Reinforcement Learning++

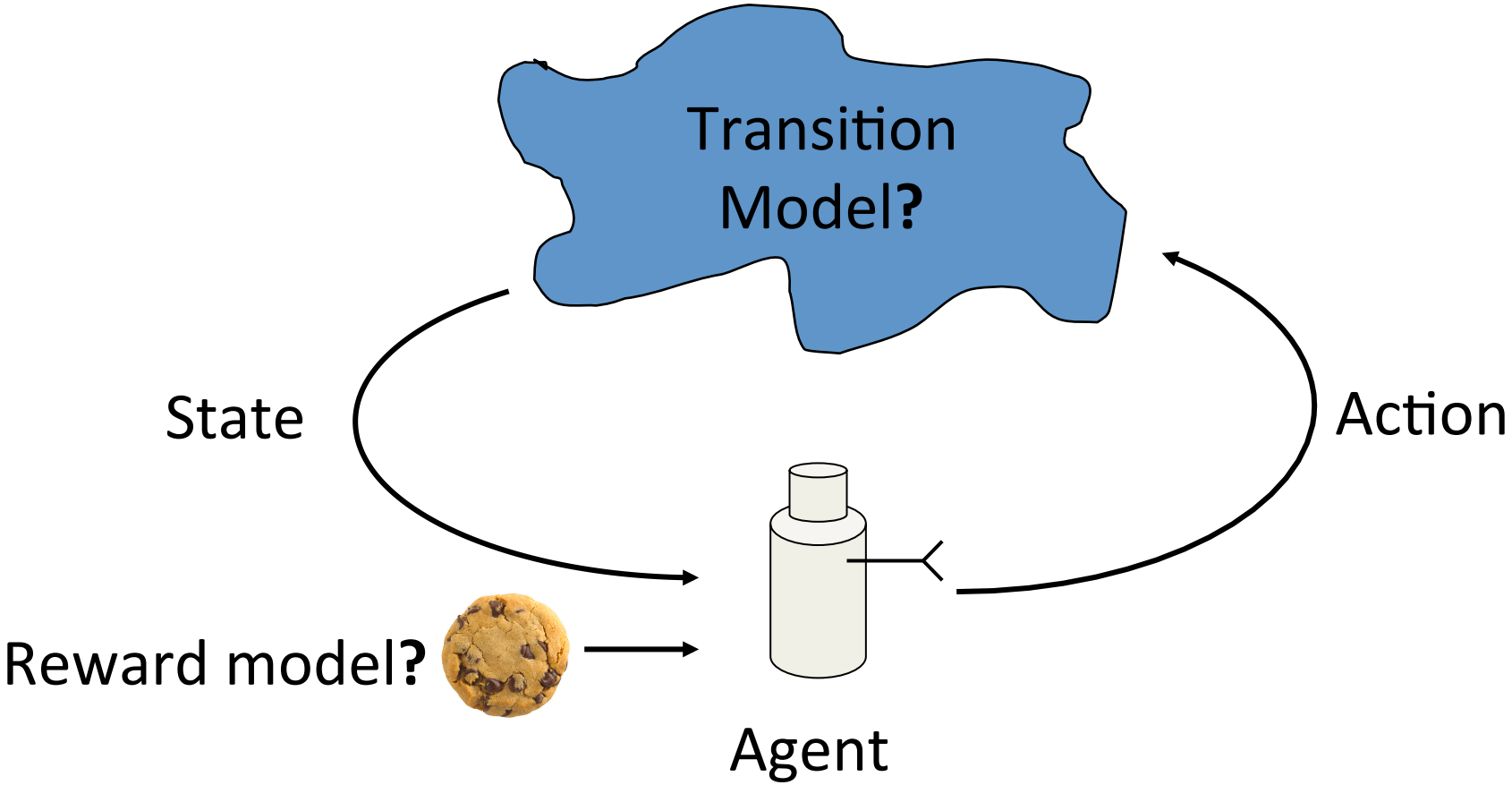
Emma Brunskill (today)

Ariel Procaccia

Recall MDPs: What You Should Know

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

Reinforcement Learning



Goal: Maximize expected sum of future rewards

Recap of Last Time

- Model-based RL when select actions randomly
  - Estimate a model of the dynamics and rewards from data (e.g. T(s1|s2,a2) ~ 0.3)
  - Do MDP planning given those estimated models
- Q-learning
  - No model of dynamics and rewards
  - Directly estimate state-action value function

Q-Learning

- At each step, for current state s and action taken
  - Observe r and s'
  - Update Q(s,a)

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

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

- 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!)

Towards Gathering High Reward

- Fortunately, acting randomly is sufficient, but not necessary, to learn the optimal values and policy

How to Act?

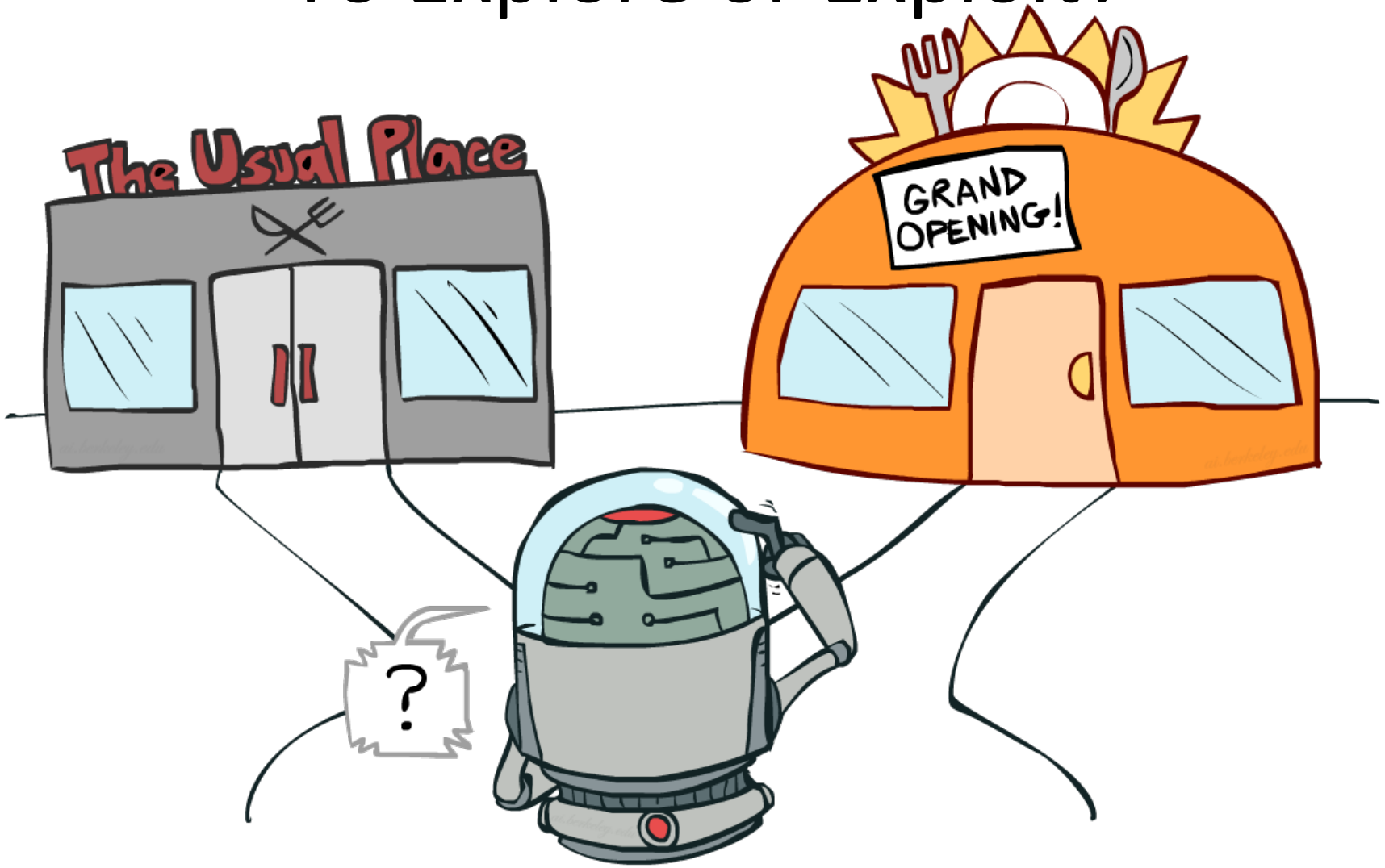
- Initialize s to a starting state
- Initialize Q(s,a) values
- For t=1,2,…
  - Choose a = argmax Q(s,a)
  - Observe s’,r(s,a,s’)
  - Update/Compute Q values

Is this Approach Guaranteed to Learn Optimal Policy?

- Initialize s to a starting state
- Initialize Q(s,a) values
- For t=1,2,…
  - Choose a = argmax Q(s,a)
  - Observe s',r(s,a,s')
  - Update/Compute Q values (using model-based or Q-learning approach)

1. Yes 2. No 3. Not sure

To Explore or Exploit?



Drawing by Ketrina Yim

Simple Approach: E-greedy

- With probability 1-e
  - Choose 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 policy compute (the argmax 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

How should we evaluate the performance of an algorithm?

How should we evaluate the performance of an algorithm?

- Computational efficiency
- How much reward gathered under algorithm?

The Speed of Learning and Speeding Learning

Objectives for an RL Algorithm

- Asymptotic guarantees
  - In limit converge to a policy identical to the optimal policy if knew unknown model parameters

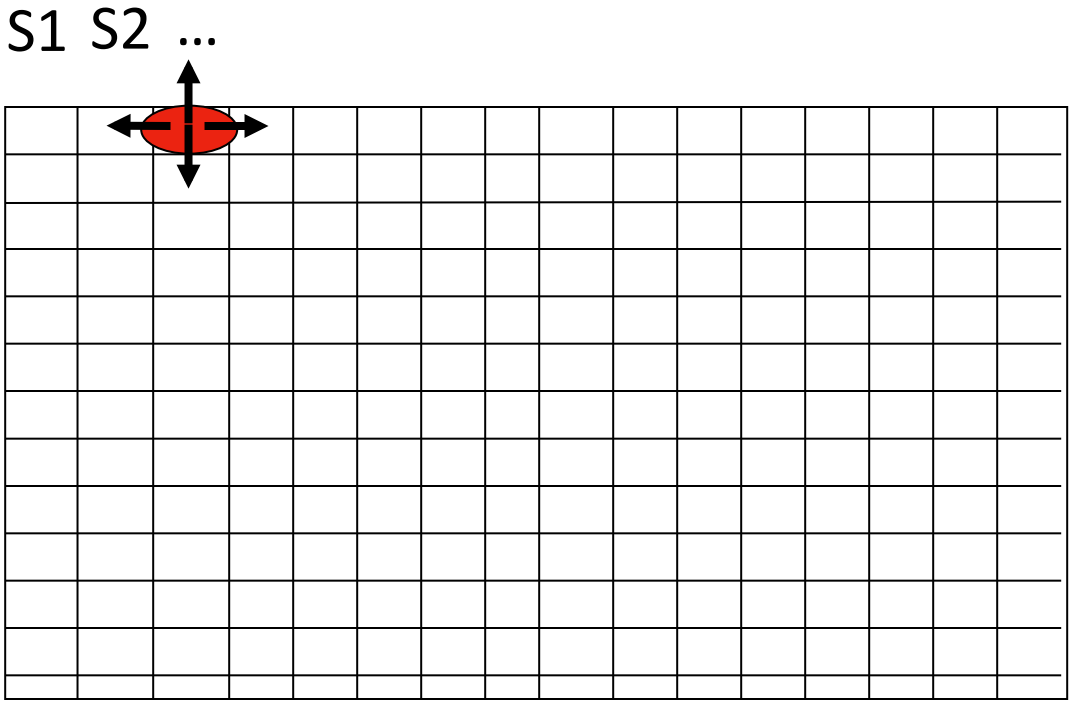
Objectives for an RL Algorithm

- Asymptotic guarantees
  - In limit converge to a policy identical to the optimal policy if knew unknown model parameters
  - Q-learning! (under what conditions?)
- **Probably Approximately Correct**
  - On all but finite number of samples, choose action whose expected reward is close to expected reward of action take if knew model parameters
  - E^3 (Kearns & Singh), R-MAX (Brafman & Tennenholtz)

Model-Based RL

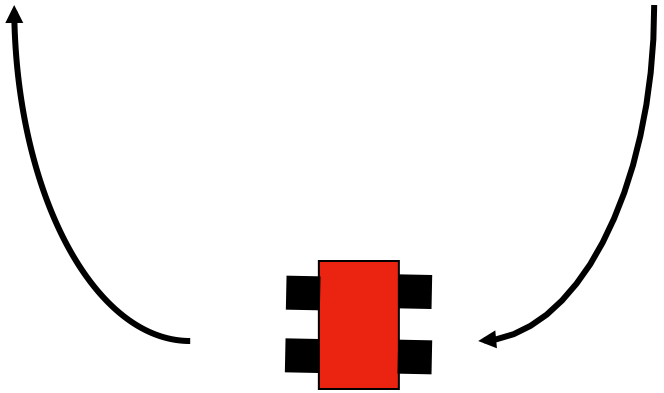
- Given data seen so far
- Build an explicit model of the MDP
- Compute policy for it
- Select action for current state given policy, observe next state and reward
- Repeat

R-max (Brafman & Tennenholtz)



R-max is Model-based RL

Think hard: estimate models & compute policies



Act in world

Rmax leverages optimism under uncertainty!

R-max Algorithm: Initialize: Define "Known" MDP

Known/ Unknown

	S1	S2	S3	S4	…
↑	U	U	U	U	
→	U	U	U	U	
↓	U	U	U	U	
←	U	U	U	U	

Reward

	S1	S2	S3	S4	…
↑	R_{max}	R_{max}	R_{max}	R_{max}	
→	R_{max}	R_{max}	R_{max}	R_{max}	
↓	R_{max}	R_{max}	R_{max}	R_{max}	
←	R_{max}	R_{max}	R_{max}	R_{max}	

Transition Counts

	S1	S2	S3	S4	…
↑	0	0	0	0	
→	0	0	0	0	
↓	0	0	0	0	
←	0	0	0	0	

In the "known" MDP, any unknown (s,a) pair has its dynamics set as a self loop & reward = Rmax

R-max Algorithm

Plan in known MDP

R-max: Planning

- Compute optimal policy π_{known} for "known" MDP

Exercise: What Will Initial Value of Q(s,a) be for each (s,a) Pair in the Known MDP? What is the Policy?

Known/ Unknown

	S1	S2	S3	S4	…
↑	U	U	U	U	
→	U	U	U	U	
↓	U	U	U	U	
←	U	U	U	U	

Reward

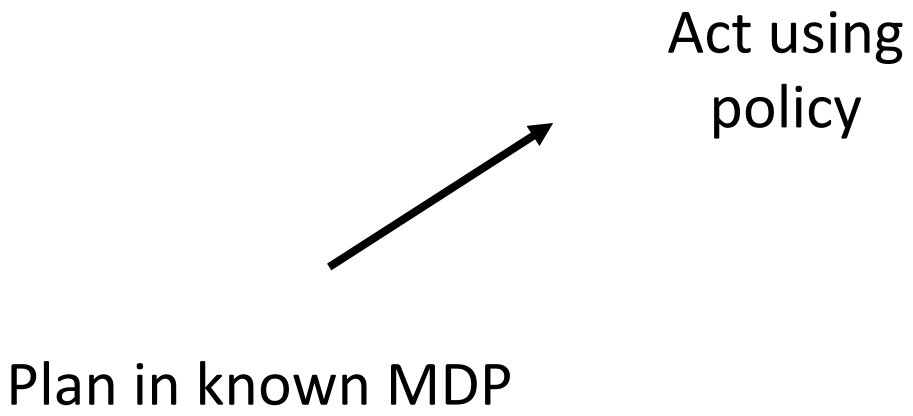
	S1	S2	S3	S4	…
↑	R_{max}	R_{max}	R_{max}	R_{max}	
→	R_{max}	R_{max}	R_{max}	R_{max}	
↓	R_{max}	R_{max}	R_{max}	R_{max}	
←	R_{max}	R_{max}	R_{max}	R_{max}	

Transition Counts

	S1	S2	S3	S4	…
↑	0	0	0	0	
→	0	0	0	0	
↓	0	0	0	0	
←	0	0	0	0	

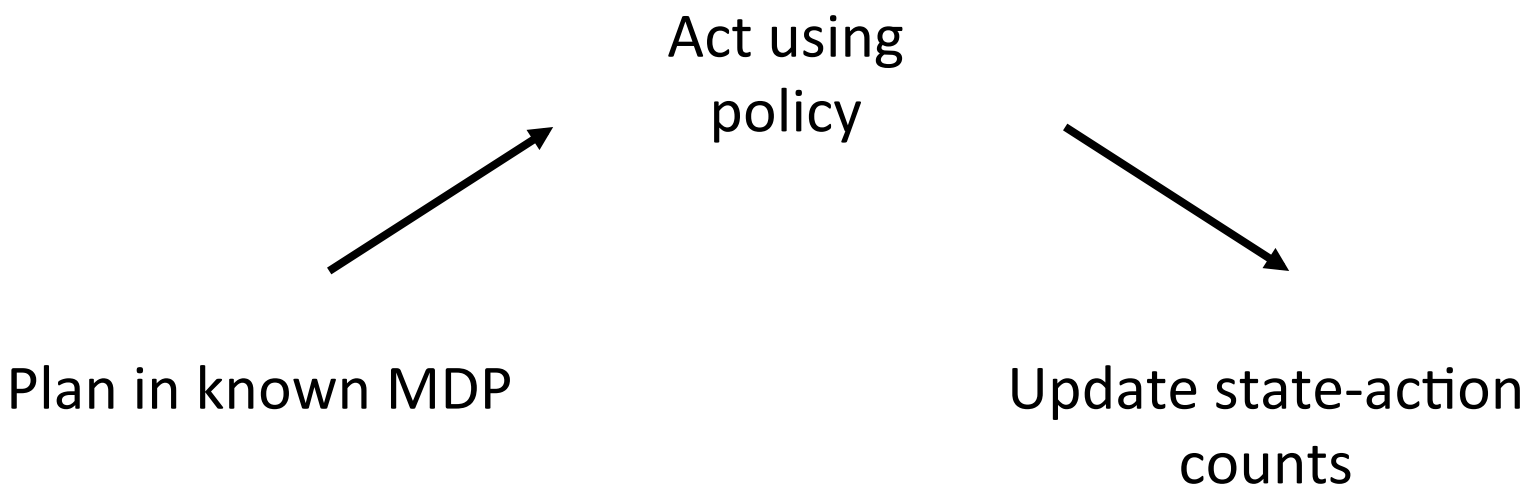
In the "known" MDP, any unknown (s,a) pair has its dynamics set as a self loop & reward = Rmax

R-max Algorithm



- Given optimal policy π_{known} for "known" MDP
- Take best action for current state $\pi_{known}(s)$, transition to new state s' and get reward r

R-max Algorithm



Update Known MDP

Known/
Unknown

	S2	S2	S3	S4	…
↑	U	U	U	U	
→	U	U	U	U	
↓	U	U	U	U	
←	U	U	U	U	

Reward

	S2	S2	S3	S4	…
↑	R_{max}	R_{max}	R_{max}	R_{max}	
→	R_{max}	R_{max}	R_{max}	R_{max}	
↓	R_{max}	R_{max}	R_{max}	R_{max}	
←	R_{max}	R_{max}	R_{max}	R_{max}	

Transition
Counts

	S2	S2	S3	S4	…
↑	0	0	0	0	
→	0	0	**1**	0	
↓	0	0	0	0	
←	0	0	0	0	

Increment counts for
state-action tuple

Update Known MDP

Known/ Unknown

	S2	S2	S3	S4	…
↑	U	U	U	U	
→	U	U	**K**	U	
↓	U	U	U	U	
←	U	U	U	U	

Reward

	S2	S2	S3	S4	…
↑	R_{max}	R_{max}	R_{max}	R_{max}	
→	R_{max}	R_{max}	**R**	R_{max}	
↓	R_{max}	R_{max}	R_{max}	R_{max}	
←	R_{max}	R_{max}	R_{max}	R_{max}	

Transition Counts

	S2	S2	S3	S4	…
↑	3	3	4	3	
→	2	4	**5**	0	
↓	4	0	4	4	
←	2	2	4	1	

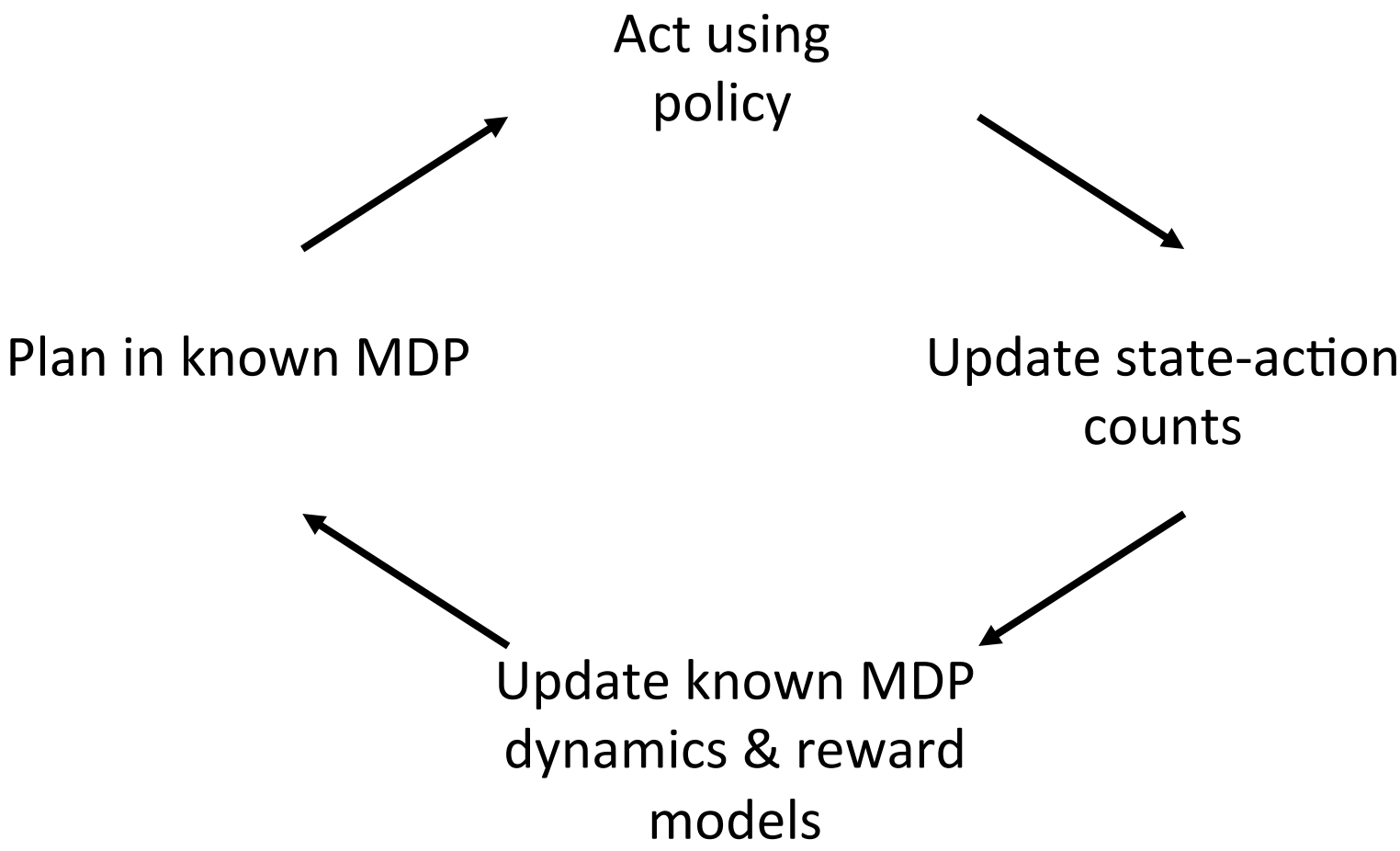
If counts for (s,a) > N, (s,a) becomes known: **use observed data to estimate transition & reward model** for (s,a) when planning

Estimating MDP Model for a (s,a) Pair Given Data

- Transition model estimation

- Reward model estimation

R-max Algorithm



R-max Behavior

Sample Complexity of R-max

$$\tilde{O}\left(\frac{SA}{\varepsilon(1-\gamma)^2}\underbrace{\frac{S}{\varepsilon^2(1-\gamma)^4}}\right)$$

samples need per (s,a) pair

On all but the above number of steps, chooses action whose expected reward is close to expected reward of action take if knew model parameters, with high probability

Common Idea of Model-Based PAC RL Algorithms:

Quantify how **many samples** do we need to build a **good model** that we can use to **act well** in the world?

“Good” RL Models

- Estimate model parameters from experience
- More experience means our estimated model parameters will closer be to the true unknown parameters, with high probability

Acting Well in the World

$p(s' \mid s,a)$ known → Compute ε-optimal policy

Bound $\left(\tilde{p}(s' \mid s,a) - p(s' \mid s,a)\right)$ → Bound error in policy calculated using $\tilde{p}(s' \mid s,a)$

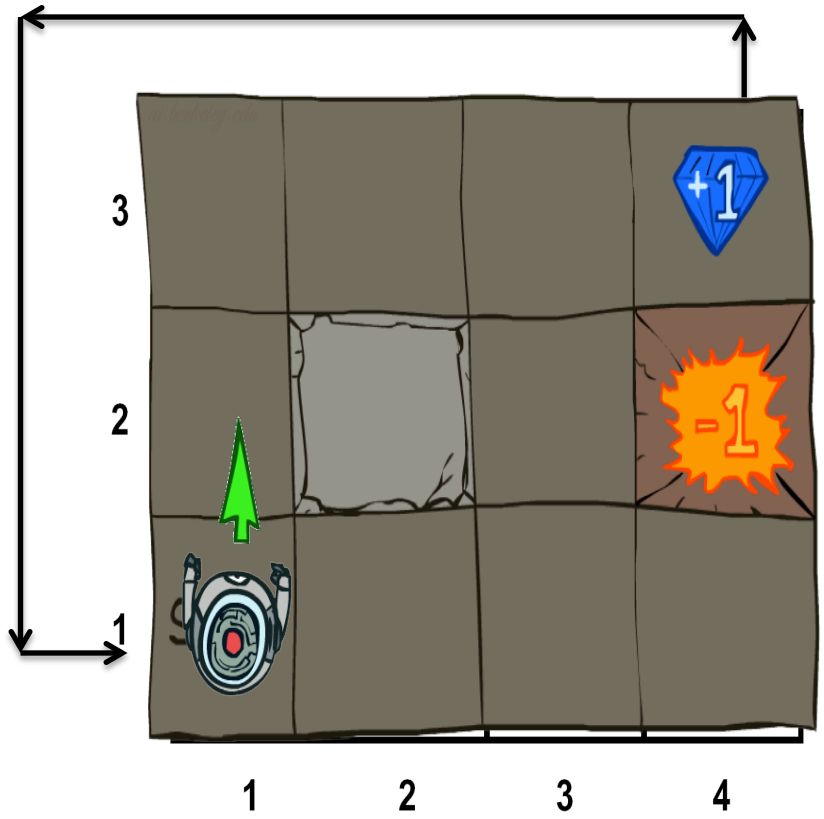
How many samples do we need to build a good model that we can use to act well in the world?

Sample complexity = # steps on which may not act well (could be far from optimal)

(R-MAX and E^3) = Poly(# of states)

Is this a small number of steps?

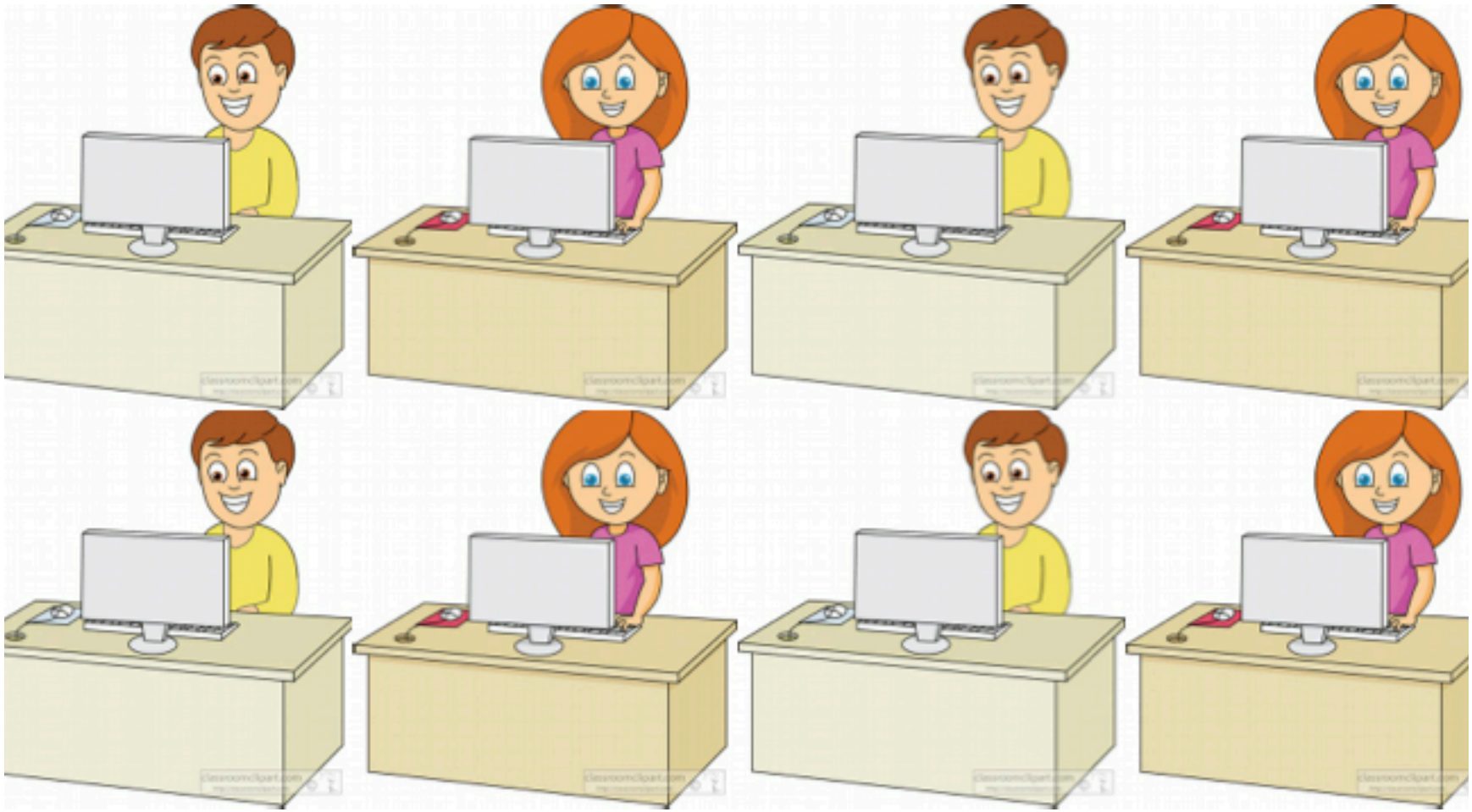
$$\tilde{O}\left(\frac{SA}{\varepsilon(1-\gamma)^2}\frac{S}{\varepsilon^2(1-\gamma)^4}\right)$$



γ=.99, ε=.05

Concurrent RL

(Guo and Brunskill 2015)



Other examples: all customers using Amazon, or patients in a hospital, …

What You Should Know About RL

- Define
  - Exploration vs exploitation tradeoff
  - Model free and model based RL
- Implement Q-learning and R-max
- Describe evaluation criteria for RL algorithms
  - empirical performance, computational complexity sample/data efficiency
  - Contrast strengths & weaknesses of Q-learning vs R-max in terms of these criteria. Give examples of where each might be prefered

- AI in the real world: Flappy Bird

- http://sarvagyavaish.github.io/FlappyBirdRL/

Real Objective