CSE 473: Artificial Intelligence

Reinforcement Learning

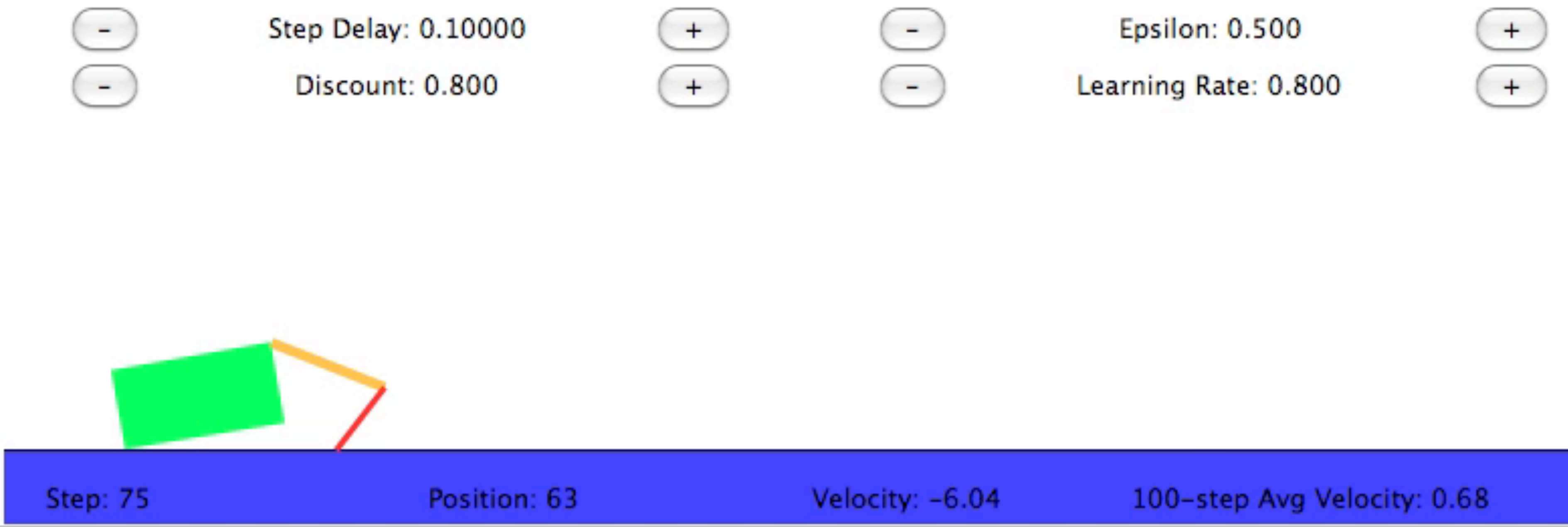
Hanna Hajishirzi

Many slides over the course adapted from either Luke Zettlemoyer, Pieter Abbeel, Dan Klein, Stuart Russell or Andrew Moore

Outline

- Reinforcement Learning
  - Passive Learning
  - TD Updates
  - Q-value iteration
  - Q-learning
  - Linear function approximation

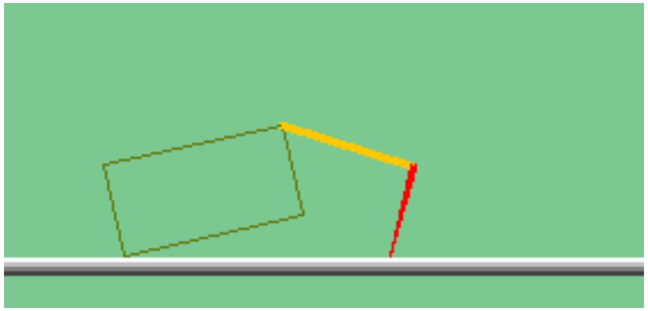
What is it doing?



Reinforcement Learning

- Reinforcement learning:
  - Still have 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

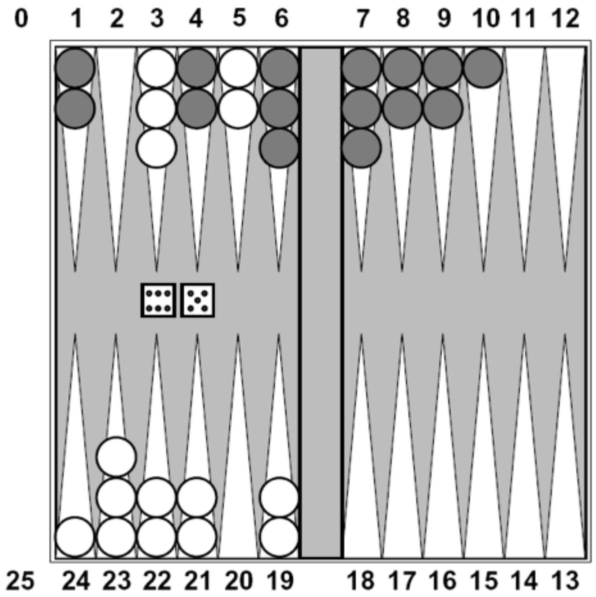


Example: Animal Learning

- RL studied experimentally for more than 60 years in psychology
  - Rewards: food, pain, hunger, drugs, etc.
  - Mechanisms and sophistication debated
- Example: foraging
  - Bees learn near-optimal foraging plan in field of artificial flowers with controlled nectar supplies
  - Bees have a direct neural connection from nectar intake measurement to motor planning area

Example: Backgammon

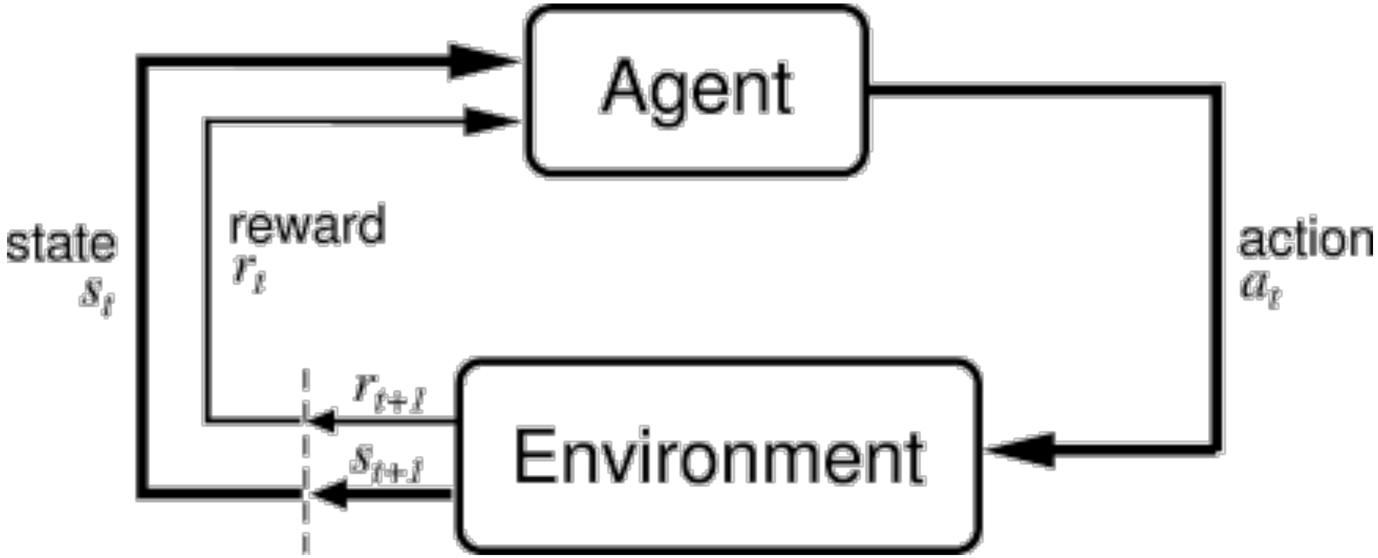
- Reward only for win / loss in terminal states, zero otherwise
- TD-Gammon learns a function approximation to V(s) using a neural network
- Combined with depth 3 search, one of the top 3 players in the world



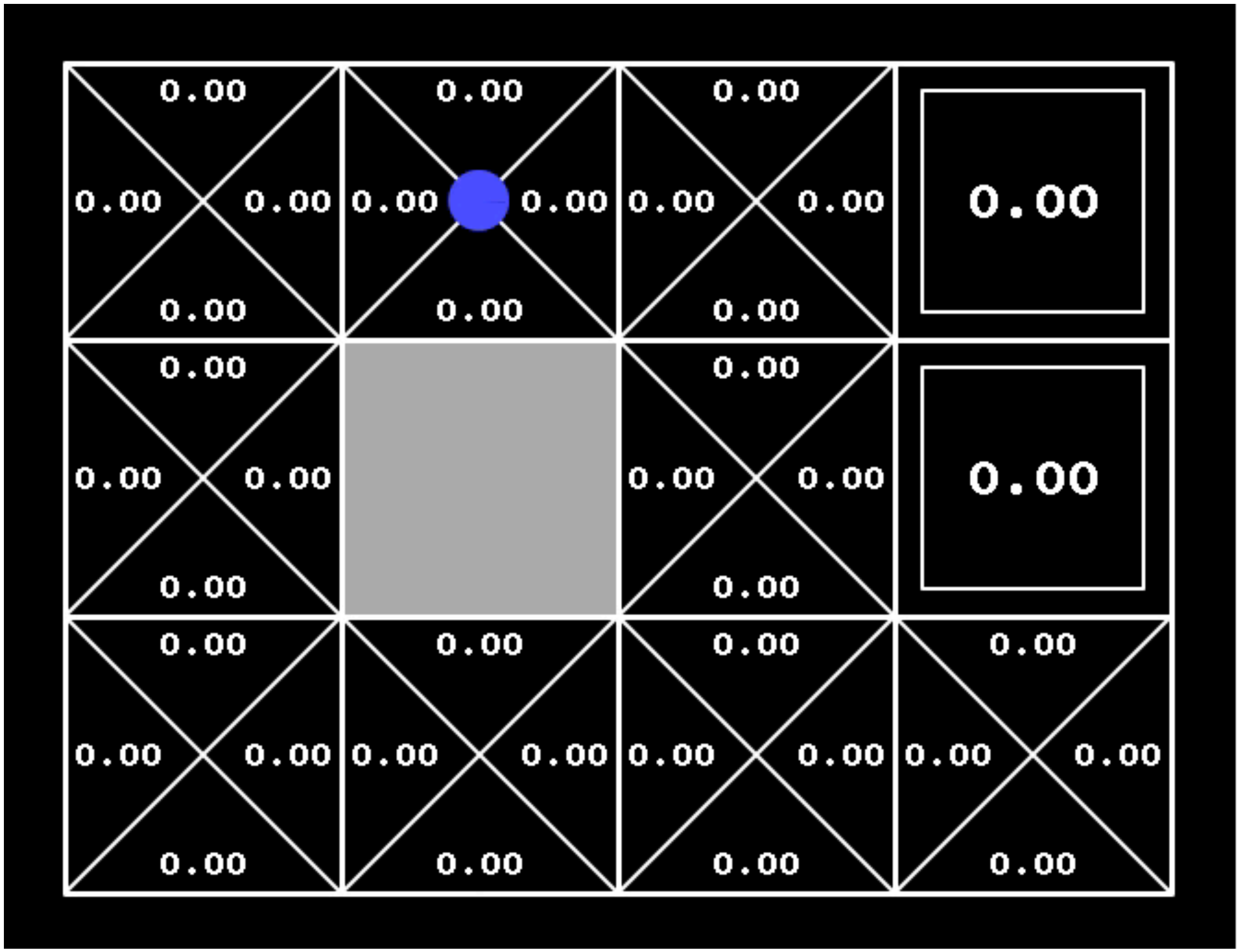
- You could imagine training Pacman this way…
- … but it's tricky! (It's also P3)

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



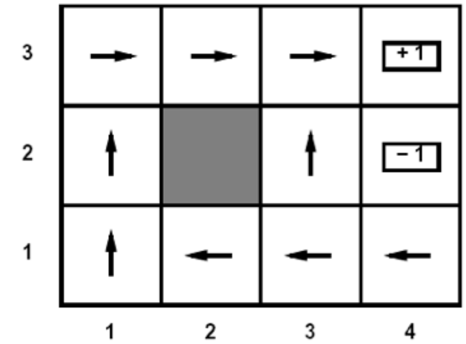
What is the dot doing?



Key Ideas for Learning

- Online vs. Batch
  - Learn while exploring the world, or learn from fixed batch of data
- Active vs. Passive
  - Does the learner actively choose actions to gather experience? or, is a fixed policy provided?
- Model based vs. Model free
  - Do we estimate T(s,a,s') and R(s,a,s'), or just learn values/policy directly

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 $\pi(s)$
  - Goal: learn the state values (and maybe the model)
  - I.e., policy evaluation

- 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!

Detour: Sampling Expectations

- Want to compute an expectation weighted by P(x):

$$E[f(x)] = \sum_x P(x) f(x)$$

- Model-based: estimate P(x) from samples, compute expectation

$$x_i \sim P(x)$$

$$\hat{P}(x) = \text{count}(x)/k$$

$$E[f(x)] \approx \sum_x \hat{P}(x) f(x)$$

- Model-free: estimate expectation directly from samples

$$x_i \sim P(x)$$

$$E[f(x)] \approx \frac{1}{k} \sum_i f(x_i)$$

- Why does this work? Because samples appear with the right frequencies!

Model-Based Learning

- Idea:
  - Learn the model empirically (rather than values)
  - Solve the MDP as if the learned model were correct
- Empirical model learning
  - Simplest case:
    - Count outcomes for each s,a
    - Normalize to give estimate of T(s,a,s')
    - Discover R(s,a,s') the first time we experience (s,a,s')
  - More complex learners are possible (e.g. if we know that all squares have related action outcomes, e.g. "stationary noise")

Example: Model-Based Learning

- Episodes:

(1,1) up -1

(1,2) up -1

(1,2) up -1

(1,3) right -1

(2,3) right -1

(3,3) right -1

(3,2) up -1

(3,3) right -1

(4,3) exit +100

(done)

(1,1) up -1

(1,2) up -1

(1,3) right -1

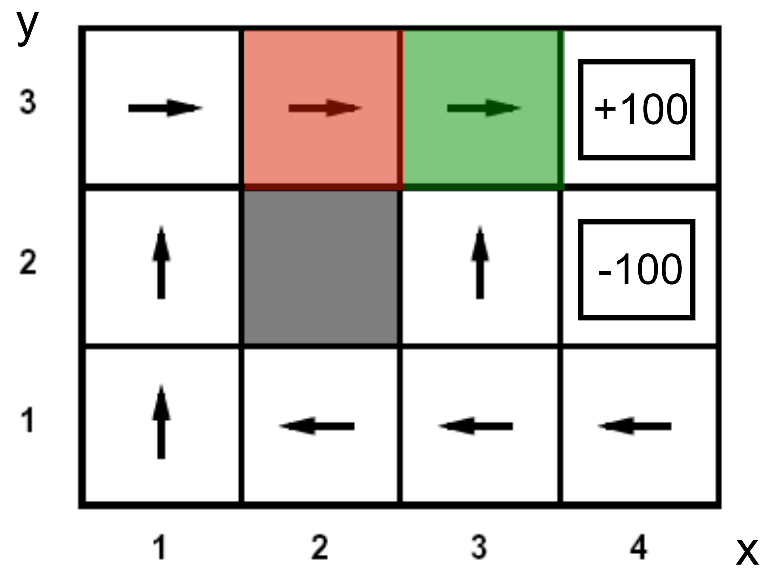
(2,3) right -1

(3,3) right -1

(3,2) up -1

(4,2) exit -100

(done)



$\gamma = 1$

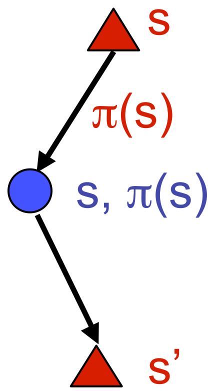
T(<3,3>, right, <4,3>) = 1 / 3

T(<2,3>, right, <3,3>) = 2 / 2

Model-free Learning

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

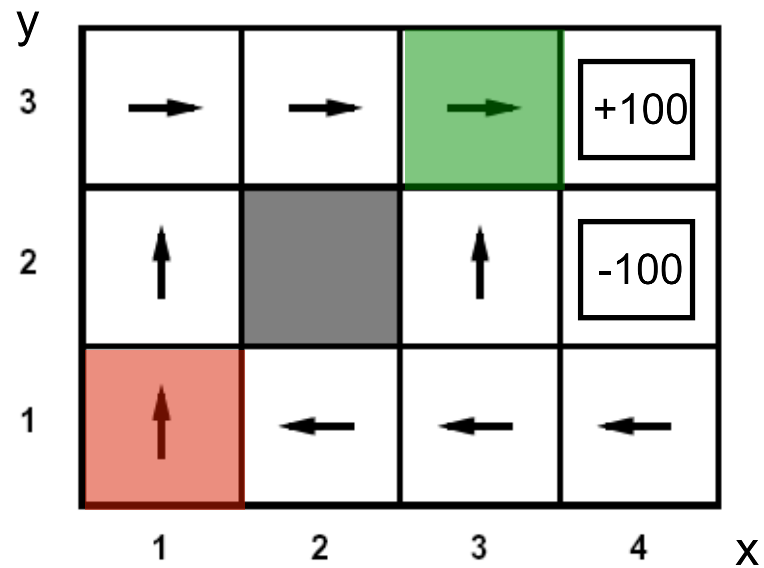
- Big idea: why bother learning T?
- Question: how can we compute V if we don't know T?
  - Use direct estimation to sample complete trials, average rewards at end
  - Use sampling to approximate the Bellman updates, compute new values during each learning step



Simple Case: Direct Estimation

- Average the total reward for every trial that visits a state:

(1,1) up -1	(1,1) up -1
(1,2) up -1	(1,2) up -1
(1,2) up -1	(1,3) right -1
(1,3) right -1	(2,3) right -1
(2,3) right -1	(3,3) right -1
(3,3) right -1	(3,2) up -1
(3,2) up -1	(4,2) exit -100
(3,3) right -1	(done)
(4,3) exit +100	
(done)	



$\gamma = 1$, $R = -1$

V(1,1) ~ (92 + -106) / 2 = -7

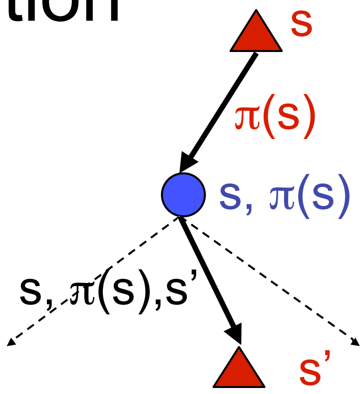
V(3,3) ~ (99 + 97 + -102) / 3 = 31.3

Problems with Direct Evaluation

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

Towards Better Model-free Learning

Review: Model-Based 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

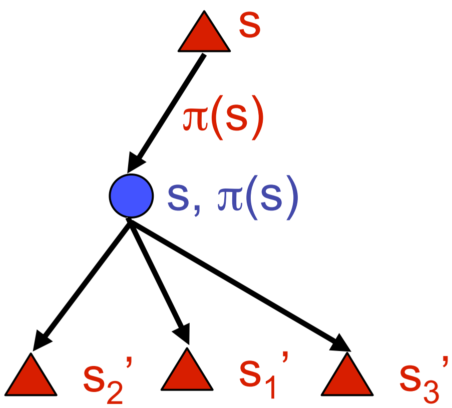
$$V_0^\pi(s) = 0$$

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

Sample Avg to Replace Expectation?

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

- Who needs T and R? Approximate the expectation with samples (drawn from T!)



$$sample_1 = R(s,\pi(s),s_1') + \gamma V_i^{\pi}(s_1')$$

$$sample_2 = R(s,\pi(s),s_2') + \gamma V_i^{\pi}(s_2')$$

$$\ldots$$

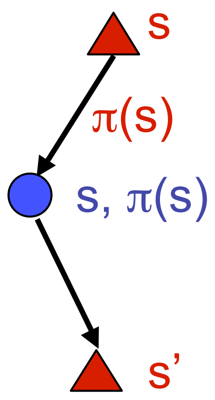
$$sample_k = R(s,\pi(s),s_k') + \gamma V_i^{\pi}(s_k')$$

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

Temporal Difference Learning

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

- Big idea: why bother learning T?
  - Update V each time we experience a transition
- Temporal difference learning (TD)
  - Policy still fixed!
  - Move values toward value of whatever successor occurs: running average!



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

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

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

Detour: Exp. Moving Average

- Exponential moving average
  - Makes recent samples more important

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

  - Forgets about the past (distant past values were wrong anyway)
  - Easy to compute from the running average

$$\bar{x}_n = (1-\alpha)\cdot \bar{x}_{n-1} + \alpha \cdot x_n$$

- Decreasing learning rate can give converging averages

TD Policy Evaluation

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

(1,1) up -1

(1,2) up -1

(1,2) up -1

(1,3) right -1

(2,3) right -1

(3,3) right -1

(3,2) up -1

(3,3) right -1

(4,3) exit +100

(done)

(1,1) up -1

(1,2) up -1

(1,3) right -1

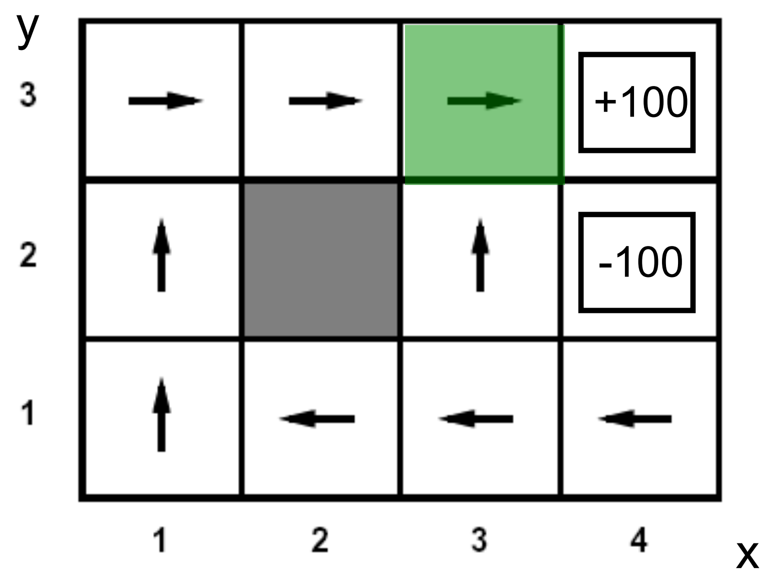
(2,3) right -1

(3,3) right -1

(3,2) up -1

(4,2) exit -100

(done)



Updates for V(<3,3>):

V(<3,3>) = 0.5*0 + 0.5*[-1 + 1*0] = -0.5

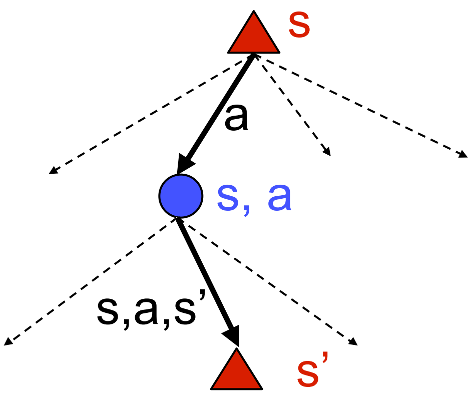
V(<3,3>) = 0.5*-0.5 + 0.5*[-1+1*100] = 49.475

V(<3,3>) = 0.5*49.475 + 0.5*[-1 + 1*-0.75]

Take $\gamma = 1$, $\alpha = 0.5$, V_0(<4,3>)=100, V_0(<4,2>)=-100, $V_0 = 0$ otherwise

Problems with TD Value Learning

- TD value leaning is model-free for policy evaluation (passive learning)
- However, if we want to turn our value estimates into a policy, we're sunk:



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

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

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