CMU MDPs 15-381/781

Emma Brunskill (THIS TIME)
Ariel Procaccia

Discuss and Report Back: Does Initialization Impact Final Value?

Value Iteration Algorithm

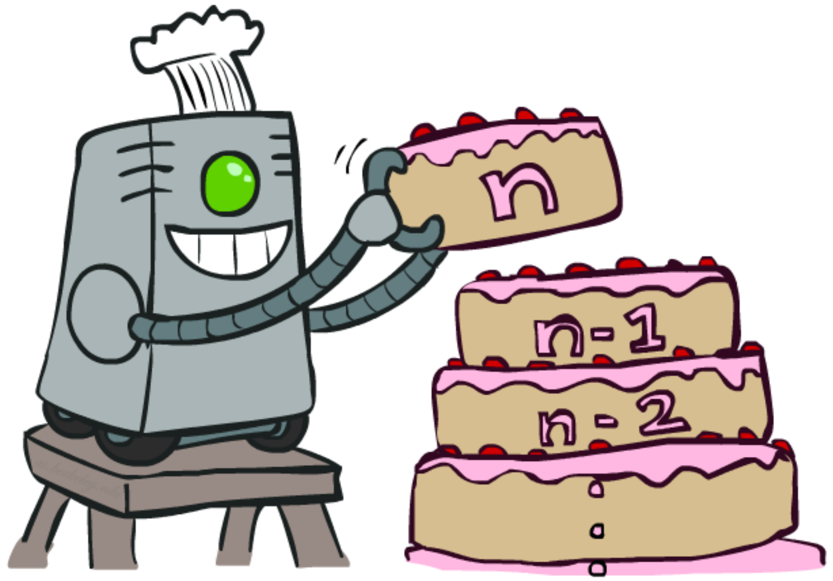
1. Init $V_0(s_i)$ for all states s_i
2. k=1
3. While k < desired horizon or (if infinite horizon) values have converged
   - For all s,

$$V_k(s_i) = \max_a \left(\sum_{s_j \in S} p(s_j \mid s_i, a) \left[R(s,a,s') + \gamma V_{k-1}(s_j) \right] \right)$$

Carnegie Mellon University

Value Iteration in Infinite Horizon

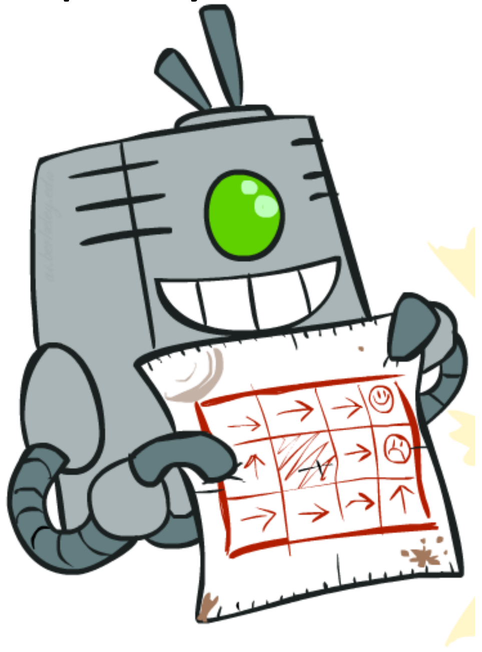
- Always have optimal values as if had only t more decisions to make
- Extracting optimal policy for t-th step yields optimal action should take, if have t more steps to act
  - But not for (t-1), (t-2)… steps
- Before convergence, these are approximations (because actually get to act forever!)
  - After convergence, value is always the same if do another update, and so is the policy



Drawing by Ketrina Yim

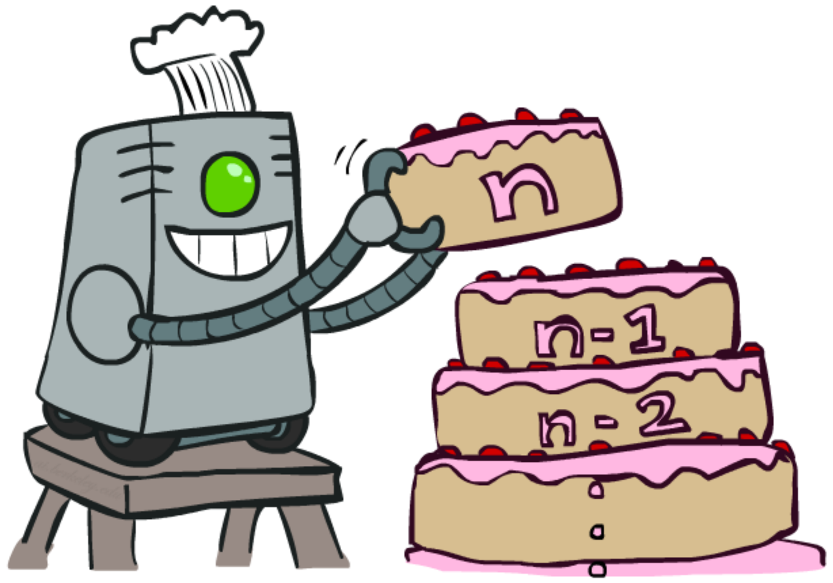
Policy Iteration

Maintain value of following a particular policy forever



Value Iteration

Maintain optimal values if have n more actions



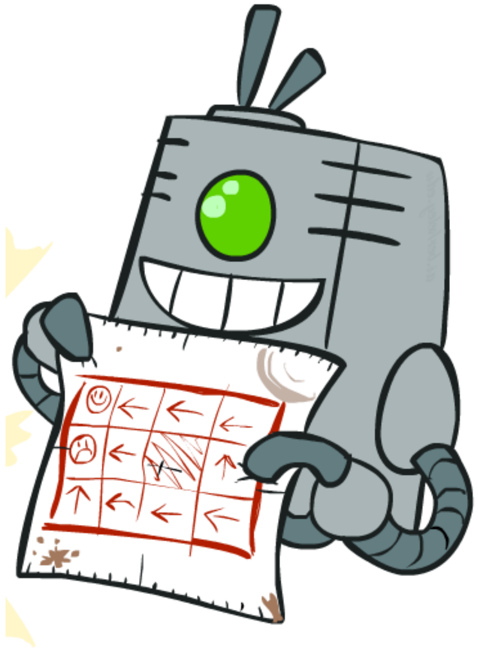
Drawings by Ketrina Yim

Carnegie Mellon University

Policy Iteration for Infinite Horizon

- Instead of maintaining optimal value if have t steps left…

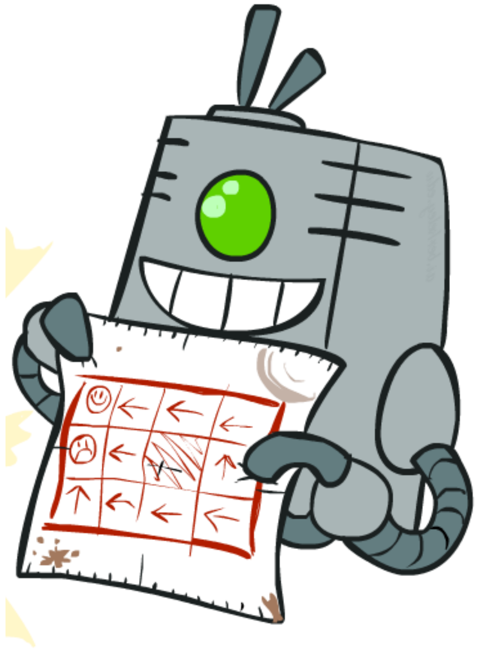
1. Calculate exact value of acting in infinite horizon for a particular policy
2. Then try to improve the policy
3. Repeat 1 & 2 until policy doesn't change


Drawing by Ketrina Yim

Policy Iteration for Infinite Horizon

- Instead of maintaining optimal value if have t steps left…

1. Calculate exact value of acting in infinite horizon for a particular policy **(evaluation)**
2. Then try to improve the policy **(improvement)**
3. Repeat 1 & 2 until policy doesn't change


Drawing by Ketrina Yim

Review: Value of a Policy

- Expected immediate reward for taking action prescribed by policy
- And expected future reward get after taking policy from that state and following π

$$V^{\pi}(s) = \sum_{s' \in S} p(s' \mid s, \pi(s)) \left[R(s, \pi(s), s') + \gamma V^{\pi}(s') \right]$$

Carnegie Mellon University

Policy Evaluation

- Goal: compute $V^{\pi}(s)$ for all s

Idea 1: Use Small Variant of Value Iteration

- Initialize $V_0(s)$ to 0 for all s
- For k=1… convergence

$$V_k^{\pi}(s_i) = \sum_{s_j \in S} p(s_j \mid s_i, \pi(s_i)) \left[R(s, \pi(s), s') + \gamma V_{k-1}^{\pi}(s_j) \right]$$

Carnegie Mellon University

Idea 2: No max in eqn so linear set of equations… Analytic Solution!

$$V^\pi(s) = \sum_{s' \in S} p(s' \mid s, \pi(s)) \left[R(s, \pi(s), s') + \gamma V^\pi(s') \right]$$

Let T^π be a S x S matrix where the (i,j) entry is:

$$T^\pi(s_i, s_j) = p(s_j \mid s_i, \pi(s_i))$$

$$\vec{V} = T^\pi \vec{R} + \gamma T^\pi \vec{V}$$

$$\vec{V} - \gamma T^\pi \vec{V} = T^\pi \vec{R}$$

$$\vec{V} = (1 - \gamma T^\pi)^{-1} T^\pi \vec{R}$$

Requires taking an inverse of a S by S matrix
O(S³)

Carnegie Mellon University

Policy Improvement

- Have $V^{\pi}(s)$ for all s
- Want a better policy
- Idea:
  - Find the state-action Q value of doing an action followed by following π forever, for each state
  - Then take argmax of Qs

Carnegie Mellon University

Policy Improvement

- Have $V^{\pi}(s)$ for all s
- First compute

$$Q^{\pi}(s,a) = \sum_{s' \in S} p(s' \mid s,a)\left[R(s,a,s') + \gamma V^{\pi}(s')\right]$$

- Then extract new policy.
  For each s,

$$\pi'(s) = \arg\max_a Q^{\pi}(s,a)$$

Carnegie Mellon University

Delving Deeper Into Improvement Step

$$Q^{\pi}(s,a) = \sum_{s' \in S} p(s' \mid s,a)\left[R(s,a,s') + \gamma V^{\pi}(s')\right]$$

$$\max_a Q^{\pi}(s,a) \geq V^{\pi}(s)$$

$$\pi'(s) = \arg\max_a Q^{\pi}(s,a)$$

- So if, starting at any state, we followed π'(s), transitioned to a new state s', and then followed π forever, our expected sum of rewards would be at least as good as if we had always followed π
- But we're not doing that, we're getting a new policy and proposing always following that policy π'…

Monotonic Improvement in Policy

- For any two value functions V1 and V2, let V1 >= V2 → for all states s, V1(s) >= V2(s)
- Proposition: $V^{\pi'} >= V^{\pi}$ with strict inequality if π is suboptimal (where π' is the new policy we get from doing policy improvement)

Proof

$$V^{\pi}(s) \leq \max_a Q^{\pi}(s,a)$$

$$= \sum_{s' \in S} p(s' \mid s, \pi'(s)) \left[R(s, \pi'(s), s') + \gamma V^{\pi}(s') \right]$$

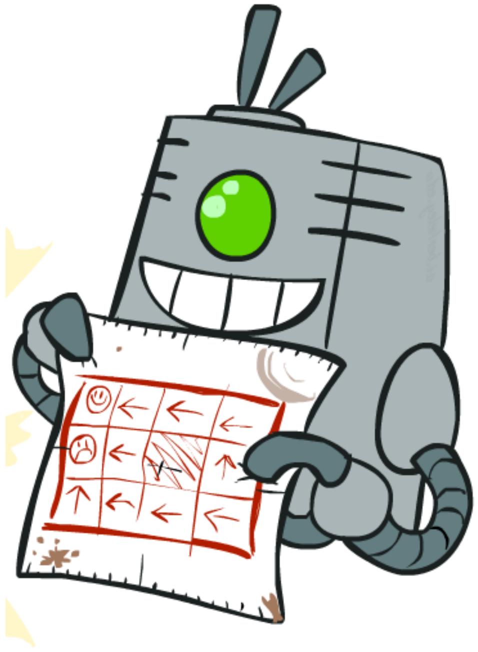
$$\leq \sum_{s' \in S} p(s' \mid s, \pi'(s)) \left[R(s, \pi'(s), s') + \gamma \max_{a'} Q^{\pi}(s', a') \right]$$

$$= \sum_{s' \in S} p(s' \mid s, \pi'(s)) \begin{bmatrix} R(s, \pi'(s), s') + \\ \gamma \sum_{s' \in S} p(s'' \mid s', \pi'(s'))(R(s', \pi'(s'), s'' + \gamma V^{\pi}(s'')) \end{bmatrix}$$

$$\ldots \leq V^{\pi'}(s)$$

Policy Iteration for Infinite Horizon

1. **Policy Evaluation:**
   Calculate exact value of acting in infinite horizon for a particular policy
2. **Policy Improvement**
3. Repeat 1 & 2 until policy doesn't change



Drawing by Ketrina Yim

Carnegie Mellon University

Discuss and Report Back: If Policy Doesn't Change ($\pi'(s) = \pi(s)$ for all s), Can It Ever Change Again in More Iterations?

- Given $V^\pi(s)$ for all s

$$Q^\pi(s,a) = \sum_{s' \in S} p(s' \mid s,a)\left[R(s,a,s') + \gamma V^\pi(s')\right]$$

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

Carnegie Mellon University

Discuss and Report Back: If Policy Doesn't Change (π'(s) =π(s) for all s), Can It Ever Change Again in More Iterations? No

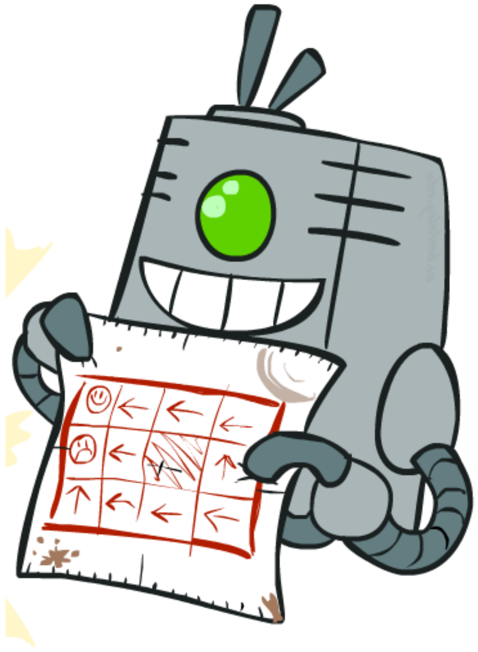
- Given $V^\pi(s)$ for all s

$$Q^\pi(s,a) = \sum_{s' \in S} p(s' \mid s,a)\left[R(s,a,s') + \gamma V^\pi(s')\right]$$

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

Policy Iteration for Infinite Horizon

1. **Policy Evaluation:**
   Calculate exact value of acting in infinite horizon for a particular policy
2. **Policy Improvement**
3. Repeat 1 & 2 until policy doesn't change



Drawing by Ketrina Yim

- Discuss and Report Back Policy Iteration is Guaranteed to Converge. Think About Why, and How Many Iterations Could it Take (How many policies could we have to sort through)?

Policy Iteration is Guaranteed to Converge. Think About Why, and How Many Iterations Could it Take?

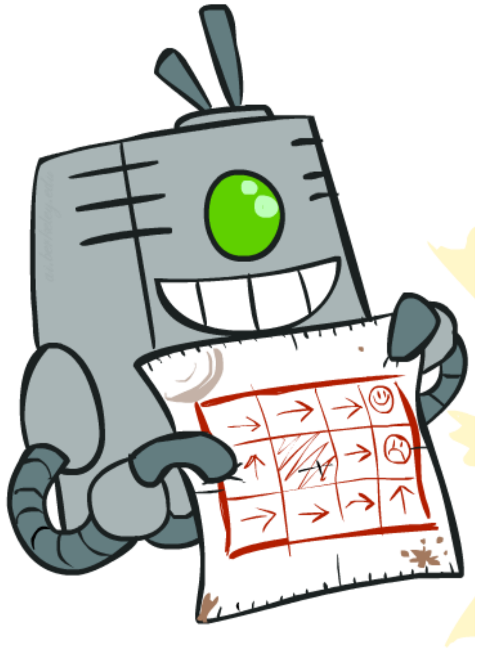
A^S

Carnegie Mellon University

Policy Iteration

Maintain value of policy

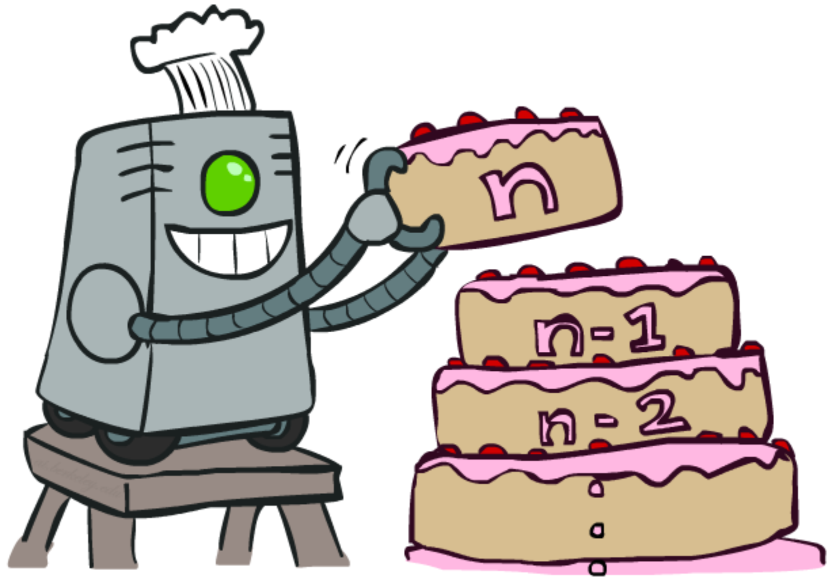
Improve policy



Value Iteration

Keep optimal value for

finite steps, increase steps



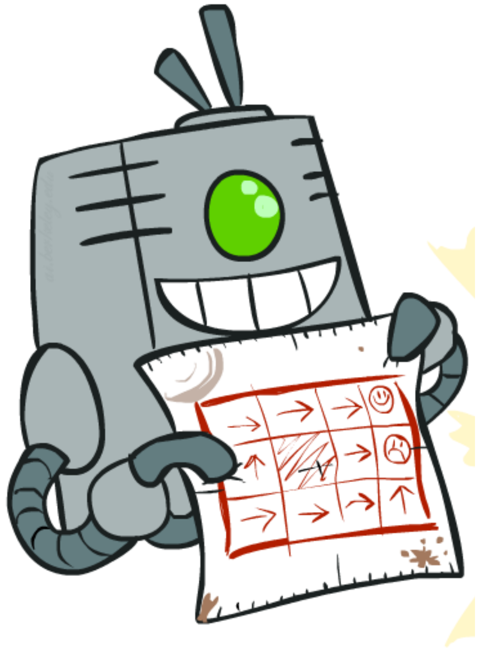
Drawings by Ketrina Yir

Carnegie Mellon University

Policy Iteration

Fewer Iterations

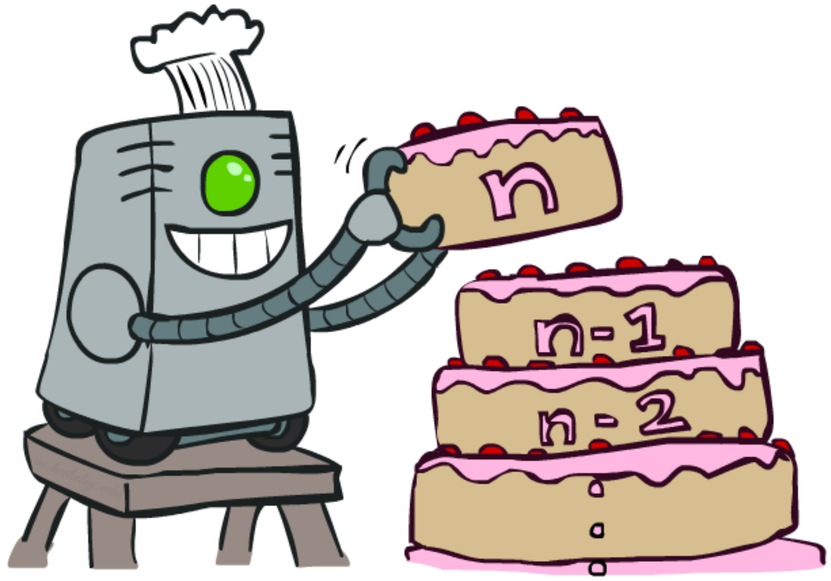
More expensive per iteration



Value Iteration

More iterations

Cheaper per iteration



Drawings by Ketrina Yim

MDPs: What You Should Know

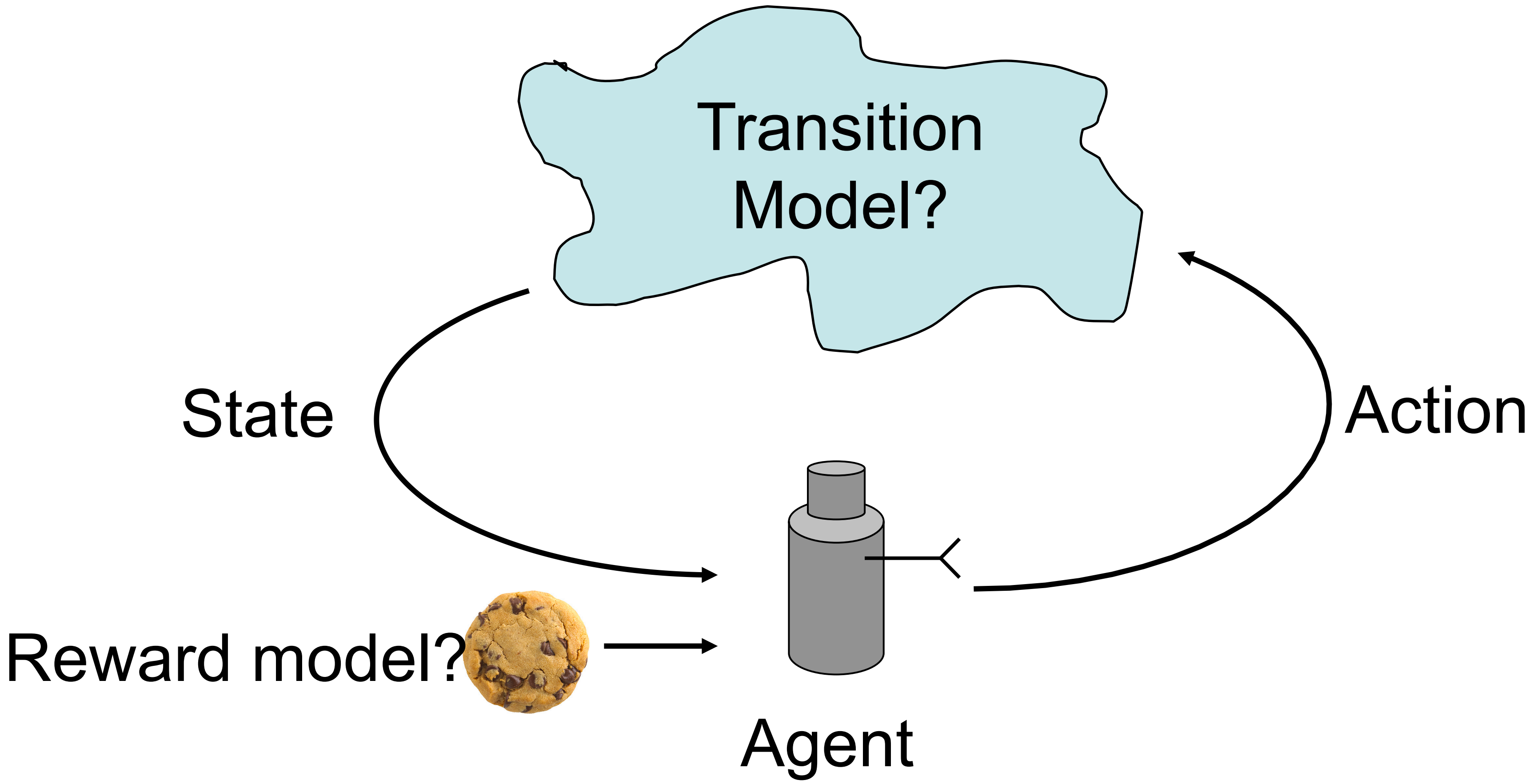
- Definition
- How to define for a problem
- Value iteration and policy iteration
    - How to implement
    - Convergence guarantees
    - Computational complexity

Reasoning Under Uncertainty

Learn model of outcomes	Multi-armed bandits	**Reinforcement Learning**
Given model of stochastic outcomes	Decision theory	Markov Decision Processes
	Actions Don't Change State of the World	**Actions Change State of the World**

Carnegie Mellon University

Reinforcement Learning

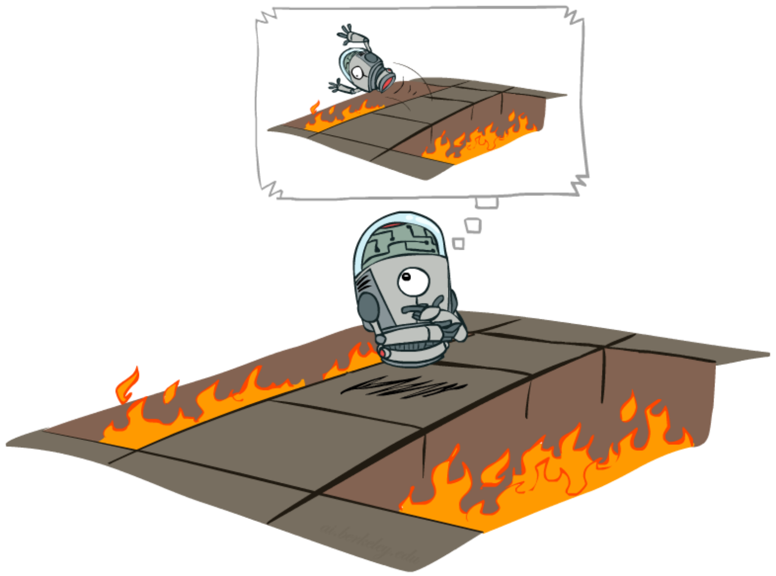


Goal: Maximize expected sum of future rewards

Toddler Robot

https://www.youtube.com/watch?v=goqWX7bC-ZY&list=PL28ekMsVlKa1mKRif05Uxn-iuP2Ms6qDL

MDP Planning vs Reinforcement Learning



Don't have a simulator! Have to actually learn what happens if take an action in a state

- Before figuring out how to act, let’s first just try to figure out how good a particular strategy is

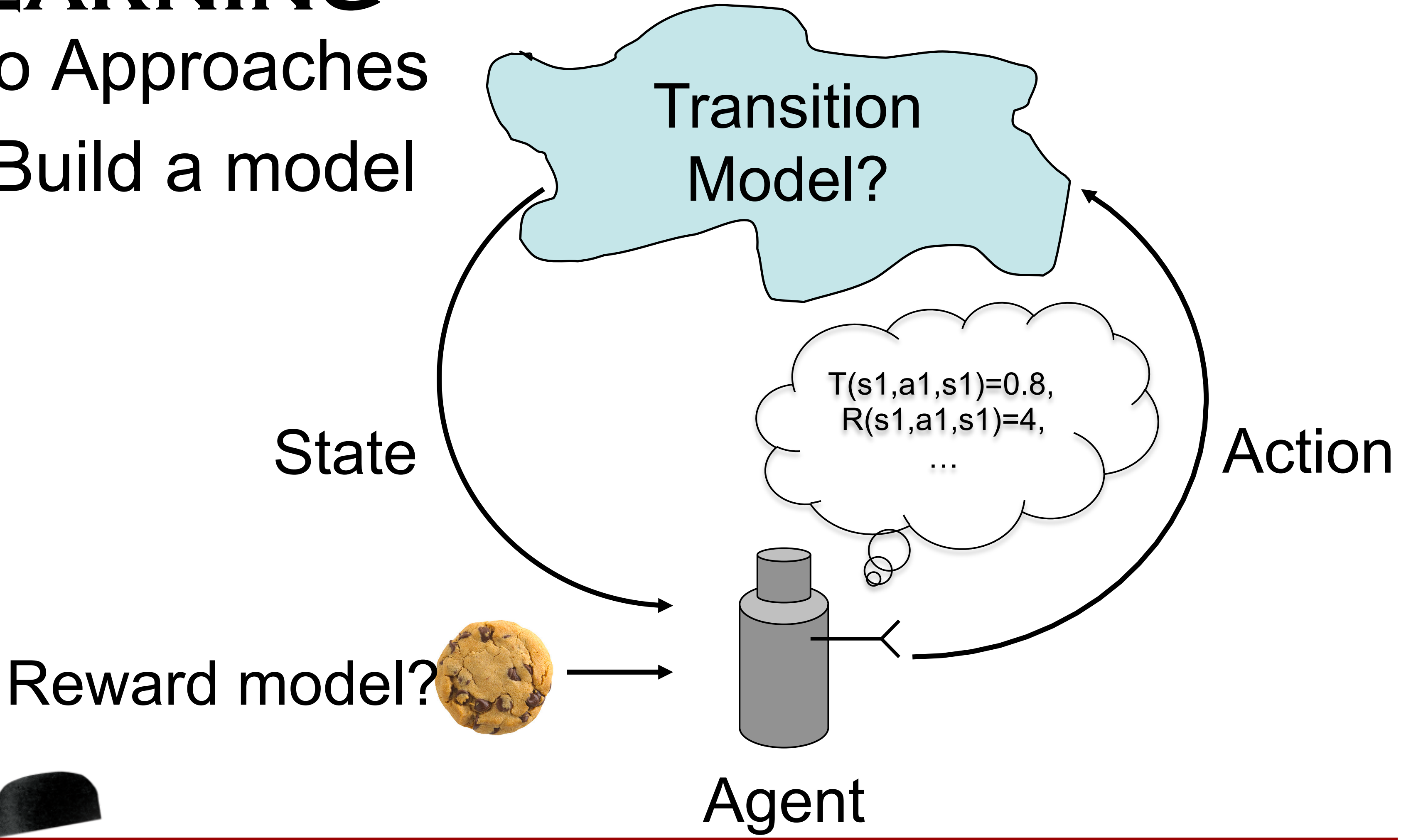
Policy Evaluation given MDP's model parameters

Learning a Policy's Value (Passive Reinforcement Learning)

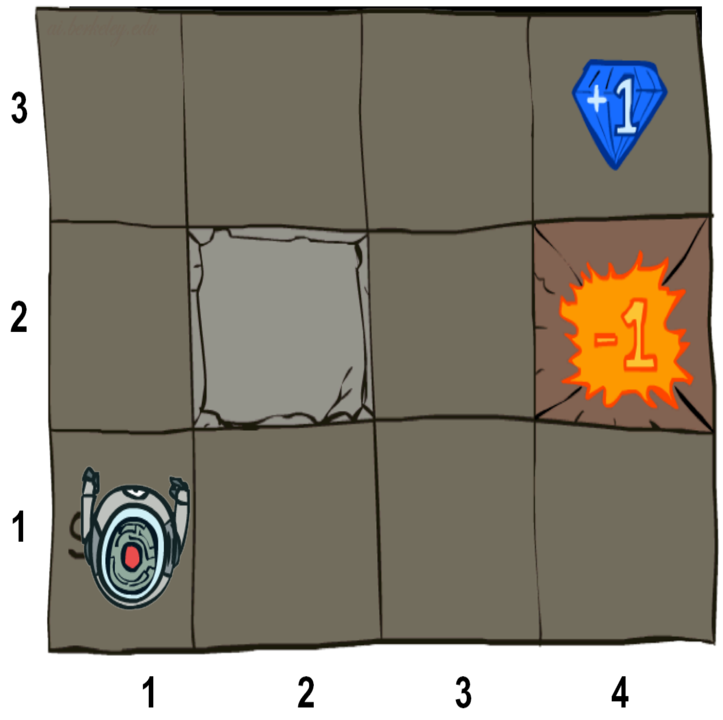
Passive Reinforcement Learning

Two Approaches

1. Build a model

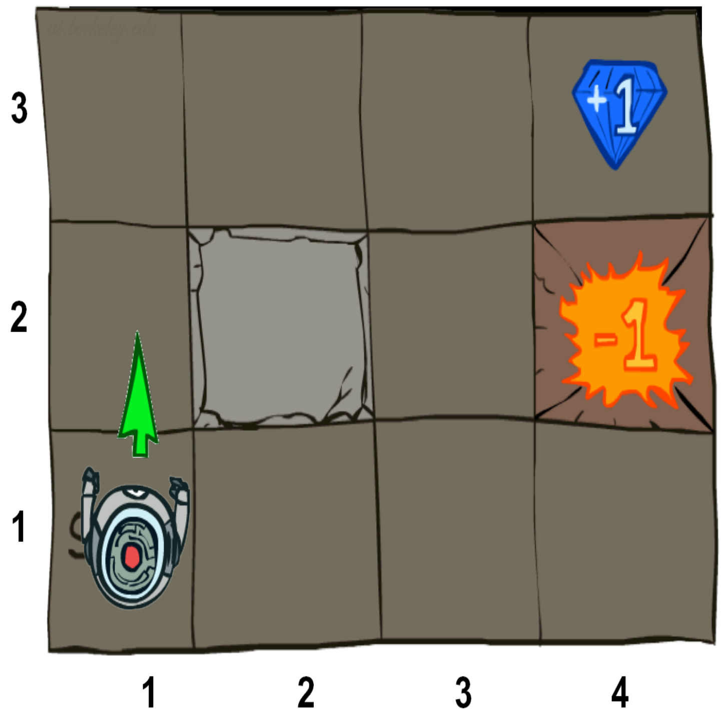


Start at (1,1)



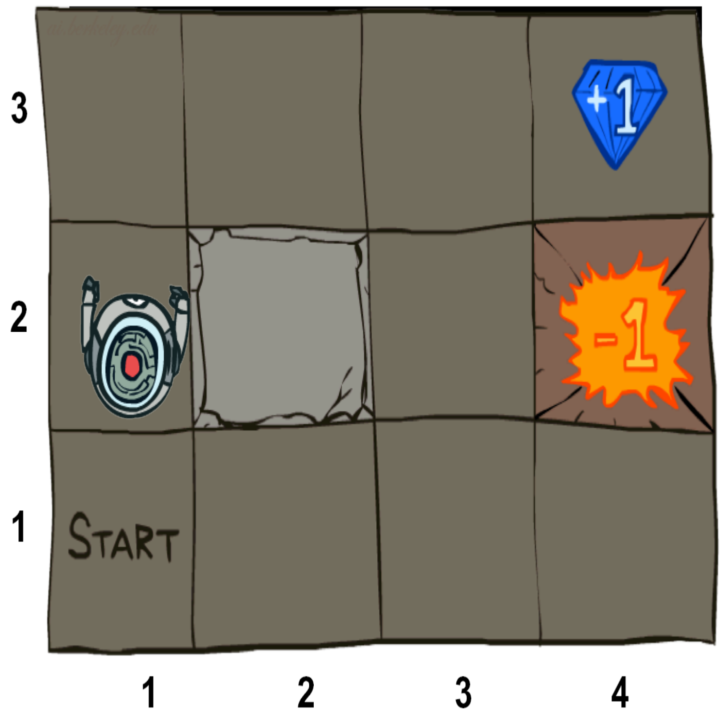
Start at (1,1)

s=(1,1) action= tup



Start at (1,1)

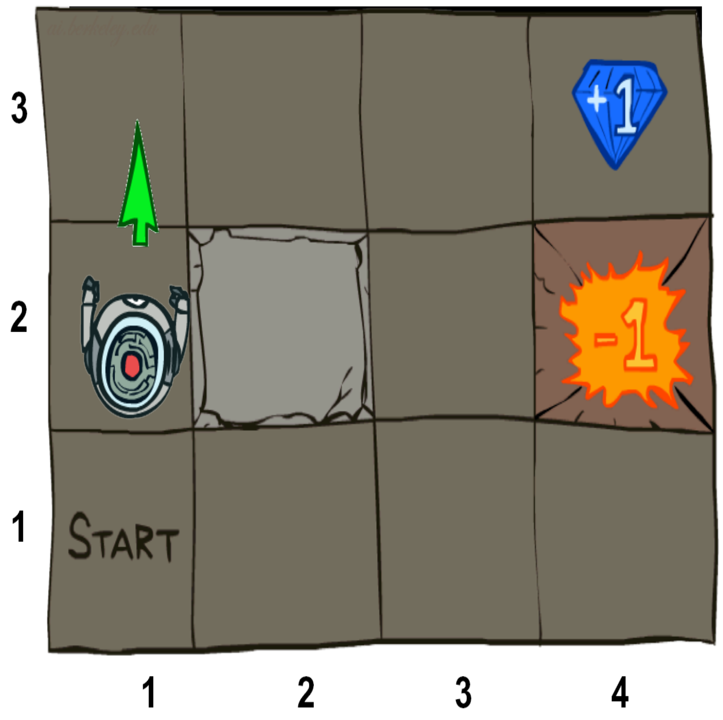
s=(1,1) action= tup, s'=(1,2), r = -.01



Start at (1,1)

s=(1,1) action= tup, s'=(1,2), r = -.01

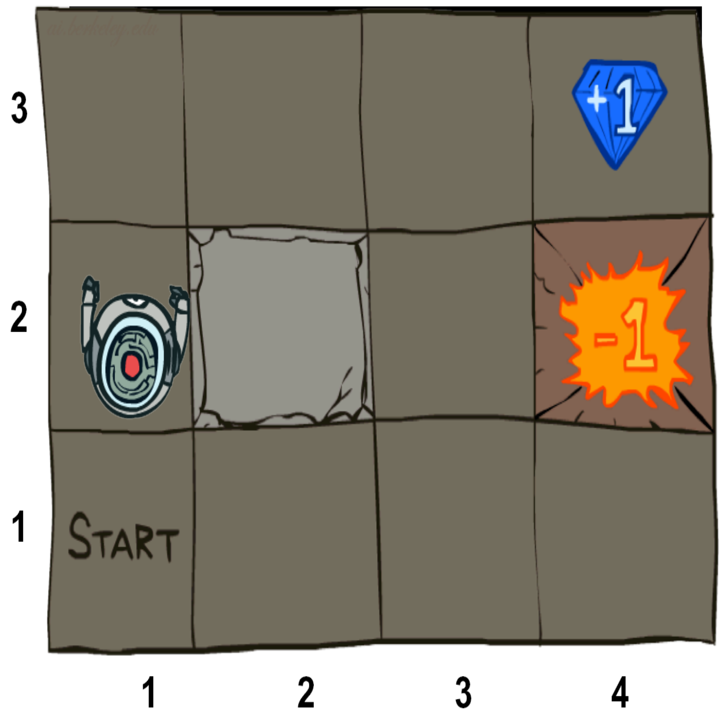
s=(1,2) action= tup



Start at (1,1)

s=(1,1) action= tup, s'=(1,2), r = -.01

s=(1,2) action= tup, s'=(1,2), r = -.01



Start at (1,1)

s=(1,1) action= tup, s'=(1,2), r = -.01

s=(1,2) action= tup, s'=(1,2), r = -.01

s=(1,2) action=tup, s'=(1,3), r = -.01

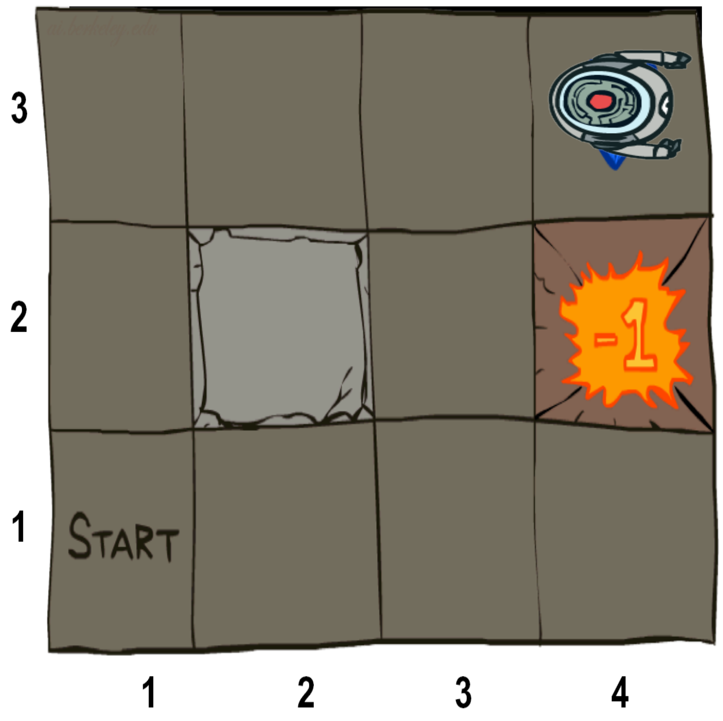
s=(1,3) action=tright, s'=(2,3), r = -.01

s=(2,3) action=tright, s'=(3,3), r = -.01

s=(3,3) action=tright, s'=(3,2), r = -.01

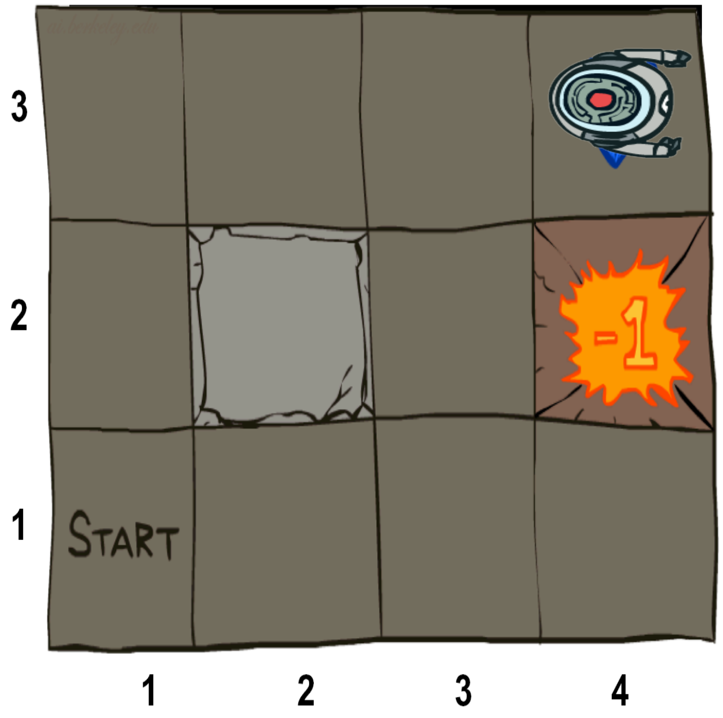
s=(3,2) action=tup, s'=(3,3), r = -.01

s=(3,3) action=tright, s'=(4,3), r = 1



Start at (1,1)

s=(1,1) action= tup,	s'=(1,2), r = -.01
s=(1,2) action= tup,	s'=(1,2), r = -.01
s=(1,2) action=tup,	s'=(1,3), r = -.01
s=(1,3) action=tright,	s'=(2,3), r = -.01
s=(2,3) action=tright,	s'=(3,3), r = -.01
s=(3,3) action=tright,	s'=(3,2), r = -.01
s=(3,2) action=tup,	s'=(3,3), r = -.01
s=(3,3) action=tright,	s'=(4,3), r = 1



Can use experience to estimate MDP T and R models

Start at (1,1)

s=(1,1) action= tup, s'=(1,2), r = -.01

s=(1,2) action= tup, s'=(1,2), r = -.01

s=(1,2) action=tup, s'=(1,3), r = -.01

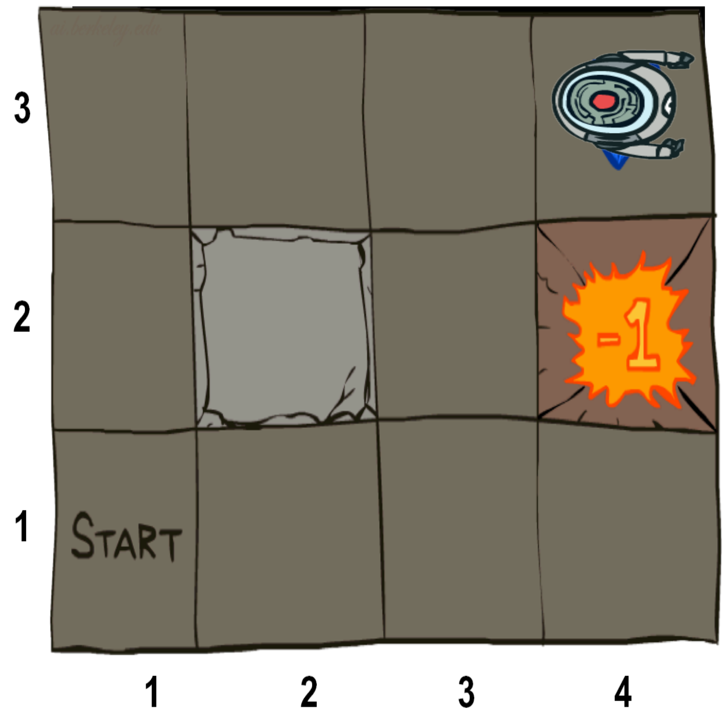
s=(1,3) action=tright, s'=(2,3), r = -.01

s=(2,3) action=tright, s'=(3,3), r = -.01

s=(3,3) action=tright, s'=(3,2), r = -.01

s=(3,2) action=tup, s'=(3,3), r = -.01

s=(3,3) action=tright, s'=(4,3), r = 1



Can use experience to estimate MDP T and R models

Estimate of T(<1,2>,tup,<1,3>) = 1 / 2

Adaption of drawing by Ketrina Yim

Carnegie Mellon University

- Given estimates of the transition model T and reward model R, we can do MDP policy evaluation to compute the value of our policy

Model-Based Passive Reinforcement Learning

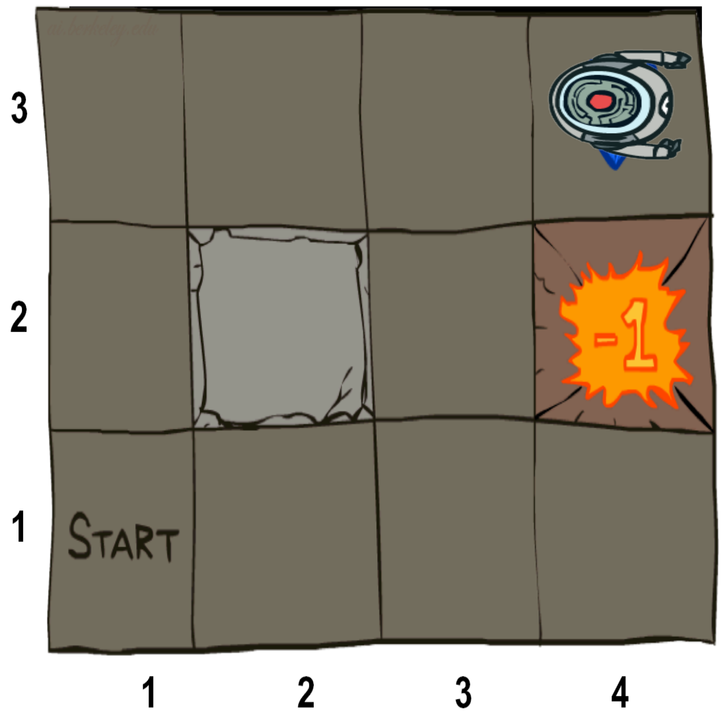
- Follow policy π
- Estimate MDP model parameters given observed transitions and rewards
    - If finite set of states and actions, can just count and average counts
- Use estimated MDP to do policy evaluation of π

Does This Give Us All the Parameters for a MDP?

- Follow policy π
- **Estimate MDP model parameters given observed transitions and rewards**
  - If finite set of states and actions, can just count and average counts
- Use estimated MDP to do policy evaluation of π

Start at (1,1)

s=(1,1) action= tup,	s'=(1,2), r = -.01
s=(1,2) action= tup,	s'=(1,2), r = -.01
s=(1,2) action=tup,	s'=(1,3), r = -.01
s=(1,3) action=tright,	s'=(2,3), r = -.01
s=(2,3) action=tright,	s'=(3,3), r = -.01
s=(3,3) action=tright,	s'=(3,2), r = -.01
s=(3,2) action=tup,	s'=(3,3), r = -.01
s=(3,3) action=tright,	s'=(4,3), r = 1



Estimate of T(<1,2>,tright,<1,3>)? No idea! Never tried this action…

Does This Give Us All the Parameters for a MDP? No. But Does That Matter for Computing Policy Value? No. Have all Parameters We Need! (After We Visit All States at Least Once)

- Follow policy π
- Estimate MDP model parameters given observed transitions and rewards
  - If finite set of states and actions, can just count and average counts
- Use estimated MDP to do policy evaluation of π

Carnegie Mellon University

Start at (1,1)

s=(1,1) action= tup, s'=(1,2), r = -.01

s=(1,2) action= tup, s'=(1,2), r = -.01

s=(1,2) action=tup, s'=(1,3), r = -.01

s=(1,3) action=tright, s'=(2,3), r = -.01

s=(2,3) action=tright, s'=(3,3), r = -.01

s=(3,3) action=tright, s'=(3,2), r = -.01

s=(3,2) action=tup, s'=(3,3), r = -.01

s=(3,3) action=tright, s'=(4,3), r = 1

s=(1,1) action= tup, s'=(2,1), r = -.01

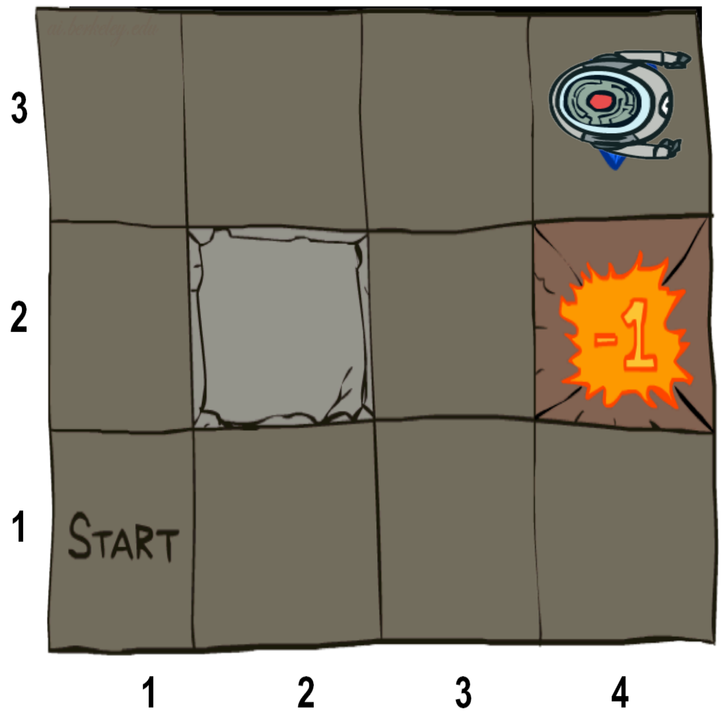
s=(2,1) action= tright, s'=(3,1), r = -.01

s=(3,1) action= tup, s'=(4,1), r = -.01

s=(4,1) action= tleft, s'=(3,1), r = -.01

s=(3,1) action= tup, s'=(3,2), r = -.01

s=(3,2) action= tup, s'=(4,2), r = -1



2 episodes of experience in MDP. Use to estimate MDP parameters & evaluate π

Start at (1,1)

s=(1,1) action= tup, s'=(1,2), r = -.01

s=(1,2) action= tup, s'=(1,2), r = -.01

s=(1,2) action=tup, s'=(1,3), r = -.01

s=(1,3) action=tright, s'=(2,3), r = -.01

s=(2,3) action=tright, s'=(3,3), r = -.01

s=(3,3) action=tright, s'=(3,2), r = -.01

s=(3,2) action=tup, s'=(3,3), r = -.01

s=(3,3) action=tright, s'=(4,3), r = 1

s=(1,1) action= tup, s'=(2,1), r = -.01

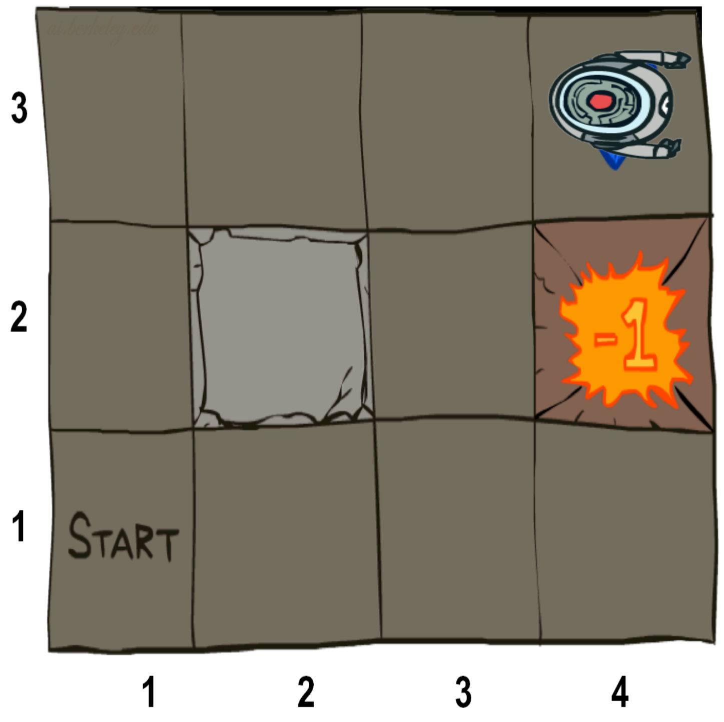
s=(2,1) action= tright, s'=(3,1), r = -.01

s=(3,1) action= tup, s'=(4,1), r = -.01

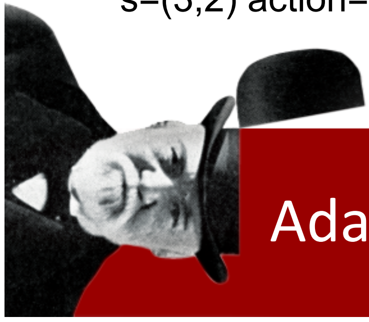
s=(4,1) action= tleft, s'=(3,1), r = -.01

s=(3,1) action= tup, s'=(3,2), r = -.01

s=(3,2) action= tup, s'=(4,2), r = -1



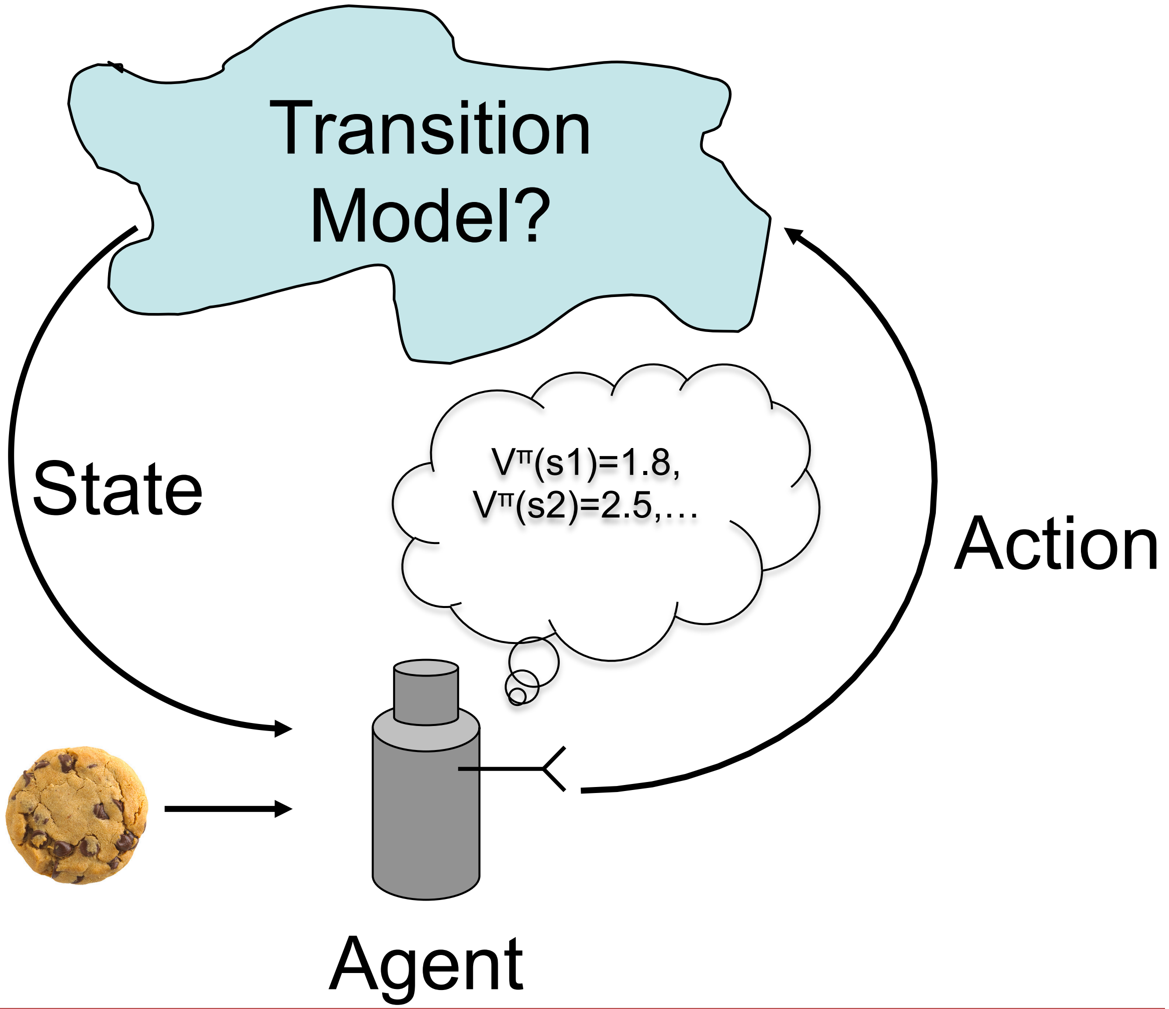
2 episodes of experience in MDP. Use to estimate MDP parameters & evaluate π
Is the computed policy value likely to be correct?
(1) Yes (2) No (3) Not sure



Passive Reinforcement Learning

Two Approaches

1. Build a model
2. **Model-free: directly estimate V^π**



Temporal Difference Learning

- No explicit model of T or R!
- But still estimate V and expectation **through samples**
- Update 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 sample of V: running average

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

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

Exponential Moving Avg

- Exponential moving average
  - The running interpolation update:

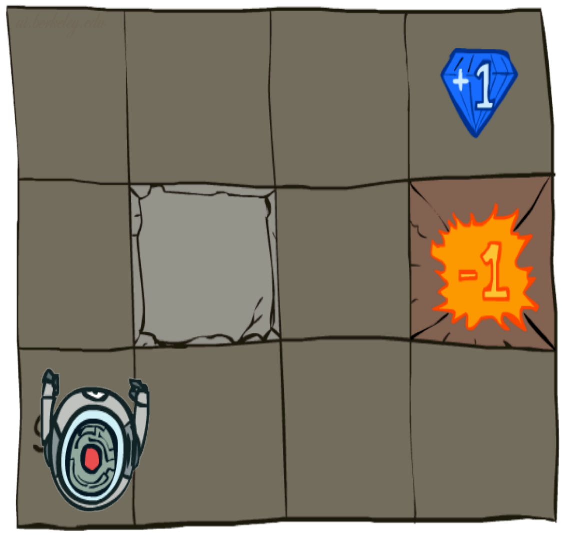
$$\bar{x}_n = (1-\alpha)\cdot\bar{x}_{n-1} + \alpha\cdot x_n$$

  - 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
- Decreasing learning rate (α) can give converging avgs

TD Learning Example

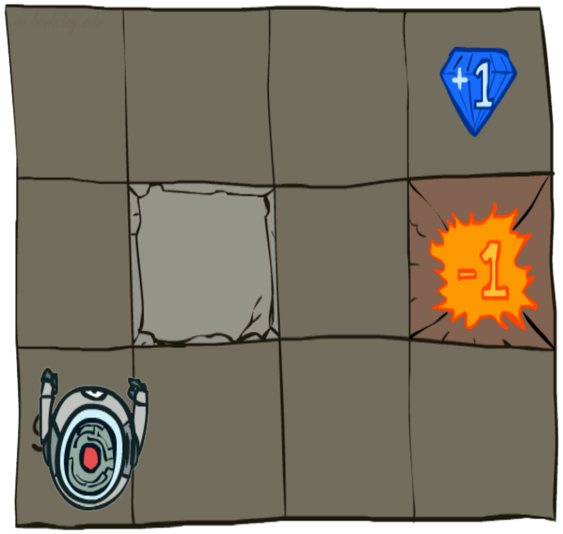


Initialize all $V^{\pi}(s)$ values: $V^{\pi}(s) = 0$

State	$V^{\pi}(s)$
(1,1)	0
(1,2)	0
(1,3)	0
(4,1)	0
(2,1)	0
(2,3)	0
(3,1)	0
(3,2)	0
(3,3)	0

TD Learning Example $\alpha=0.1, \gamma=1$

s=(1,1) action= tup, s'=(1,2), r = -.01



Update $V^\pi((1,1))$

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

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

sample = -0.01 + V^π((1,2)) = -0.01

V^π((1,1)) = (1-α) V^π((1,1)) + α*sample

= .9*0 + 0.1*-0.01 = -0.001

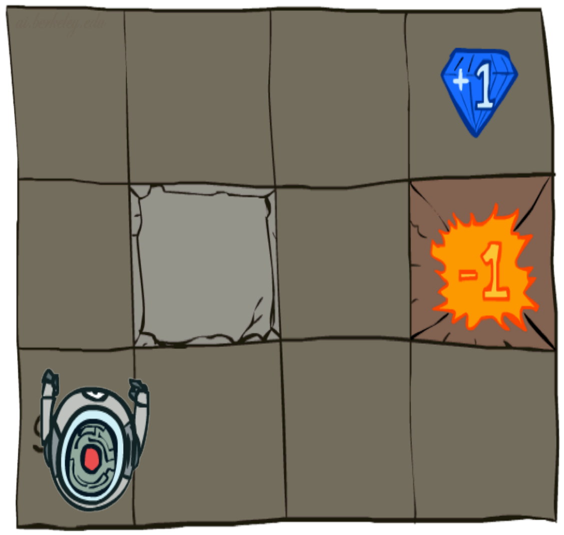
State	$V^\pi(s)$
(1,1)	0
(1,2)	0
(1,3)	0
(4,1)	0
(2,1)	0
(2,3)	0
(3,1)	0
(3,2)	0
(3,3)	0

Adaption of drawing by Ketrina Yim

Carnegie Mellon University

TD Learning Example $\alpha=0.1, \gamma=1$

s=(1,1) action= tup, s'=(1,2), r = -.01



Update $V^{\pi}((1,1))$

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

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

sample = -0.01 + V^{π}((1,2)) = -0.01

V^{π}((1,1)) = (1-α) V^{π}((1,1)) + α*sample

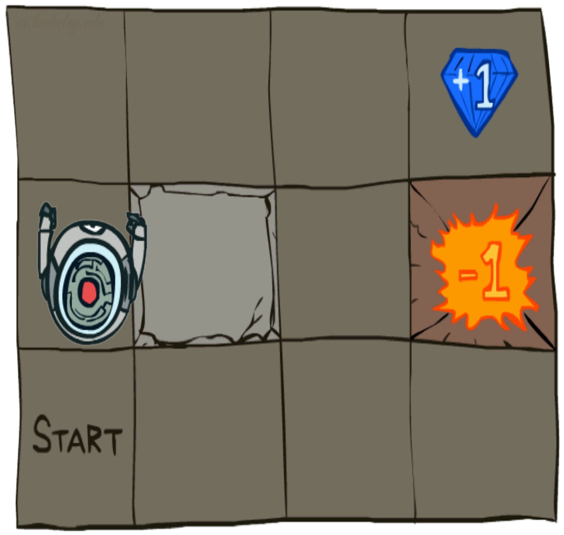
= .9*0 + 0.1*-0.01 = -0.001

State	$V^{\pi}(s)$
(1,1)	-0.001
(1,2)	0
(1,3)	0
(4,1)	0
(2,1)	0
(2,3)	0
(3,1)	0
(3,2)	0
(3,3)	0

TD Learning Example α=0.1, γ=1

s=(1,1) action= tup, s'=(1,2), r = -.01

s=(1,2) action= tup, s'=(1,2), r = -.01



State	$V^{\pi}(s)$
(1,1)	-0.001
(1,2)	0
(1,3)	0
(4,1)	0
(2,1)	0
(2,3)	0
(3,1)	0
(3,2)	0
(3,3)	0

Adaption of drawing by Ketrina Yim

Problems with Passive Learning

- Agent wants to ultimately learn to act to gather high reward in the environment.
- If have a deterministic policy, gives no experience for other actions

Can We Learn Optimal Values & Policy?

- Consider acting randomly in the world
- Can such experience allow the agent to learn the optimal values and policy?

Recall Model-Based Passive Reinforcement Learning

- **Follow policy π**
- Estimate MDP model parameters given observed transitions and rewards
    - If finite set of states and actions, can just count and average counts
- Use estimated MDP to do policy evaluation of π

Model-Based RL w/Random Actions

- Choose actions randomly
- Estimate MDP model parameters given observed transitions and rewards
    - If finite set of states and actions, can just count and average counts
- Use estimated MDP to compute estimate of optimal values and policy

Will the computed values and policy converge to the true optimal values and policy in the limit of infinite data?

Reachability

- When acting randomly forever, still need to be able to visit each state and take each action many times
- Want all states to be reachable from any other state
- Quite mild assumption but doesn't always hold

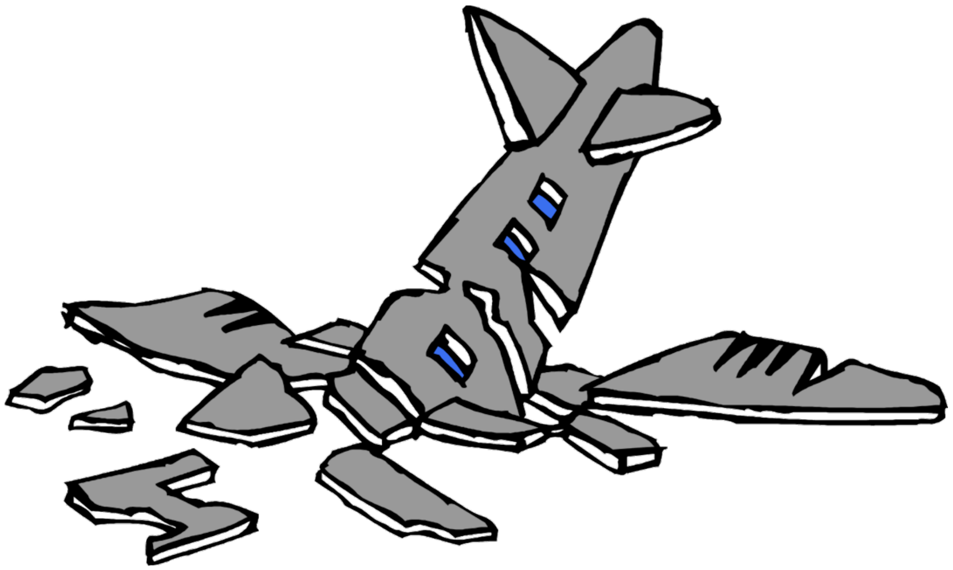


Image source: http://ancient-heritage.blogspot.com/2014/05/crash-course-on-flying-in-face-of-logic.html

Model-Free Learning with Random Actions?

- Previously introduced temporal-difference learning for policy evaluation:
  - As act in the world visit (s,a,r,s’,a’,r’,…)
  - Update V^{π} estimates at each step

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

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

- Over time updates mimic Bellman updates

Q-Learning

- Running estimate of state-action Q values (instead of V in TD learning)
- Update Q(s,a) every time experience (s,a,s’,r(s,a,s’))
  - Consider old estimate Q(s,a)
  - Create new sample estimate

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

  - Update estimate of Q(s,a)

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha * sampleQ(s,a)$$

Q-Learning

- Update Q(s,a) every time experience (s,a,s’,r(s,a,s’))

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

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha * sampleQ(s,a)$$

- Intuition: using samples to approximate
  - Future rewards
  - Expectation over next states due to transition model uncertainty

Q-Learning Properties

- If acting randomly, Q-learning converges to optimal state—action values, and also therefore finds optimal policy
- Off-policy learning
    - Can act in one way
    - But learning values of another policy (the optimal one!)