

# Lecture 2: From MDP Planning to RL Basics

CS234: RL

Emma Brunskill

Spring 2017

# Recap: Value Iteration (VI)

1. Initialize  $V_0(s_i)=0$  for all states  $s_i$ ,
2. Set  $k=1$
3. Loop until [finite horizon, convergence]
  - For each state  $s$ ,

$$V_{k+1}(s) = \max_a \left[ r(s, a) + \gamma \sum_{s' \in S} p(s'|a, s) V_k(s') \right]$$

4. Extract Policy



Bellman  
backup

$V_k$  is optimal value if horizon=k

1. Initialize  $V_0(s_i)=0$  for all states  $s_i$ ,
2. Set  $k=1$
3. Loop until [finite horizon, convergence]
  - For each state  $s$ ,

$$V_{k+1}(s) = \max_a \left[ r(s, a) + \gamma \sum_{s' \in S} p(s'|a, s) V_k(s') \right]$$

4. Extract Policy

Bellman  
backup



# Value vs Policy Iteration

- Value iteration:

- Compute optimal value if horizon= $k$ 
  - Note this can be used to compute optimal policy if horizon =  $k$
- Increment  $k$

- Policy iteration:

- Compute infinite horizon value of a policy
- Use to select another (better) policy
- Closely related to a very popular method in RL: policy gradient

# Policy Iteration (PI)

1.  $i=0$ ; Initialize  $\pi_0(s)$  randomly for all states  $s$
2. Converged = 0;
3. While  $i \neq 0$  or  $|\pi_i - \pi_{i-1}| > 0$ 
  - $i=i+1$   $V^\pi$
  - Policy evaluation
  - Policy improvement

# Policy Evaluation

1. Use minor variant of value iteration

$$V_{k+1}(s) = \underset{a}{\text{argmax}} \left[ r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V_k(s') \right]$$

# Policy Evaluation

1. Use minor variant of value iteration

$$V_{k+1}(s) = \max_a \left[ r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V_k(s') \right]$$

$$V_{k+1}^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_k^\pi(s')$$

→ restricts action to be one chosen by policy

# Policy Evaluation

## 1. Use minor variant of value iteration

$$V_{k+1}(s) = \max_a \left[ r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V_k(s') \right]$$

$$V_{k+1}^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_k^\pi(s')$$

## 2. Analytic solution (for discrete set of states)

- Set of linear equations (no max!)
- Can write as matrices and solve directly for V



# Policy Evaluation: Example

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Deterministic actions of TryLeft or TryRight
- Reward: +1 in state S1, +10 in state S7, 0 otherwise
- Let  $\pi_0(s)$ =TryLeft for all states (e.g. always go left)
- Assume  $\gamma=0$ . What is the value of this policy in each  $s$ ?

$$V_{k+1}^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_k^{\pi}(s')$$

# Policy Improvement

- Have  $V^\pi(s)$  for all  $s$  (from policy evaluation step!)
- Want to try to find a better (higher value) policy
- Idea:
  - Find the state-action Q value of doing an action followed by following  $\pi$  forever, for each state
  - Then take argmax of  $Q_s$

# Policy Improvement

- Compute Q value of different 1st action and then following  $\pi_i$

$$Q^{\pi_i}(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V^{\pi_i}(s')$$

- Use to extract a new policy

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$

# Delving Deeper Into Improvement

$$Q^{\pi_i}(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V^{\pi_i}(s')$$

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

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$

- So if take  $\pi_{i+1}(s)$  then followed  $\pi_i$  forever,
  - expected sum of rewards would be at least as good as if we had always followed  $\pi_i$
- But new proposed policy is to always follow  $\pi_{i+1} \dots$

# Monotonic Improvement in Policy

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

# Proof

$$\begin{aligned} V^\pi(s) &\leq \max_a Q^\pi(s, a) \\ &= \sum_{s' \in S} p(s' | s, \pi'(s)) \left[ R(s, \pi'(s), s') + \gamma V^\pi(s') \right] \\ &\leq \sum_{s' \in S} p(s' | s, \pi'(s)) \left[ R(s, \pi'(s), s') + \gamma \max_{a'} Q^\pi(s', a') \right] \\ &= \sum_{s' \in S} p(s' | s, \pi'(s)) \left[ R(s, \pi'(s), s') + \right. \\ &\quad \left. \gamma \sum_{s'' \in S} p(s'' | s', \pi'(s')) (R(s', \pi'(s'), s'') + \gamma V^\pi(s'')) \right] \\ &\dots \leq V^{\pi'}(s) \end{aligned}$$

## If Policy Doesn't Change ( $\pi_{i+1}(s) = \pi_i(s)$ for all $s$ ) Can It Ever Change Again in More Iterations?

- Recall policy improvement step

$$Q^{\pi_i}(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V^{\pi_i}(s')$$

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$

# Policy Iteration (PI)

1.  $i=0$ ; Initialize  $\pi_0(s)$  randomly for all states  $s$
2. Converged = 0;
3. While  $i \neq 0$  or  $|\pi_i - \pi_{i-1}| > 0$ 
  - $i=i+1$
  - Policy **evaluation**: Compute  $V^\pi$
  - Policy **improvement**:

$$Q^{\pi_i}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi_i}(s')$$

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$



## Policy Iteration Can Take At Most $|A|^{|S|}$ Iterations (Size of # Policies)

1.  $i=0$ ; Initialize  $\pi_0(s)$  randomly for all states  $s$
2. Converged = 0;
3. While  $i \neq 0$  or  $|\pi_i - \pi_{i-1}| > 0$ 
  - $i=i+1$
  - Policy **evaluation**: Compute  $V^\pi$
  - Policy **improvement**:

$$Q^{\pi_i}(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^{\pi_i}(s')$$

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$

\* For finite state and action spaces

## Policy Iteration

Fewer Iterations

More expensive per iteration

## Value Iteration

More iterations

Cheaper per iteration

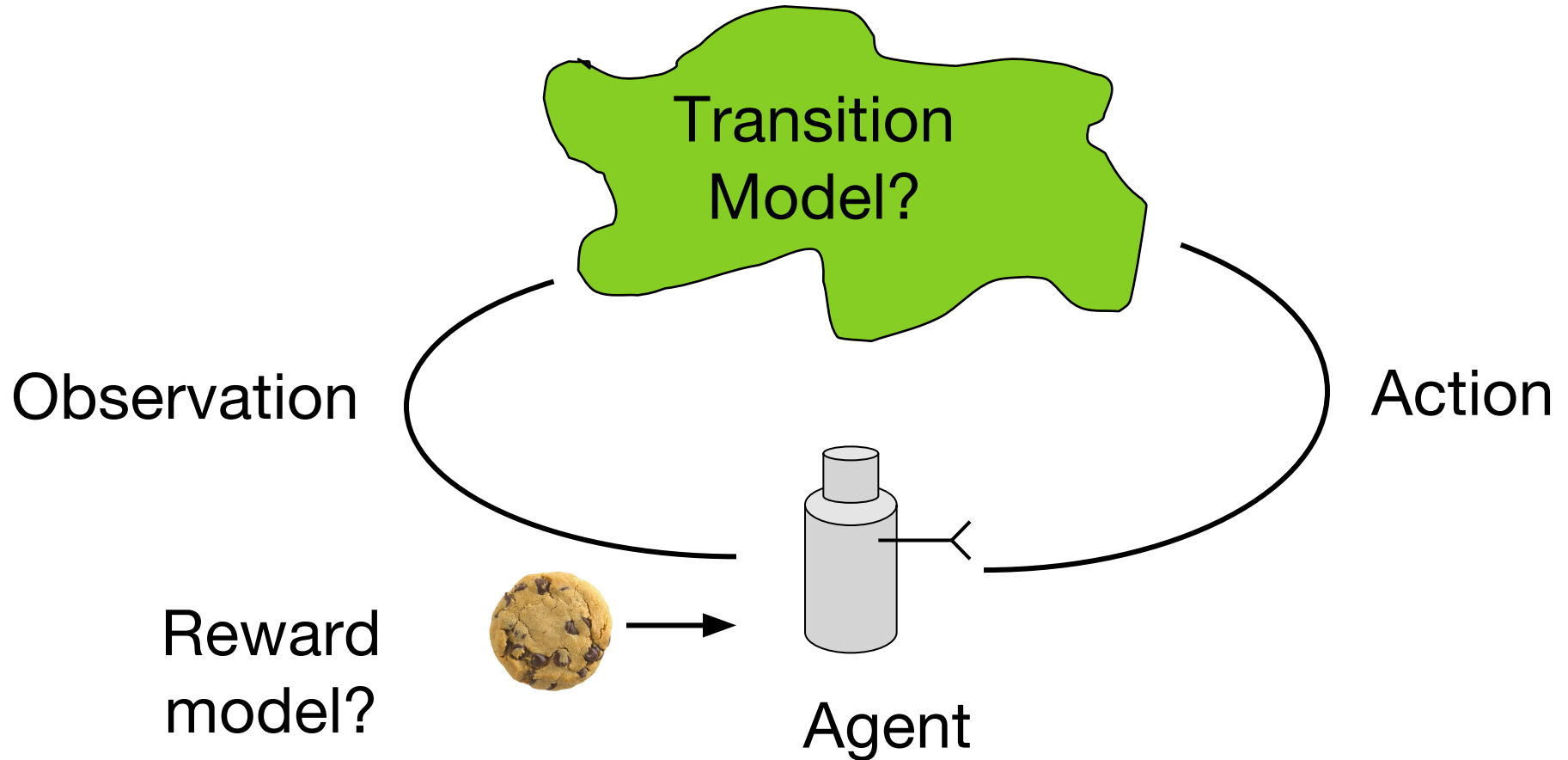
# MDPs: What You Should Know

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

# Reasoning Under Uncertainty

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

# Reinforcement Learning



*Goal: Maximize expected sum of future rewards*

# MDP Planning vs Reinforcement Learning

- No world models (or simulators)
- Have to learn how world works by trying things out

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

# Policy Evaluation While Learning

- Before figuring out how should act
- 1st figure out how good a particular policy is (passive RL)

# Passive RL

1. Estimate a model (and use to do policy evaluation)
2. Q-learning




# Learn a Model

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Start in state S3, take TryLeft, go to S2
- In state S2, take TryLeft, go to S2
- In state S2, take TryLeft, go to S1
- What's an estimate of  $p(s'=S2 \mid S=S2, a=\text{TryLeft})$ ?

# Use Maximum Likelihood Estimate

E.g. Count & Normalize

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Start in state S3, take TryLeft, go to S2
- In state S2, take TryLeft, go to S2
- In state S2, take TryLeft, go to S1
- What's an estimate of  $p(s'=S2 \mid S=S2, a=\text{TryLeft})$ ?
  - 1/2

# Model-Based Passive Reinforcement Learning

- Follow policy  $\pi$
- Estimate MDP model parameters from data
  - If finite set of states and actions: count & average
- Use estimated MDP to do policy evaluation of  $\pi$


# Model-Based Passive Reinforcement Learning

- Follow policy  $\pi$
- Estimate MDP model parameters from data
  - If finite set of states and actions: count & average
- Use estimated MDP to do policy evaluation of  $\pi$
- Does this give us dynamics model parameter estimates for all actions?
- How good is the model parameter estimates?
- What about the resulting policy value estimate?

# Model-Based Passive Reinforcement Learning

- Follow policy  $\pi$
  - Estimate MDP model parameters from data
    - If finite set of states and actions: count & average
  - Use estimated MDP to do policy evaluation of  $\pi$
- 
- Does this give us dynamics model parameter estimates for all actions?
    - No. But all ones need to estimate the value of the policy.
  - How good is the model parameter estimates?
    - Depends on amount of data we have
  - What about the resulting policy value estimate?
    - Depends on quality of model parameters

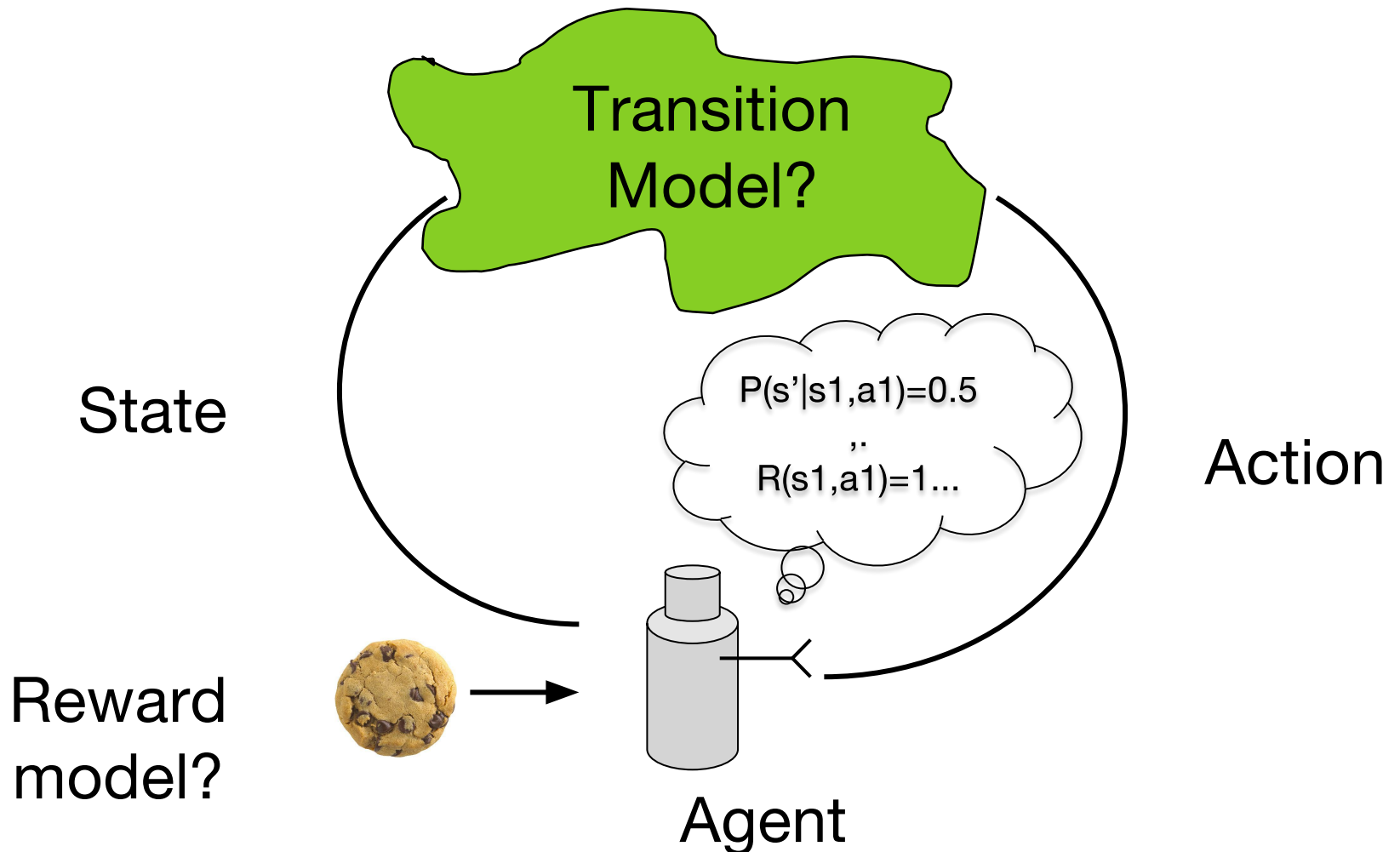
# Good Estimate if Use 2 Data Points?

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Start in state S3, take TryLeft, go to S2,  $r=0$
- In state S2, take TryLeft, go to S2,  $r = 0$
- In state S2, take TryLeft, go to S1,
- What's an estimate of  $p(s'=S2 \mid S=S2, a=\text{TryLeft})$ ?
  - $1/2$

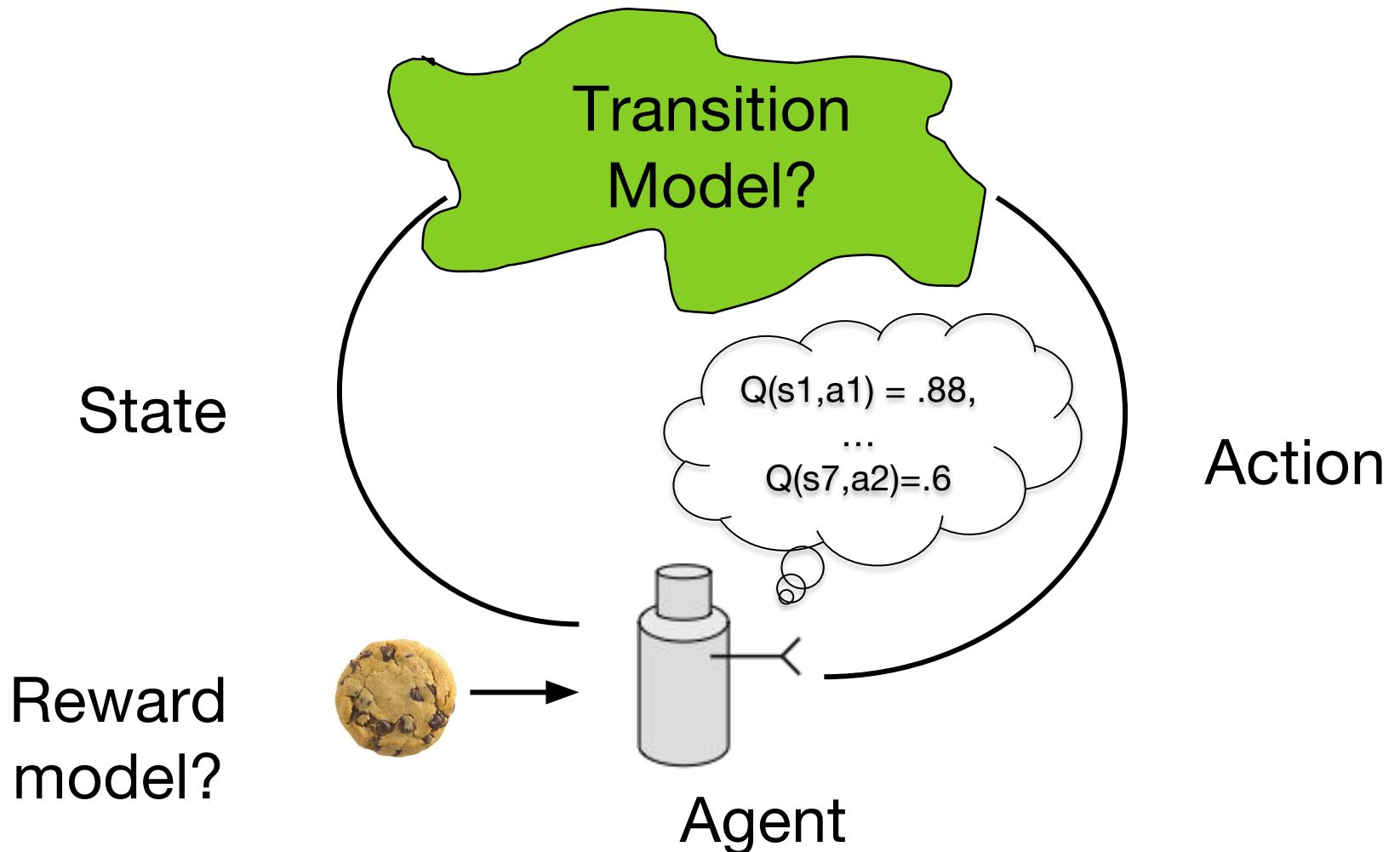
# Model-based Passive RL:

Agent has an estimated model in its head



# Model-free Passive RL:

Only maintain estimate of Q





# Q-values

- Recall that  $Q^\pi(s,a)$  values are
  - expected discounted sum of rewards over H step horizon
  - if start with action a and follow  $\pi$
- So how could we directly estimate this?

# Q-values

$$Q^{\pi_i}(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V^{\pi_i}(s')$$

- Want to approximate the above with data
- Note if only following  $\pi$ , only get data for  $a=\pi(s)$

# Q-values

$$Q^{\pi_i}(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V^{\pi_i}(s')$$

- Want to approximate the above with data
- Note if only following  $\pi$ , only get data for  $a=\pi(s)$
- TD-learning
  - Approximate expectation with samples
  - Approximate future reward with estimate

# Temporal Difference Learning


$$V^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V^{\pi}(s')$$

- Maintain estimate of  $V^{\pi}(s)$  for all states
  - Update  $V^{\pi}(s)$  each time after each transition  $(s, a, s', r)$
  - Likely outcomes  $s'$  will contribute updates more often
  - Approximating expectation over next state with samples
  - Running average

$$V_{samp}(s) = r + \gamma V^{\pi}(s')$$

$$V^{\pi}(s) = (1 - \alpha) V^{\pi}(s) + \alpha V_{samp}(s)$$

Decrease  
learning rate  
over time  
(why?)



$$V_{samp}(s) = r + \gamma V^{\pi}(s')$$

$$V^{\pi}(s) = (1 - \alpha)V^{\pi}(s) + \alpha V_{samp}(s)$$

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Policy: TryLeft in all states, use alpha = 0.5,  $\gamma=1$
- Set  $V^{\pi}=[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$ ,
- Start in state S3, take TryLeft, get  $r=0$ , go to S2
  - $V_{samp}(S3) = 0 + 1 * 0 = 0$
  - $V^{\pi}(S3)=(1-0.5)*0 + .5*0 = 0$  (no change!)

$$V_{samp}(s) = r + \gamma V^{\pi}(s')$$

$$V^{\pi}(s) = (1 - \alpha)V^{\pi}(s) + \alpha V_{samp}(s)$$

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Policy: TryLeft in all states, use alpha = 0.5,  $\gamma=1$
- Set  $V^{\pi}=[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$ ,
- Start in state S3, take TryLeft, go to S2, get  $r=0$
- $V^{\pi}=[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$
- In state S2, take TryLeft, get  $r=0$ , go to S1
  - $V_{smp}(S2) = 0 + 1 * 0 = 0$
  - $V^{\pi}(S2)=(1-0.5)*0 + .5*0 = 0$  (no change!)

$$V_{samp}(s) = r + \gamma V^{\pi}(s')$$

$$V^{\pi}(s) = (1 - \alpha)V^{\pi}(s) + \alpha V_{samp}(s)$$

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Policy: TryLeft in all states, use  $\alpha = 0.5$ ,  $\gamma = 1$
- Start in state S3, take TryLeft, go to S2, get  $r=0$
- In state S2, take TryLeft, go to S1, get  $r=0$
- $V^{\pi} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$
- In state S1, take TryLeft, go to S1, get  $r=+1$ 
  - $V_{smp}(S1) = 1 + 1 * 0 = 1$
  - $V^{\pi}(S1) = (1-0.5)*0 + .5*1 = 0.5$

$$V_{samp}(s) = r + \gamma V^{\pi}(s')$$

$$V^{\pi}(s) = (1 - \alpha)V^{\pi}(s) + \alpha V_{samp}(s)$$

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Policy: TryLeft in all states, use alpha = 0.5,  $\gamma=1$
- Start in state S3, take TryLeft, go to S2, get  $r=0$
- In state S2, take TryLeft, go to S1, get  $r=0$
- $V^{\pi}=[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$
- In state S1, take TryLeft, go to S1, get  $r=+1$
- $V^{\pi}=[0.5 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$



# Problems with Passive Learning

- Want to make good decisions
- Initial policy may be poor -- don't know what to pick
- And getting only experience for that policy

# 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  $\pi$**
- Estimate MDP model params from observed transitions & rewards
  - If finite set of states and actions, count & avg counts
- Use estimated MDP to do policy evaluation of  $\pi$

# Recall Model-Based Passive Reinforcement Learning

- **Choose actions randomly**
- Estimate MDP model params from observed transitions & rewards
  - If finite set of states and actions, count & avg counts
- **Use estimated MDP to compute estimate of optimal value and policy**
- **Will policy converge to optimal value & policy**
  - (In limit of infinite data)?

# Yes, if have 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

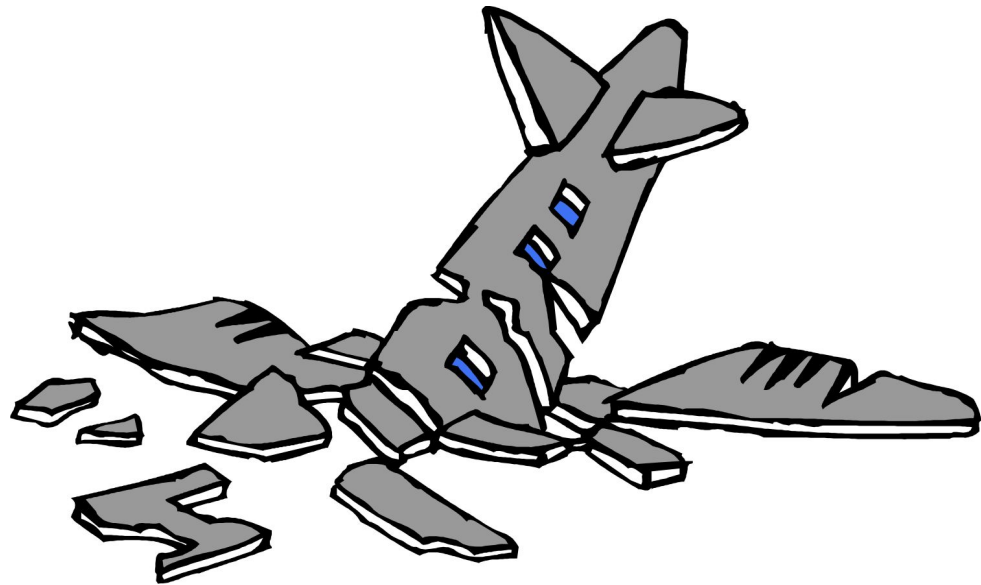


Image source:  
<http://ancient-heritage.blogspot.com/2014/05/crash-course-on-flying-in-face-of-logic.html>

# Model-Free Learning w/Random Actions

- TD learning for policy evaluation:
  - As act in the world go through  $(s, a, r, s', a', r', \dots)$
  - Update  $V^\pi$  estimates at each step
- Over time updates mimic Bellman updates
- Now do for Q values

# Q-Learning

- Update  $Q(s,a)$  every time experience  $(s,a,s',r)$ 
  - Create new sample estimate

$$\begin{aligned}Q_{samp}(s, a) &= r + \gamma V(s') \\ &= r + \gamma \max_{a'} Q(s', a')\end{aligned}$$

- Update estimate of  $Q(s,a)$

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha Q_{samp}(s, a)$$

# Q-Learning Properties

- If acting randomly\*, Q-learning converges  $Q^*$ 
  - Optimal Q values
  - Finds optimal policy
- Off-policy learning
  - Can act in one way
  - But learning values of another  $\pi$  (the optimal one!)

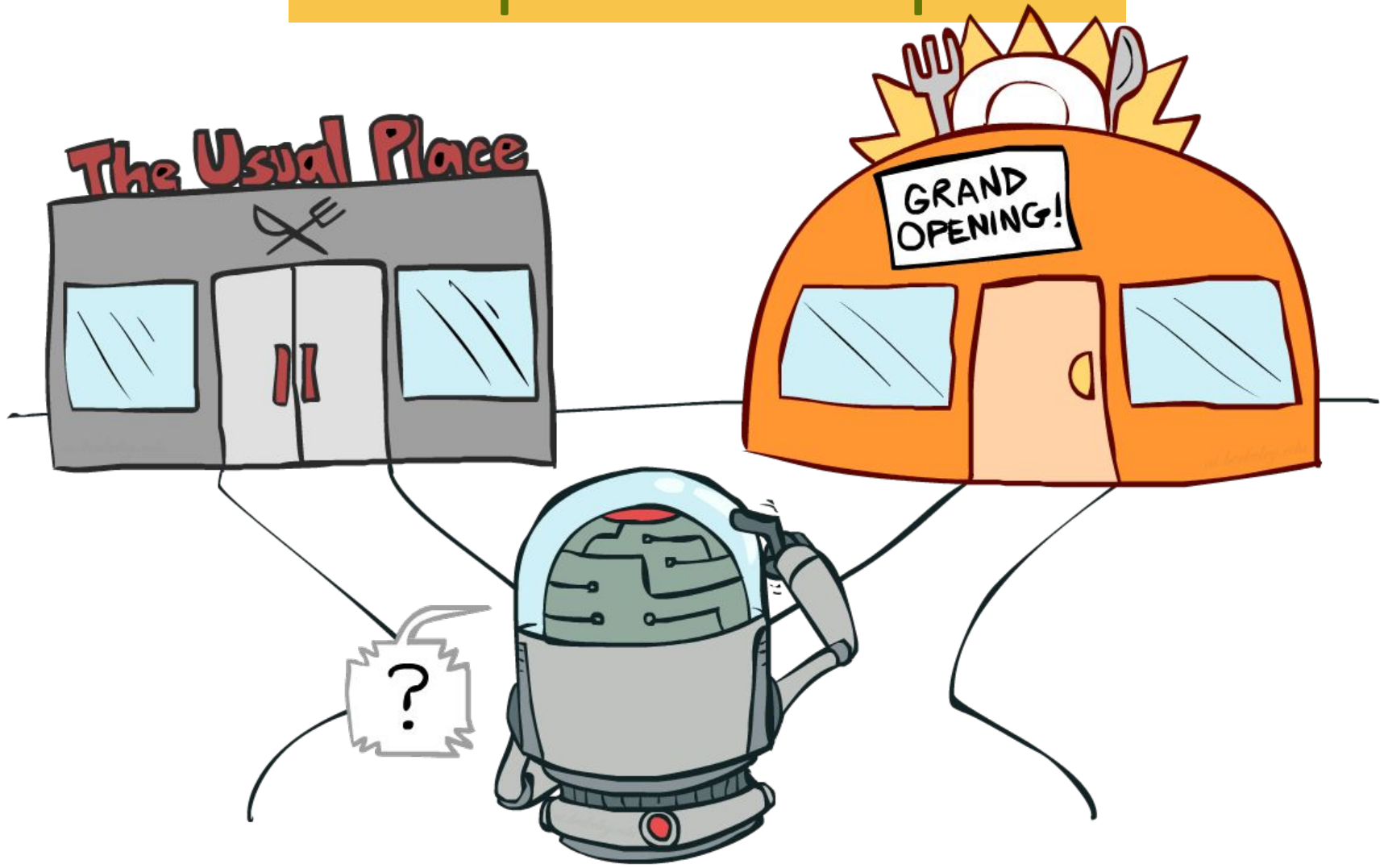
\*Again, under mild reachability assumptions



# Towards Gathering High Reward

- Fortunately, acting randomly is sufficient, but not necessary, to learn the optimal values and policy
- Ultimately want to learn to get large reward

# To Explore or Exploit?



# Simple Approach: E-greedy

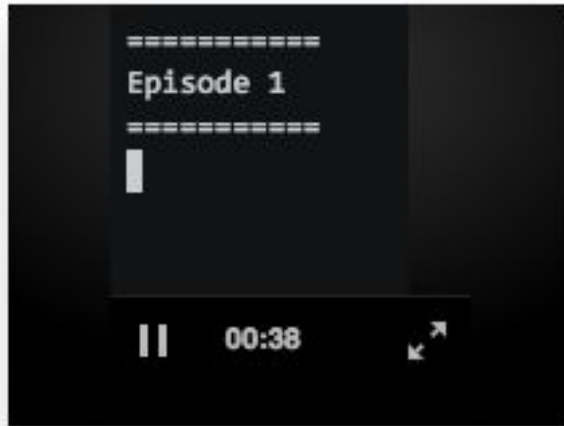
- With probability  $1-e$ 
  - Choose  $\operatorname{argmax}_a Q(s,a)$
- With probability  $e$ 
  - Select random action
- Guaranteed to compute optimal policy
- But even after millions of steps still won't always be following  $\operatorname{argmax}$  of  $Q(s,a)$

# Greedy in Limit of Infinite Exploration (GLIE)

- E-Greedy approach
- But decay epsilon over time
- Eventually will be following optimal policy almost all the time
- We'll talk more about exploration/exploitation later in the course

# Homework 1 Will Be Released This Week

---



FrozenLake-v0

Find a safe path across a  
grid of ice and water tiles.



FrozenLake8x8-v0

- Review/practice basic MDP planning
- Get familiar with Open AI gym for basic RL

# What You Should Know

- Define MDP, Bellman operator, contraction, model, Q-value, policy
- Contrast MDP planning and RL
- Be able to implement
  - Value iteration, policy iteration, Q-learning and model-based RL
- Contrast benefits and weaknesses of Q-learning and model-based RL
  - On homework!
  - Data efficiency, computational complexity, etc.