Last Time

Last Time

- What tools do we have to solve MDPs with continuous S and A?

Value Iteration – Function Approximation
LQR
Policy Search – Cross Entropy
Model Predictive Control

Course Map

Course Map

Value Function

- Outcome Uncertainty, Immediate vs Future Rewards (MDP)

Course Map

- Outcome Uncertainty, Immediate vs Future Rewards (MDP)
- Model Uncertainty (Reinforcement Learning)

Course Map

- Outcome Uncertainty, Immediate vs Future Rewards (MDP)
- Model Uncertainty (Reinforcement Learning)
- State Uncertainty (POMDP)

Course Map

- Outcome Uncertainty, Immediate vs Future Rewards (MDP)
- Model Uncertainty (Reinforcement Learning)
- State Uncertainty (POMDP)
- Interaction Uncertainty (Game)

Course Map

- Outcome Uncertainty, Immediate vs Future Rewards (MDP)
- Model Uncertainty (Reinforcement Learning)
- State Uncertainty (POMDP)
- Interaction Uncertainty (Game)

Course Map

- Outcome Uncertainty, Immediate vs Future Rewards (MDP)
- Model Uncertainty (Reinforcement Learning)
- State Uncertainty (POMDP)
- Interaction Uncertainty (Game)

Guiding Questions

Guiding Questions

- What is Reinforcement Learning?
- What are the main challenges in Reinforcement Learning?

Guiding Questions

- What is Reinforcement Learning?
- What are the main challenges in Reinforcement Learning?
- How do we categorize RL approaches?

Reinforcement Learning

~~In python, typically~~

~~`s, r = step(env, a)`~~

Reinforcement Learning

Previously: (S, A, T, R, γ)

In python, typically

```
s, r = step(env, a)
```

Reinforcement Learning

Previously: $(S, A, \cancel{T}, \cancel{R}, \gamma)$

Unknown!

In python, typically

```
s, r = step(env, a)
```

Reinforcement Learning

Previously: $(S, A, \cancel{T}, \cancel{R}, \gamma)$

Unknown!

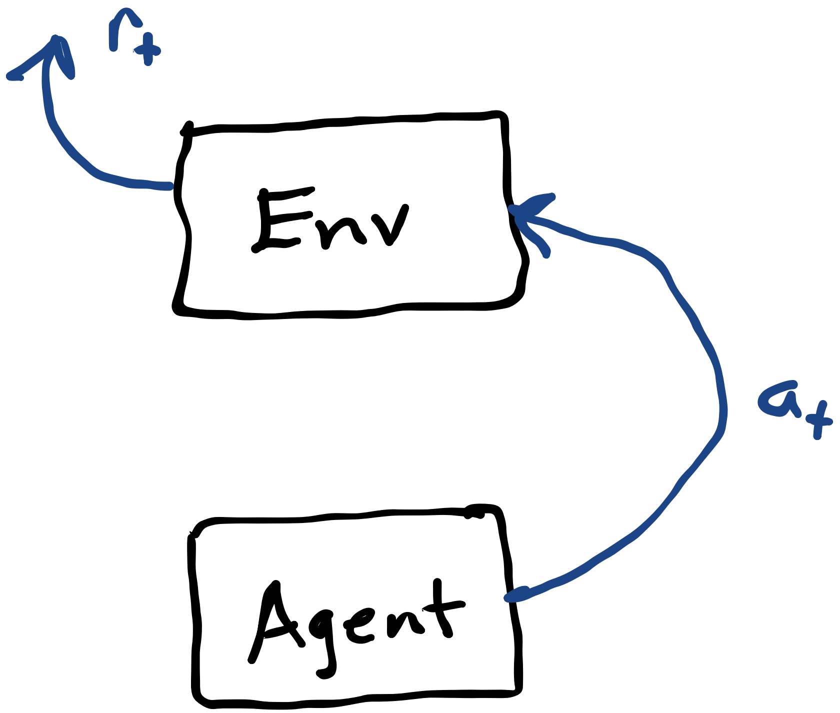
In python, typically

```
s, r = step(env, a)
```

Reinforcement Learning

Previously: $(S, A, \cancel{T}, \cancel{R}, \gamma)$

Unknown!



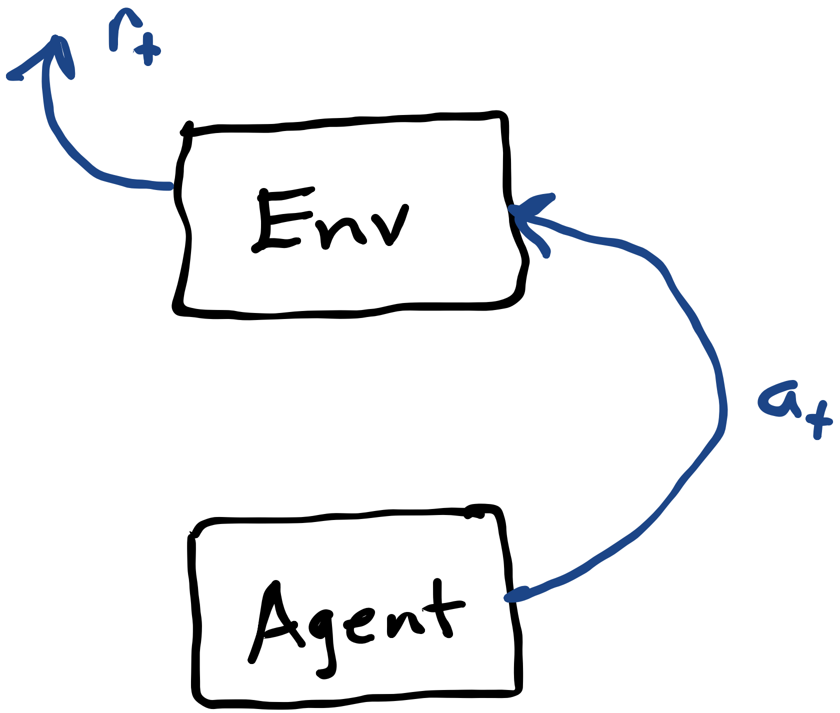
In python, typically

```
s, r = step(env, a)
```

Reinforcement Learning

Previously: $(S, A, \cancel{T}, \cancel{R}, \gamma)$

Unknown!




```
r = act!(env, a)
```

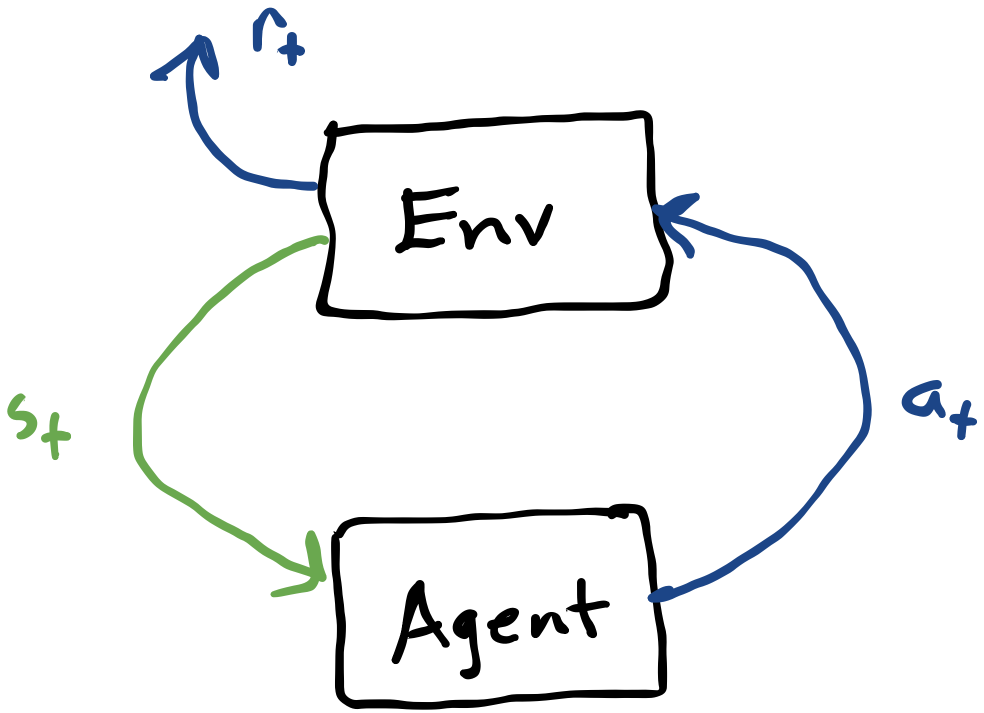
In python, typically

```
s, r = step(env, a)
```

Reinforcement Learning

Previously: $(S, A, \cancel{T}, \cancel{R}, \gamma)$

Unknown!




```
r = act!(env, a)
```

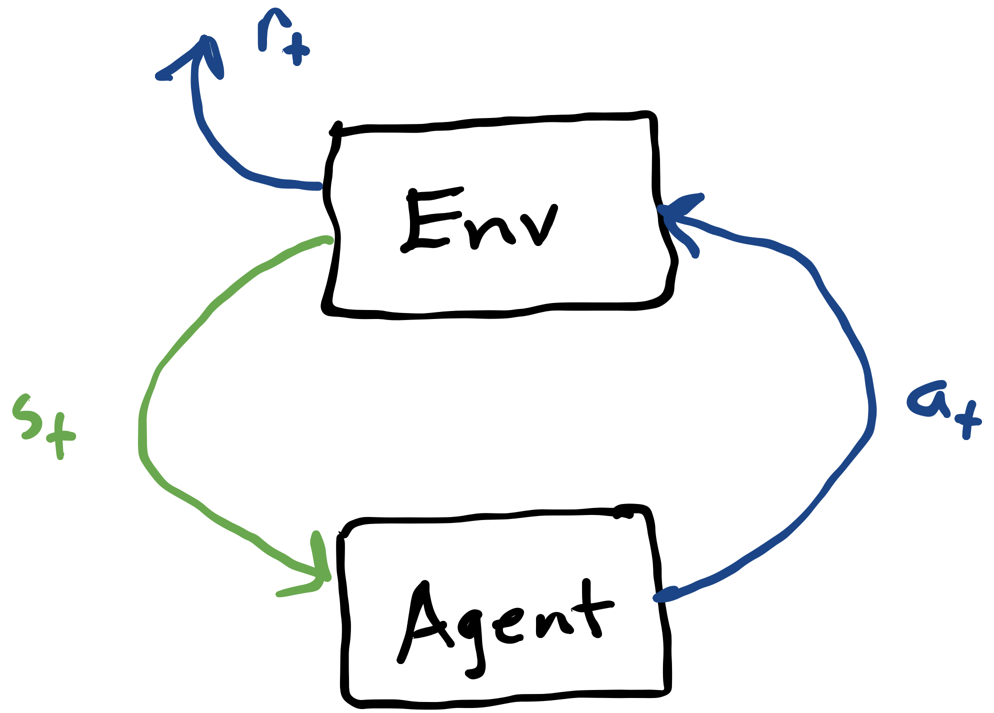
In python, typically

```
s, r = step(env, a)
```

Reinforcement Learning

Previously: $(S, A, \cancel{T}, \cancel{R}, \gamma)$

Unknown!




```
r = act!(env, a)

s = observe(env)
```

In python, typically

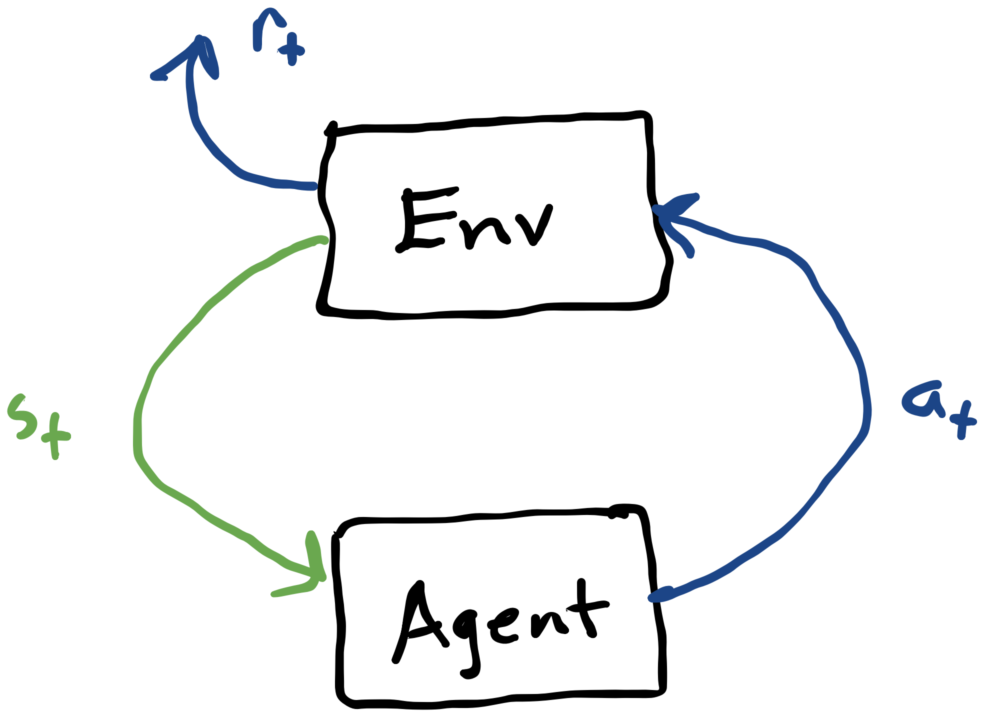
~~s, r = step(env, a)~~

s, r = env.step(a)

Reinforcement Learning

Previously: $(S, A, \cancel{T}, \cancel{R}, \gamma)$

Unknown!




```
r = act!(env, a)

s = observe(env)
```

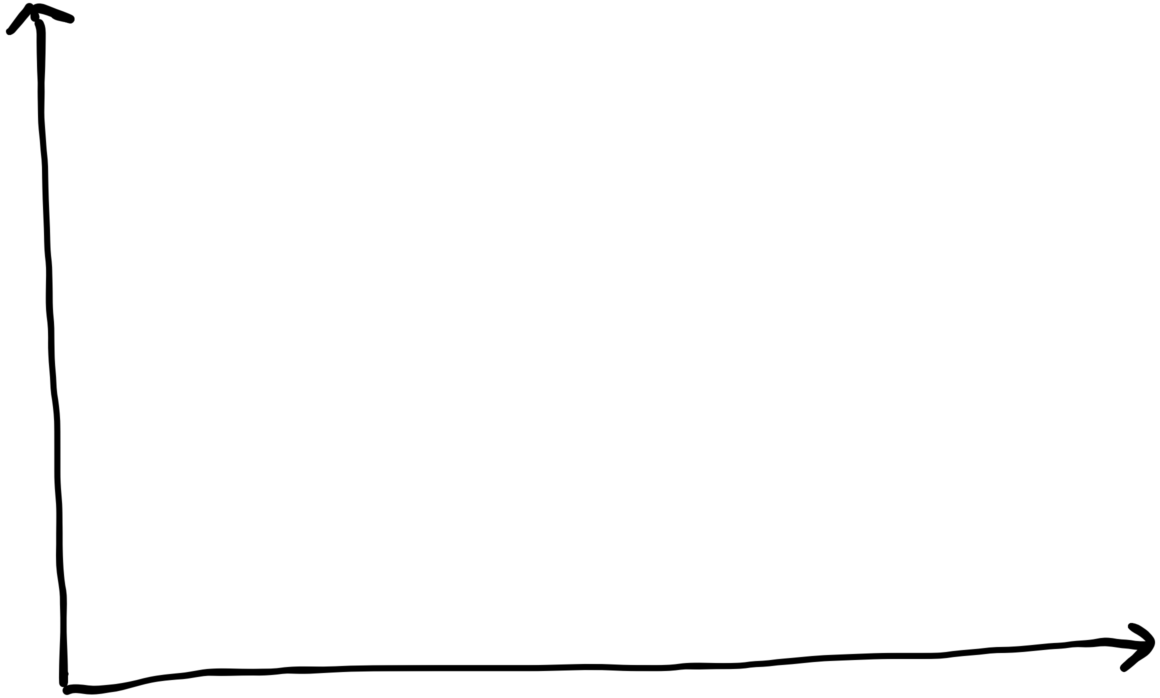
In python, typically

```
s, r = step(env, a)
```

Note: Different from $s', r = G(s, a)$

Learning Curve

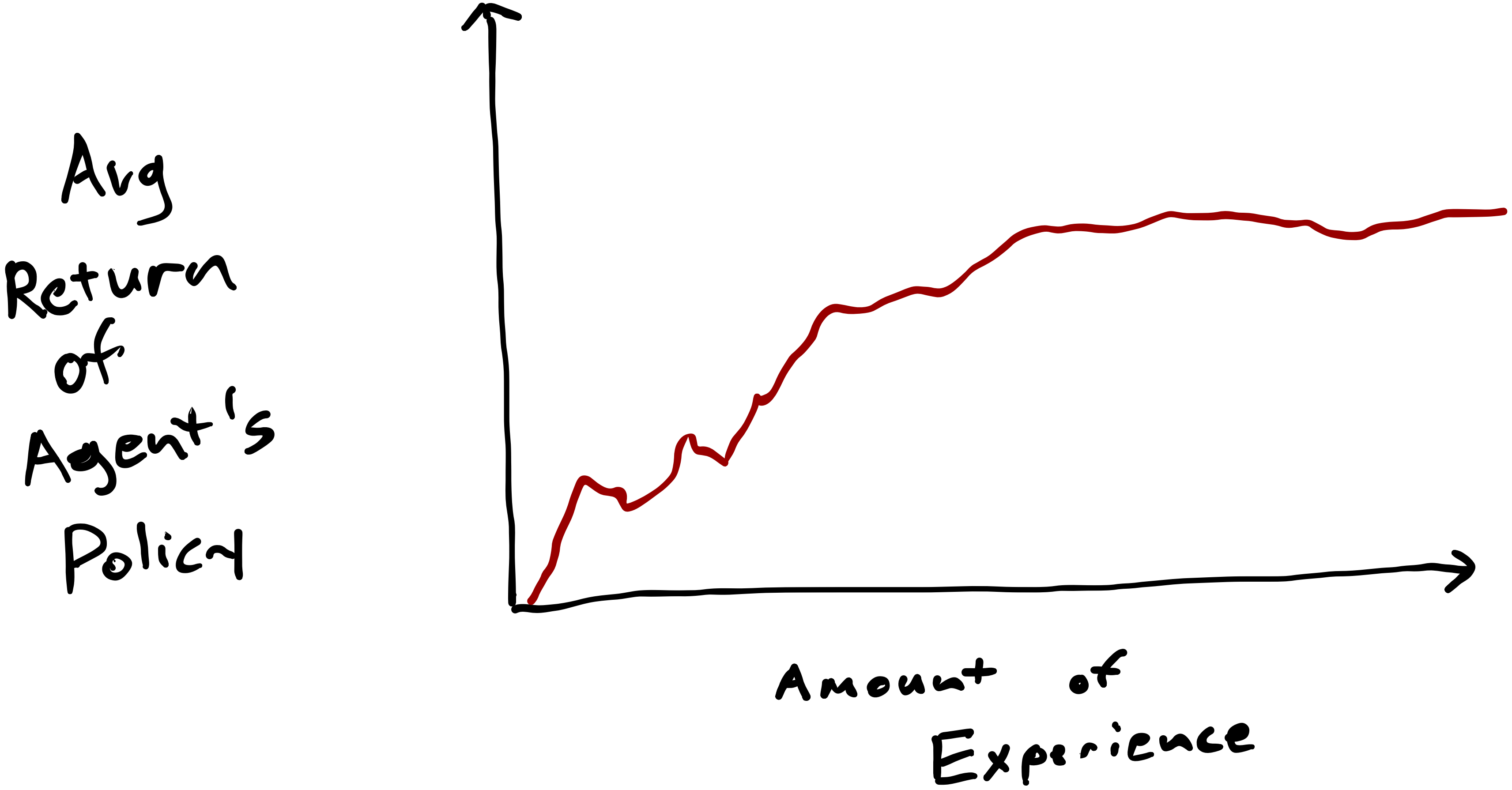
Learning Curve



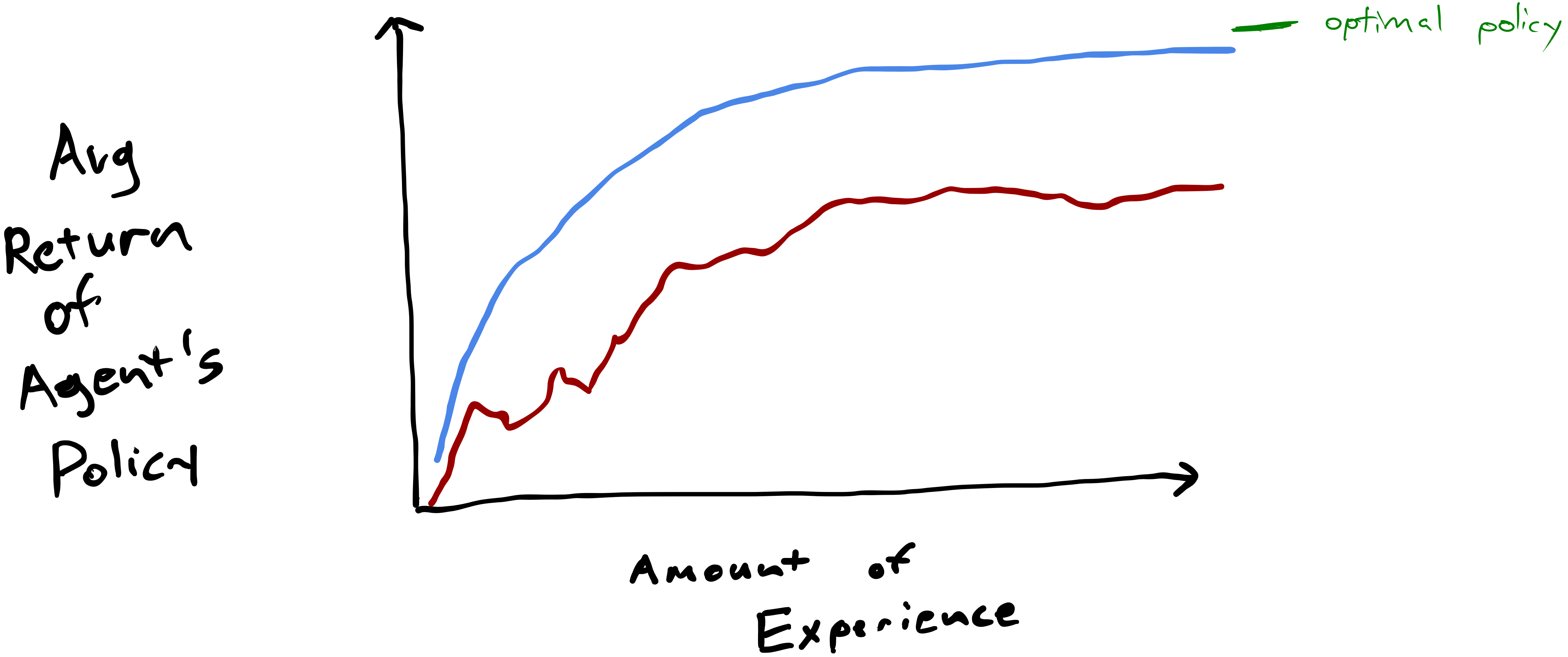
Learning Curve

Learning Curve

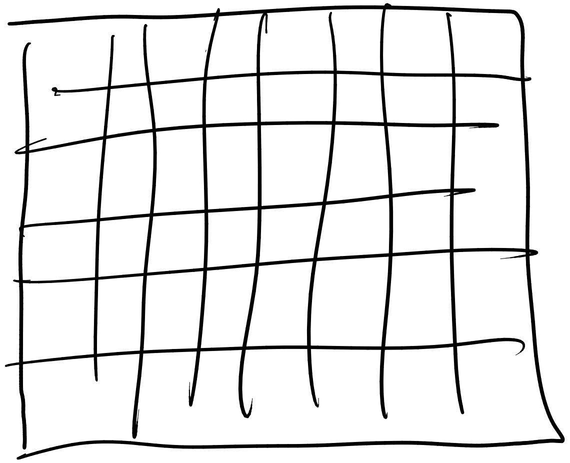
Learning Curve



Learning Curve



Breakout Rooms



T ?

R ?

$-10 \le R(s,a) \le 10$

How should we approach this?

What are challenges?

Exploration

Exploitation

1) MC-based equal prob actions
   log rewards, transition
   value iteration

2) ~~try all actions~~, estimate rewards + transitions, update policy every 100 steps

3) Cross Entropy Policy Search

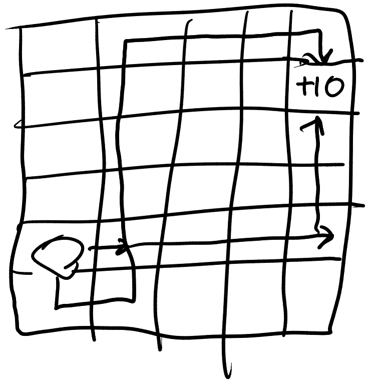
Challenges

Challenges

1. Exploration vs Exploitation

Challenges

1. Exploration vs Exploitation
2. Credit Assignment



Challenges

1. Exploration vs Exploitation
2. Credit Assignment
3. Generalization

Classifications

Classifications

- **Model Based**: Attempt to learn T and R, then find π^* by solving MDP

Classifications

1 2

- **Model Based**: Attempt to learn T and R, then find π^* by solving MDP
- **Model Free**: Attempt to find Q^* or π^* directly without estimating T or R

3

Classifications

- **Model Based**: Attempt to learn T and R, then find π^* by solving MDP
- **Model Free**: Attempt to find Q^* or π^* directly without estimating T or R

- **On-Policy**: Learn only using experience generated with the current policy.

Classifications

- **Model Based**: Attempt to learn T and R, then find π^* by solving MDP
- **Model Free**: Attempt to find Q^* or π^* directly without estimating T or R

- **On-Policy**: Learn only using experience generated with the current policy.
- **Off-Policy**: Learn using experience generated from the current policy *and* previous policies.

Classifications

- **Model Based**: Attempt to learn T and R, then find π^* by solving MDP
- **Model Free**: Attempt to find Q^* or π^* directly without estimating T or R

- **On-Policy**: Learn only using experience generated with the current policy.
- **Off-Policy**: Learn using experience generated from the current policy *and* previous policies.
- **Batch**: Learn only from previously-generated experience.

Classifications

- **Model Based**: Attempt to learn T and R, then find π^* by solving MDP
- **Model Free**: Attempt to find Q^* or π^* directly without estimating T or R

- **On-Policy**: Learn only using experience generated with the current policy.
- **Off-Policy**: Learn using experience generated from the current policy *and* previous policies.
- **Batch**: Learn only from previously-generated experience.

- **Tabular**: Keep track of learned values for each state in a table

Classifications

- **Model Based**: Attempt to learn T and R, then find π^* by solving MDP
- **Model Free**: Attempt to find Q^* or π^* directly without estimating T or R

$\pi_\theta \xrightarrow{\theta} \pi_{\theta'}$

- **On-Policy**: Learn only using experience generated with the current policy.
- **Off-Policy**: Learn using experience generated from the current policy *and* previous policies.

$\pi_{\theta_t} \xrightarrow{\theta_{t-1,t-2,\dots}} \pi_{\theta_{t+1}}$

- **Batch**: Learn only from previously-generated experience.

- **Tabular**: Keep track of learned values for each state in a table
- **Deep**: Use a neural network to approximate learned values

Tabular Maximum Likelihood Model-Based RL

given env, $|S|$, $|A|$

$N \in \mathbb{N}^{|S| \times |A| \times |S|}$

$\rho \in \mathbb{R}^{|S| \times |A|}$

$N[s, a, s']$

$\rho[s, a]$ = cumulative reward

$N \leftarrow 0$
$\rho \leftarrow 0$
$s \leftarrow$ observe(env)
$\pi \leftarrow$ random policy
loop

$a \leftarrow \begin{cases} \text{rand}(A) & \text{w.p. } \varepsilon \\ \pi(s) & \text{otherwise} \end{cases}$ ← exploration ε-greedy

$r \leftarrow$ act!(env, a)
$s' \leftarrow$ observe(env)
$N[s, a, s']$ += 1
$\rho[s, a]$ += r

$T[a][s, s'] \leftarrow \dfrac{N[s, a, s']}{\sum_{s'} N[s, a, s']}$ $\forall$ s, a, s'

$R[s, a] \leftarrow \dfrac{\rho[s, a]}{\sum_{s'} N[s, a, s']}$ $\forall$ s, a

$\pi \leftarrow$ solve(T, R) ← expensive
$s \leftarrow s'$

SARSA

Guiding Questions

Guiding Questions

- What is Reinforcement Learning?
- What are the main challenges in Reinforcement Learning?

Guiding Questions

- What is Reinforcement Learning?
- What are the main challenges in Reinforcement Learning?
- How do we categorize RL approaches?