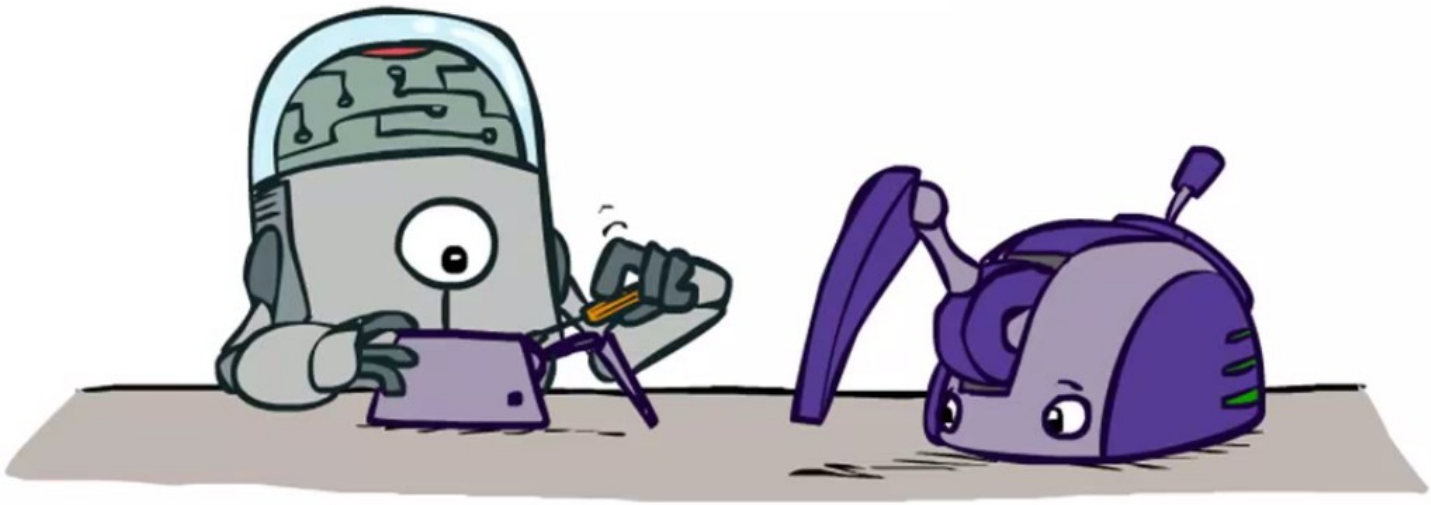


- New twist: don't know T or R
  - I.e. we don't know which states are good or what the actions do
  - Must actually try actions and states out to learn

Reinforcement learning strategies

- **Model-based**
  - Learn the model of the MDP (transition probabilities and rewards) and try to solve the MDP concurrently
- **Model-free**
  - Learn how to act without explicitly learning the transition probabilities $P(s' \mid s, a)$
  - **Q-learning:** learn an action-utility function $Q(s,a)$ that tells us the value of doing action a in state s

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

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 $\hat{T}(s, a, s')$
  - Discover each $\hat{R}(s, a, s')$ when we experience (s, a, s')
- Step 2: Solve the learned MDP
  - For example, use value iteration, as before

Model-based reinforcement learning

- **Basic idea:** try to learn the model of the MDP (transition probabilities and rewards) and learn how to act (solve the MDP) simultaneously
- **Learning the model:**
  - Keep track of how many times state s' follows state s when you take action a and update the transition probability P(s' | s, a) according to the relative frequencies
  - Keep track of the rewards R(s)
- **Learning how to act:**
  - Estimate the utilities U(s) using Bellman's equations
  - Choose the action that maximizes expected future utility:

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

Model-based reinforcement learning

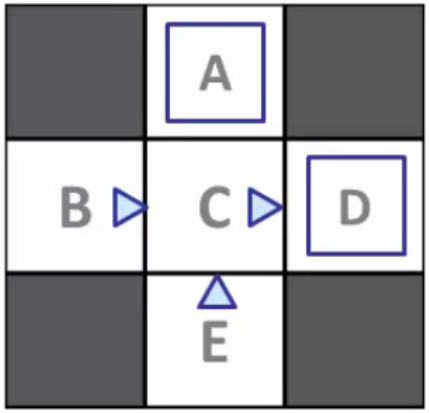
- Learning how to act:
  - Estimate the utilities U(s) using Bellman's equations
  - Choose the action that maximizes expected future utility given the model of the environment we've experienced through our actions so far:

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

- Is there any problem with this "greedy" approach?

Example: Model-Based Learning

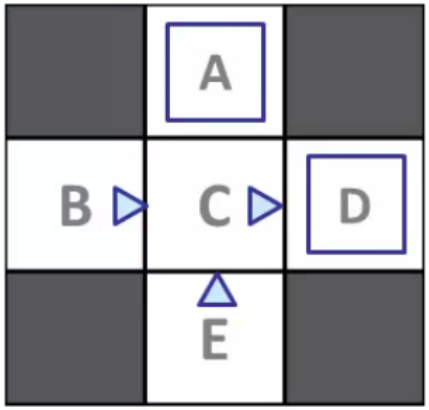
Input Policy π



Assume: $\gamma = 1$

Example: Model-Based Learning

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

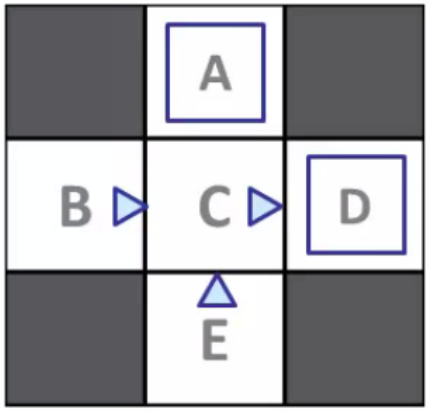
E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

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

Example: Model-Based Learning

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

Learned Model

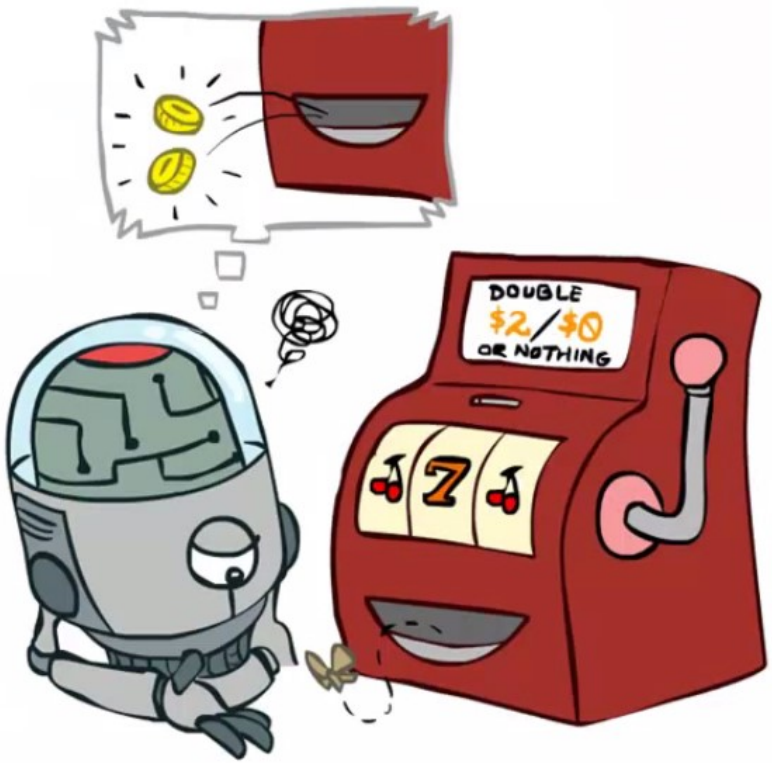
$\hat{T}(s, a, s')$

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

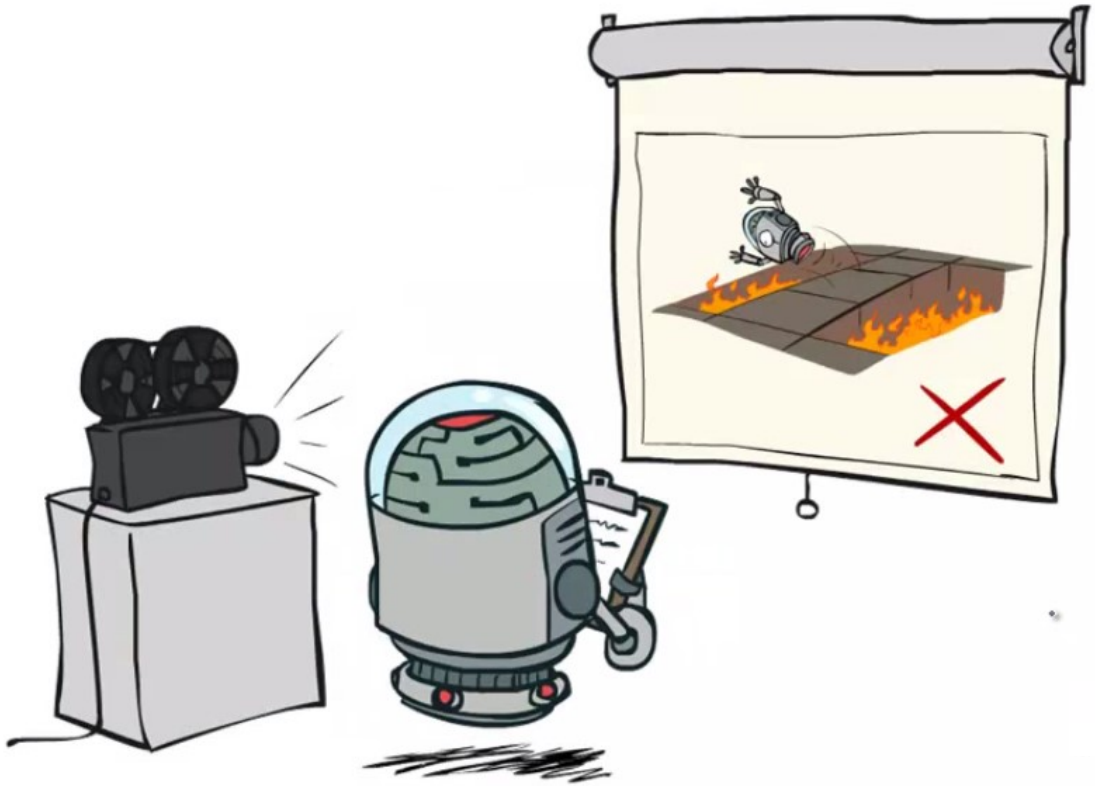
$\hat{R}(s, a, s')$

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

Model-Free Learning

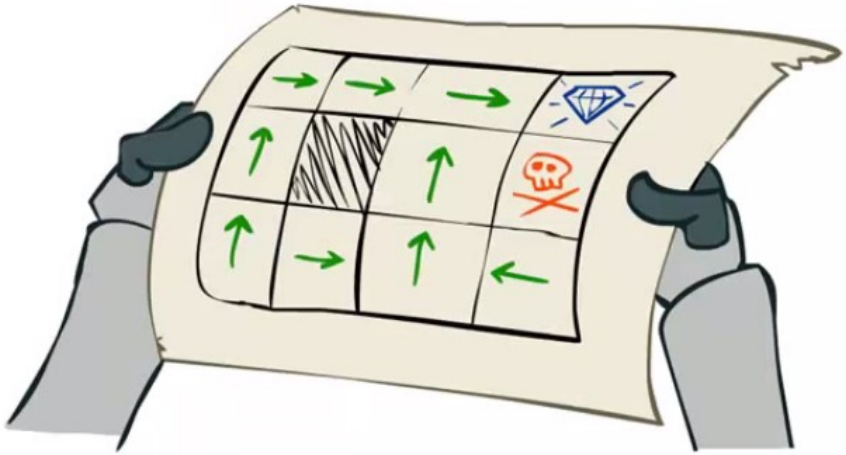


Passive Reinforcement Learning



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



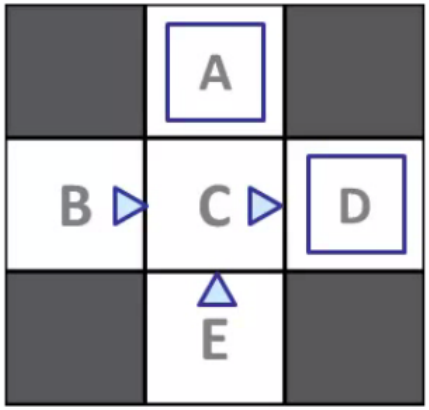
- In this case:
  - Learner is "along for the ride"
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - This is NOT offline planning! You actually take actions in the world.

Direct Evaluation

- Goal: Compute values for each state under 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

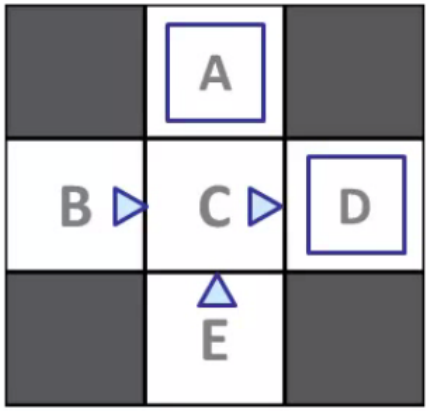
Episode 4

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

Output Values

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

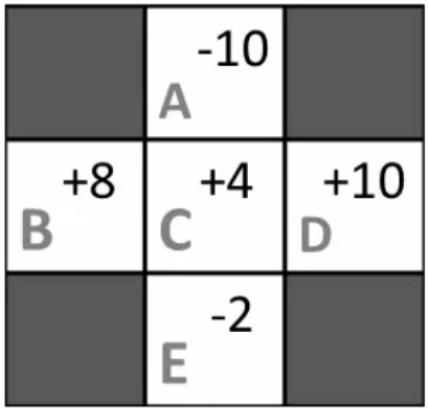
Episode 3

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

Episode 4

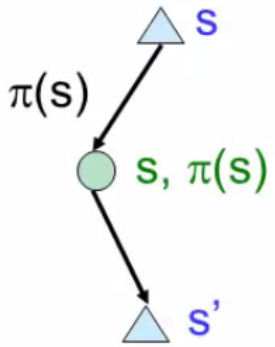
E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values



Temporal Difference Learning

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



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

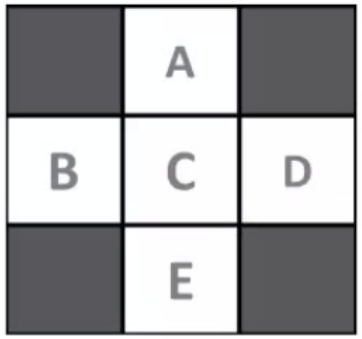
Sample of V(s): $$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

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

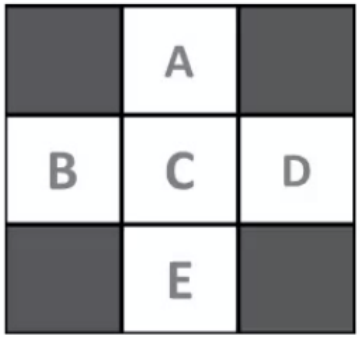
States



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

Example: Temporal Difference Learning

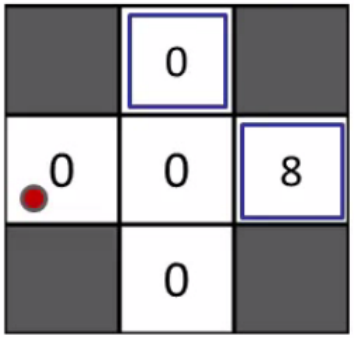
States



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

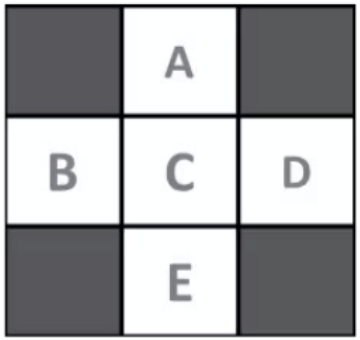
Observed Transitions

B, east, C, -2



Example: Temporal Difference Learning

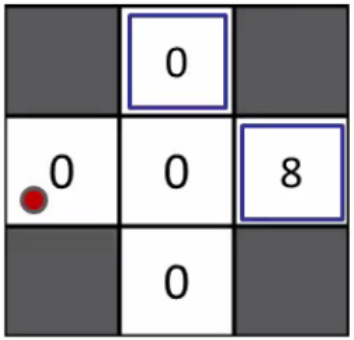
States

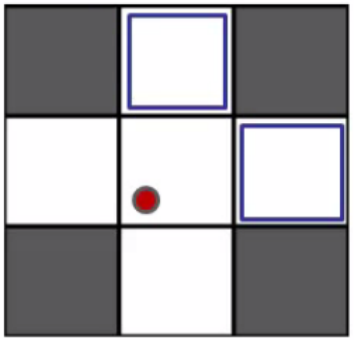


Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Transitions

B, east, C, -2

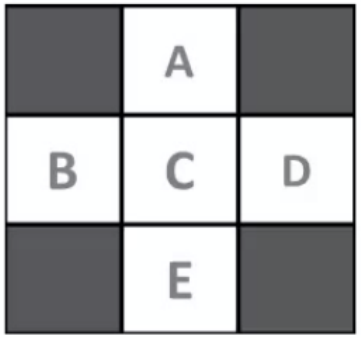




$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

Example: Temporal Difference Learning

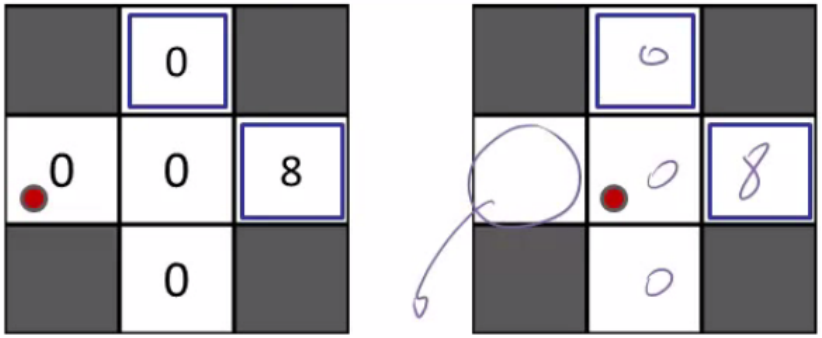
States



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

Observed Transitions

B, east, C, -2

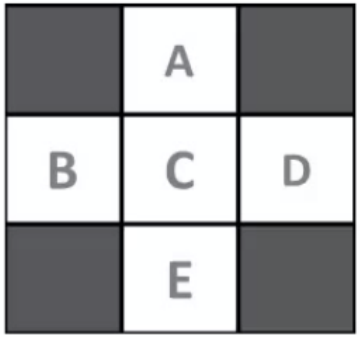


$0.5 \cdot 0 + 0.5[-2 + 0]$

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

Example: Temporal Difference Learning

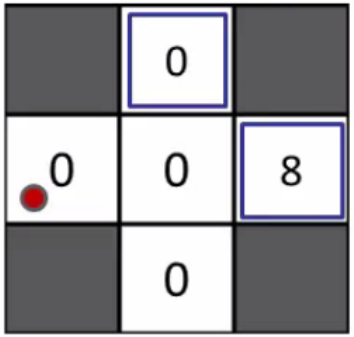
States

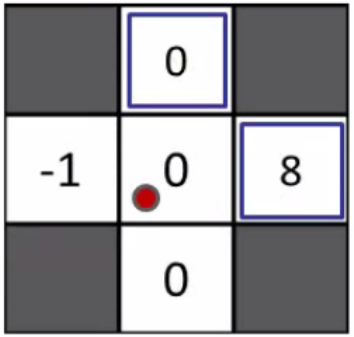


Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Transitions

B, east, C, -2

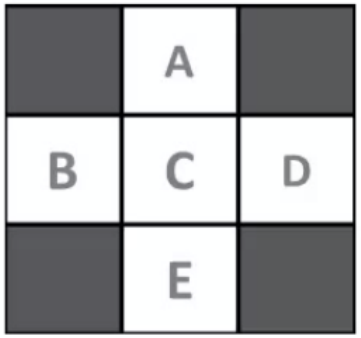




$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

Example: Temporal Difference Learning

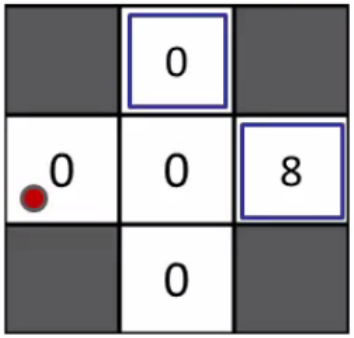
States



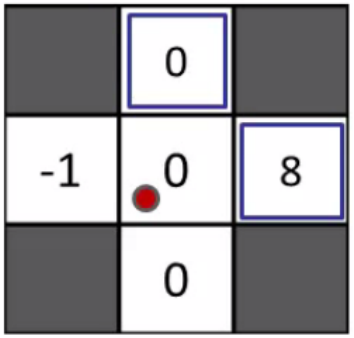
Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Transitions

B, east, C, -2



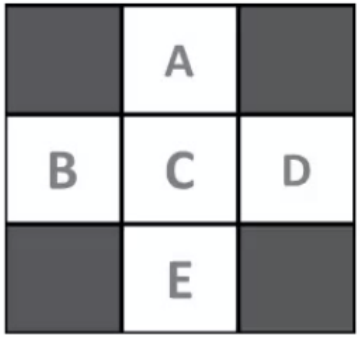
C, east, D, -2



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

Example: Temporal Difference Learning

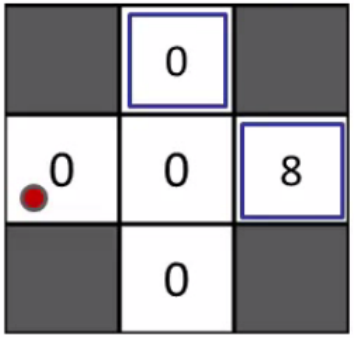
States



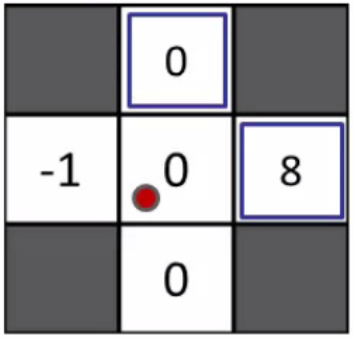
Assume: $\gamma = 1$, $\alpha = 1/2$

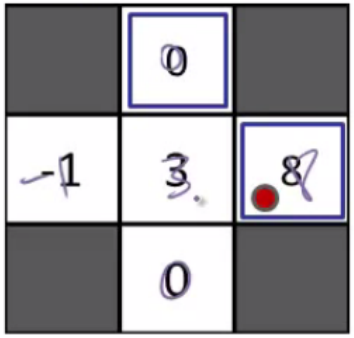
Observed Transitions

B, east, C, -2



C, east, D, -2





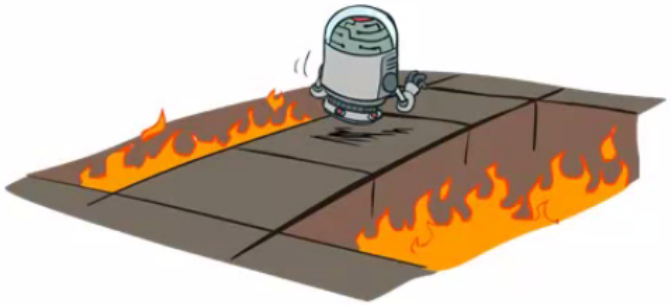
$0.5 \cdot 0 + 0.5(-2 + 1 \cdot 8]$

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

3

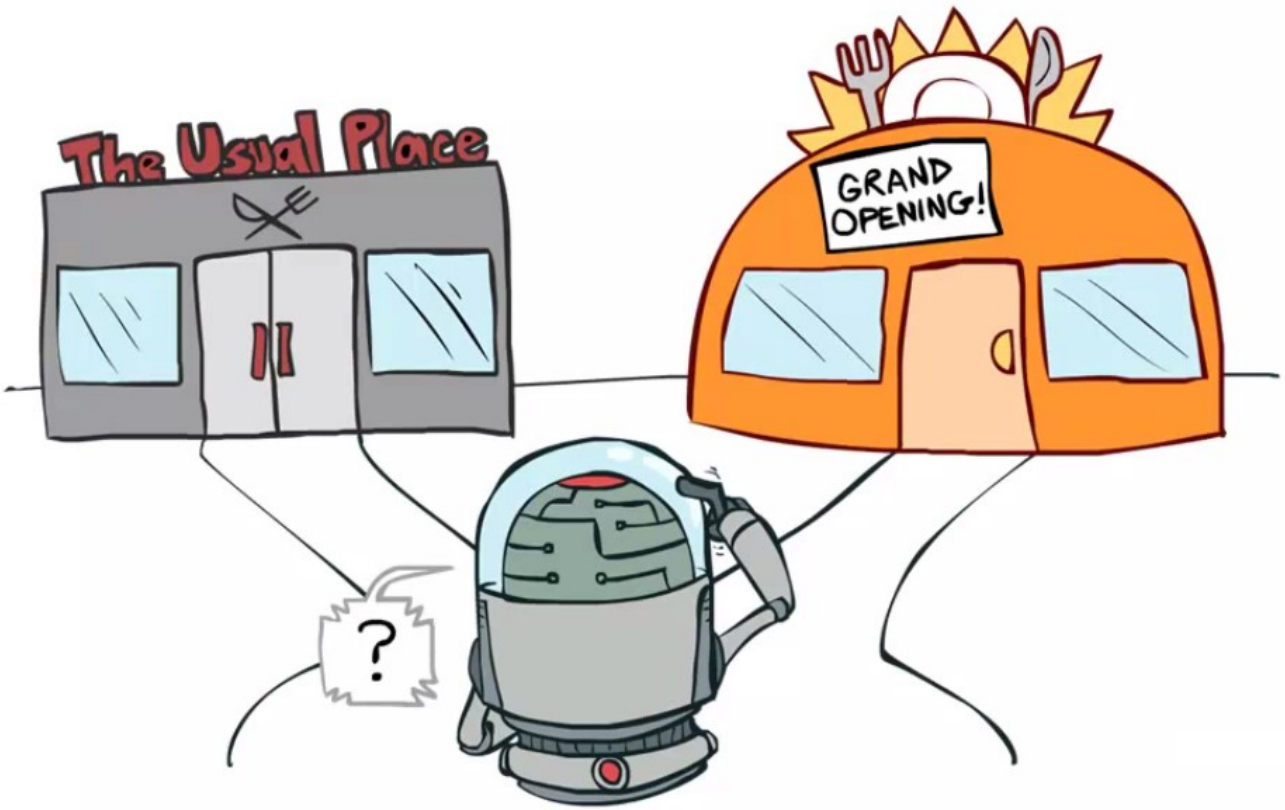
Active Reinforcement Learning

- Full reinforcement learning: optimal policies (like value iteration)
  - You don't know the transitions T(s,a,s')
  - You don't know the rewards R(s,a,s')
  - You choose the actions now
  - Goal: learn the optimal policy / values



- In this case:
  - Learner makes choices!
  - Fundamental tradeoff: exploration vs. exploitation
  - This is NOT offline planning! You actually take actions in the world and find out what happens…

Exploration vs. Exploitation



Exploration vs. exploitation

- **Exploration:** take a new action with unknown consequences
  - Pros:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - Cons:
    - When you're exploring, you're not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - Pros:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff
  - Cons:
    - Might also prevent you from discovering the true optimal strategy

How to Explore?

- Several schemes for forcing exploration
  - Simplest: random actions (ε-greedy)
    - Every time step, flip a coin
    - With (small) probability ε, act randomly
    - With (large) probability 1-ε, act on current policy
  - Problems with random actions?
    - You do eventually explore the space, but keep thrashing around once learning is done
    - One solution: lower ε over time
    - Another solution: exploration functions

Incorporating exploration

- **Idea:** explore more in the beginning, become more and more greedy over time
- Standard ("greedy") selection of optimal action:

$$a = \arg\max_{a' \in A(s)} \sum_{s'} P(s' | s, a') U(s')$$

- Modified strategy:

$$a = \arg\max_{a' \in A(s)} f\left(\sum_{s'} P(s' | s, a') U(s'), N(s, a') \right)$$

exploration function

Number of times we've taken action a' in state s

$$f(u, n) = \begin{cases} R^+ & \text{if } n < N_e \\ u & \text{otherwise} \end{cases}$$

(optimistic reward estimate)

Thank You