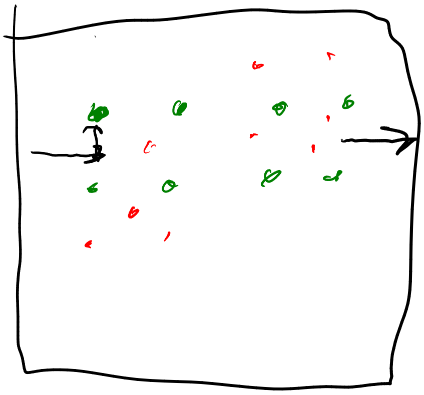
Reinforcement Learning



heuristic_pol(m, s)

for a

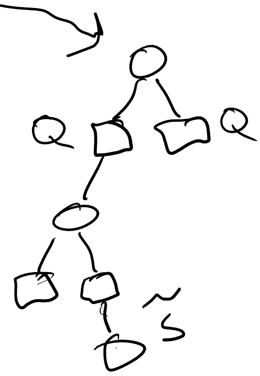
s', r ← @gen(sp, r)(m, sa)

rollout(m, select_action,)

for t

a = select_action(m, s)

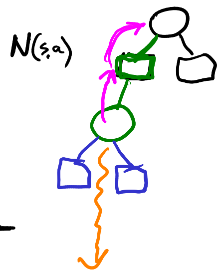
sp, r = @gen(:sp, :r)(m, s, a)



rollout(m, heuristic_policy, $\tilde{s}$)

Algorithm 4.9 Monte Carlo tree search

function SelectAction(s, d)
  loop
    Simulate(s, d, π_0)
  return $\arg\max_a Q(s,a)$
function Simulate(s, d, π_0)
  if $d = 0$
    return 0   U(s) / rollout
  if $s \notin T$
    for $a \in A(s)$
      $(N(s,a), Q(s,a)) \leftarrow (N_0(s,a), Q_0(s,a))$
    $T = T \cup \{s\}$
    return Rollout(s, d, π_0)
  $a \leftarrow \arg\max_{a \in A(s)} \left[Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \right]$
  $(s', r) \sim G(s,a)$
  $q \leftarrow r + \gamma$ Simulate($s', d-1, \pi_0$)
  $N(s,a) \leftarrow N(s,a) + 1$
  $Q(s,a) \leftarrow Q(s,a) + \frac{q - Q(s,a)}{N(s,a)}$
  return q



search

expansion

Value Estimate/rollout

Backup

Algorithm 4.10 Rollout evaluation

function Rollout(s, d, π_0)
  if $d = 0$
    return 0
  $a \sim \pi_0(s)$
  $(s', r) \sim G(s,a)$
  return $r + \gamma$ Rollout($s', d-1, \pi_0$)

simulate!(m, s, n, q, t=nothing)   (dict, dict)

γ^{100}

Last Time

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

- LQR $\pi^*(s) = -Ks$
- Sparse Sampling / PW in tree search
- Value Func. Approx.
- MPC

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?

Problem from HW2

Question 2. (25 pts) Consider a game with 3 squares in a horizontal line drawn on paper, a token, and a die. Each turn, the player can either reset or roll the die. If the player rolls and the die shows an odd number, the token is moved one square to the right, and if an even number is rolled, the token is moved two squares to the right (in both cases stopping at the rightmost square[1]). If the player resets, the token is always moved to the leftmost square. If the reset occurs when the token is in the middle square, two points are added; if the player resets when the token is on the right square, a point is subtracted.

c) Suppose you are not sure that the die is fair (i.e. whether it will yield odd and even with equal probability). Give finite upper and lower bounds for the accumulated discounted score that you can expect to receive with discount $\gamma = 0.95$.

Reinforcement Learning

Reinforcement Learning

Previously: (S, A, T, R, γ)

Reinforcement Learning

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

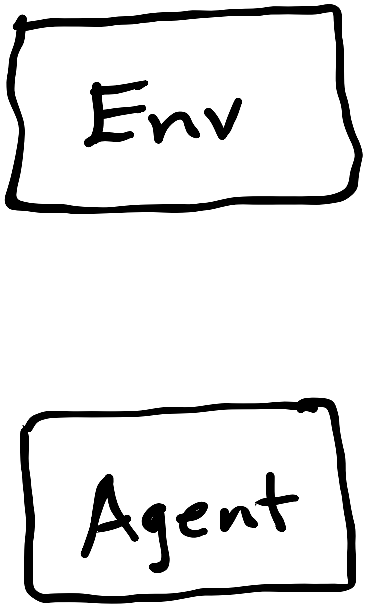
Unknown!

Reinforcement Learning

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

Unknown!

Now: Episodic Simulator

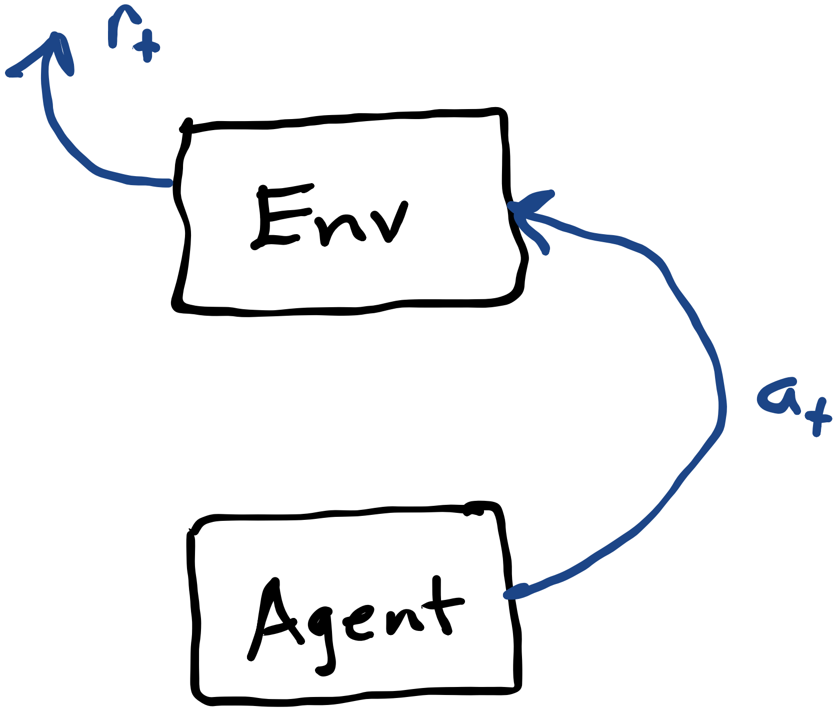


Reinforcement Learning

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

Unknown!

Now: Episodic Simulator



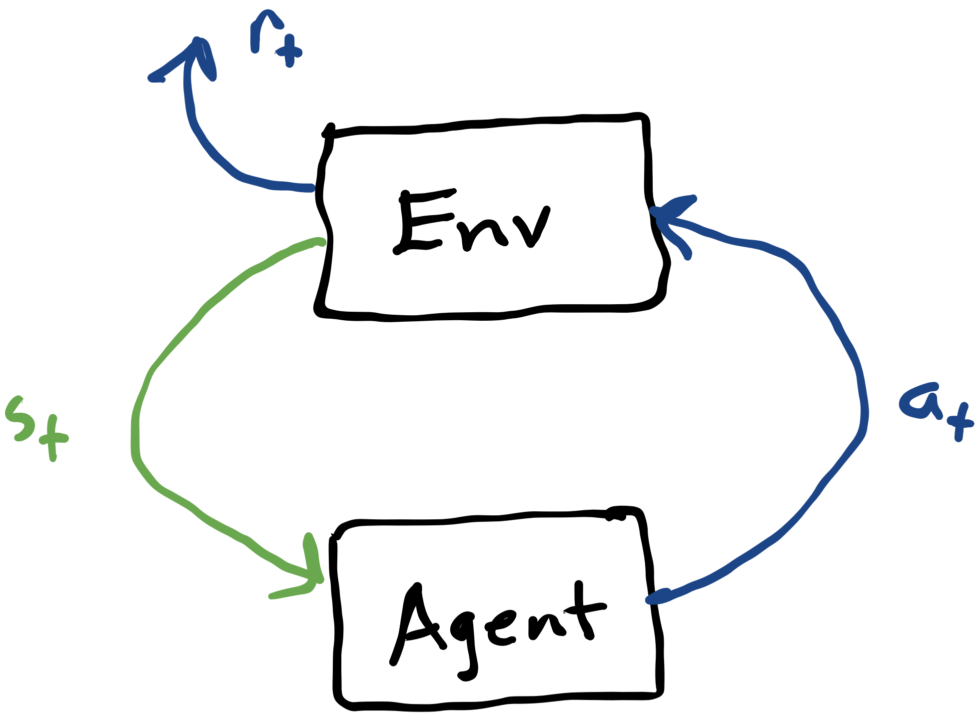

```
r = act!(env, a)
```

Reinforcement Learning

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

Unknown!

Now: Episodic Simulator



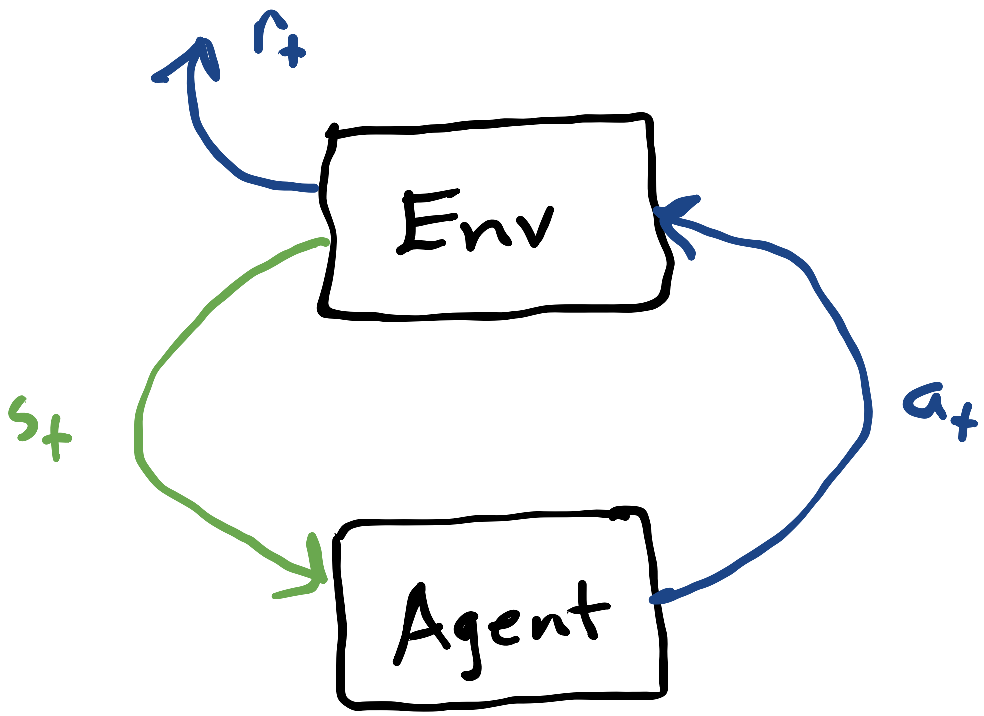

```
r = act!(env, a)

s = observe(env)
```

Reinforcement Learning

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

Unknown!

Now: Episodic Simulator




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

s = observe(env)
```

In python, typically

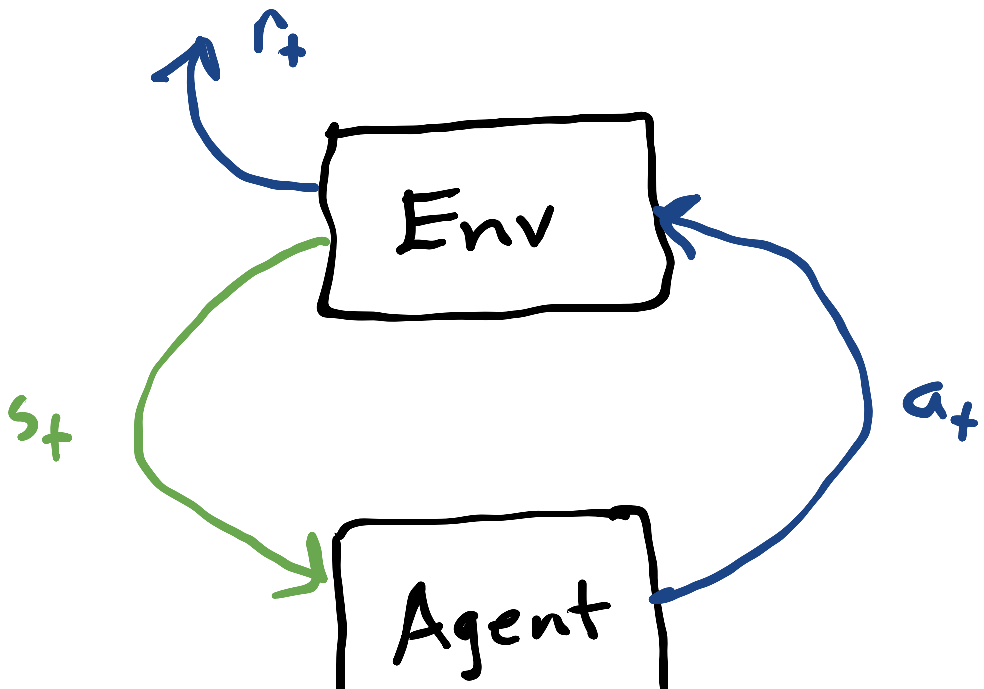
```
s, r = env.step(a)
```

Reinforcement Learning

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

Unknown!

Now: Episodic Simulator




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

s = observe(env)
```

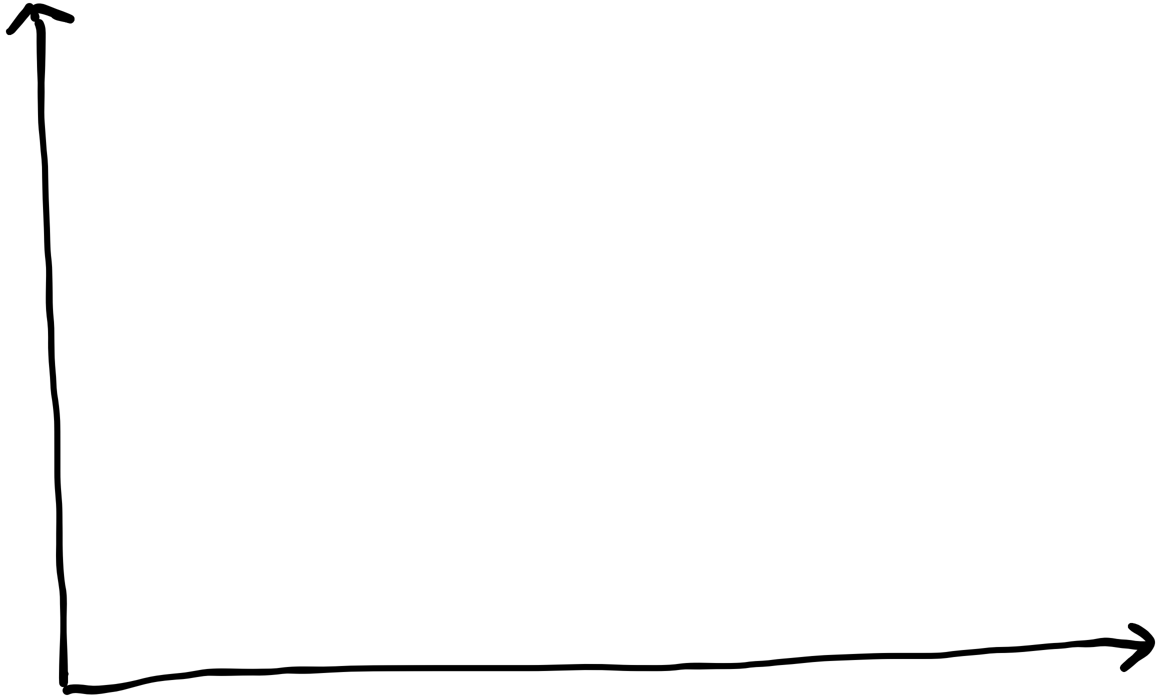
In python, typically

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

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

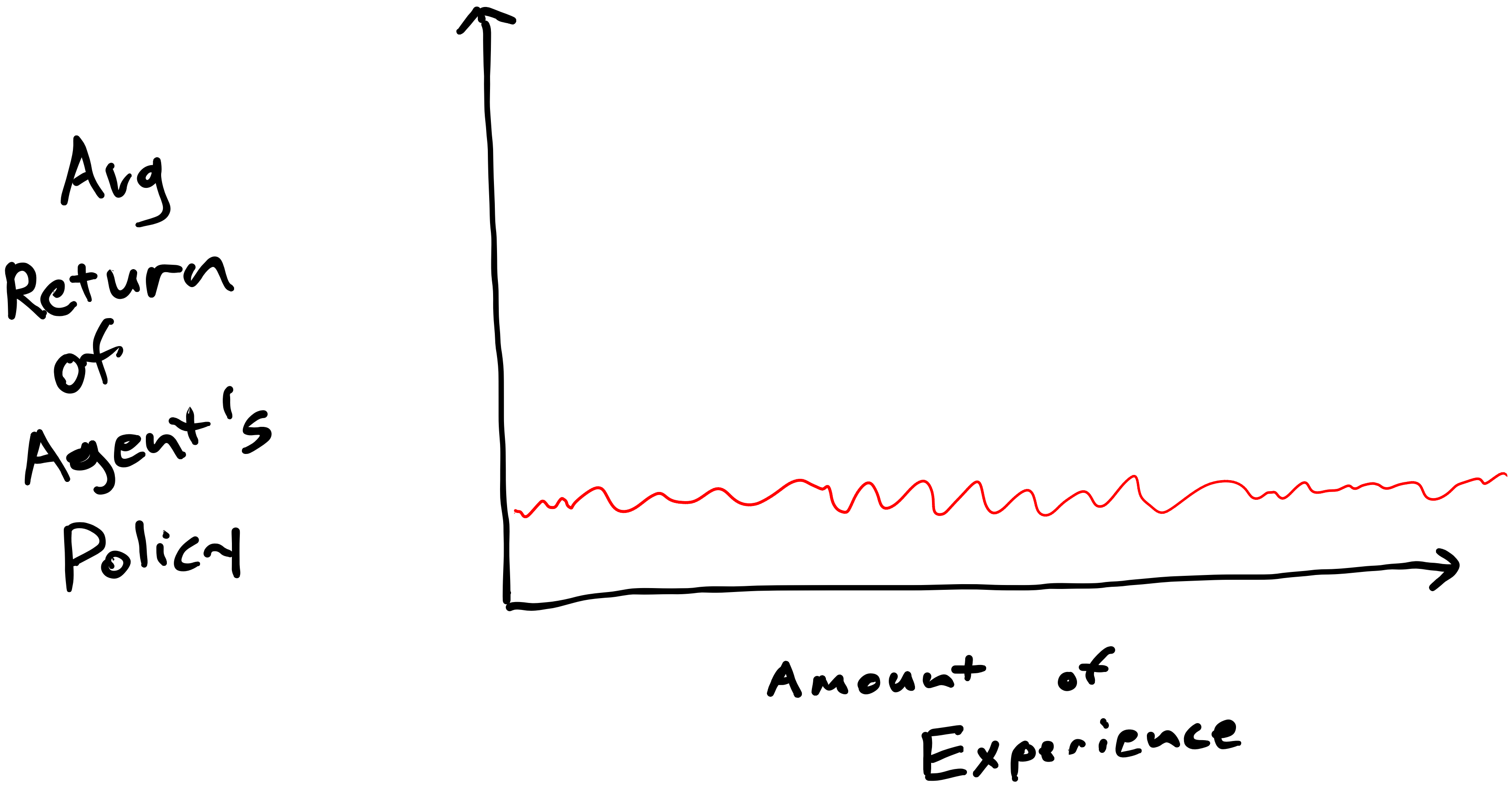
Learning Curve

Learning Curve

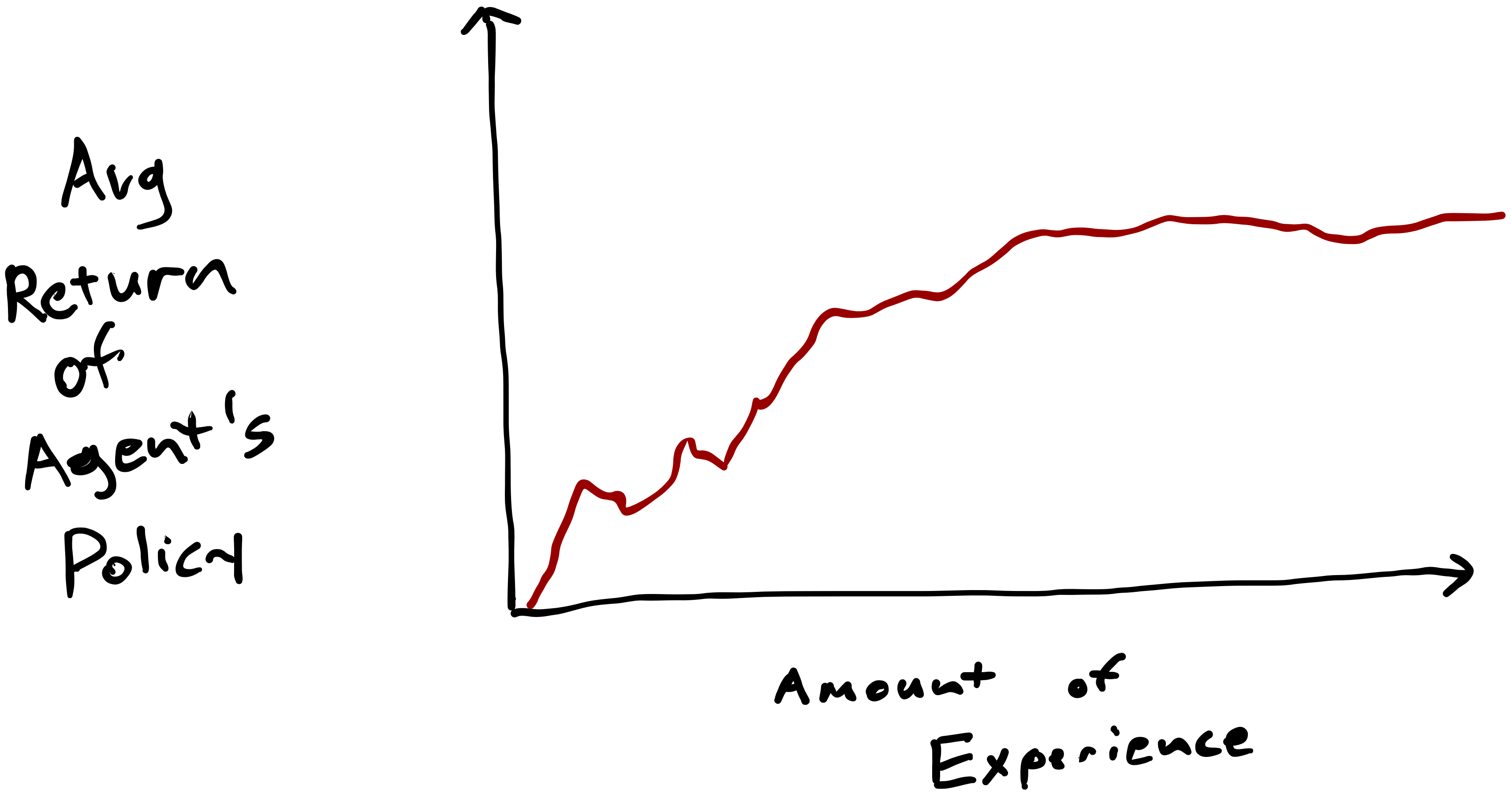


Learning Curve

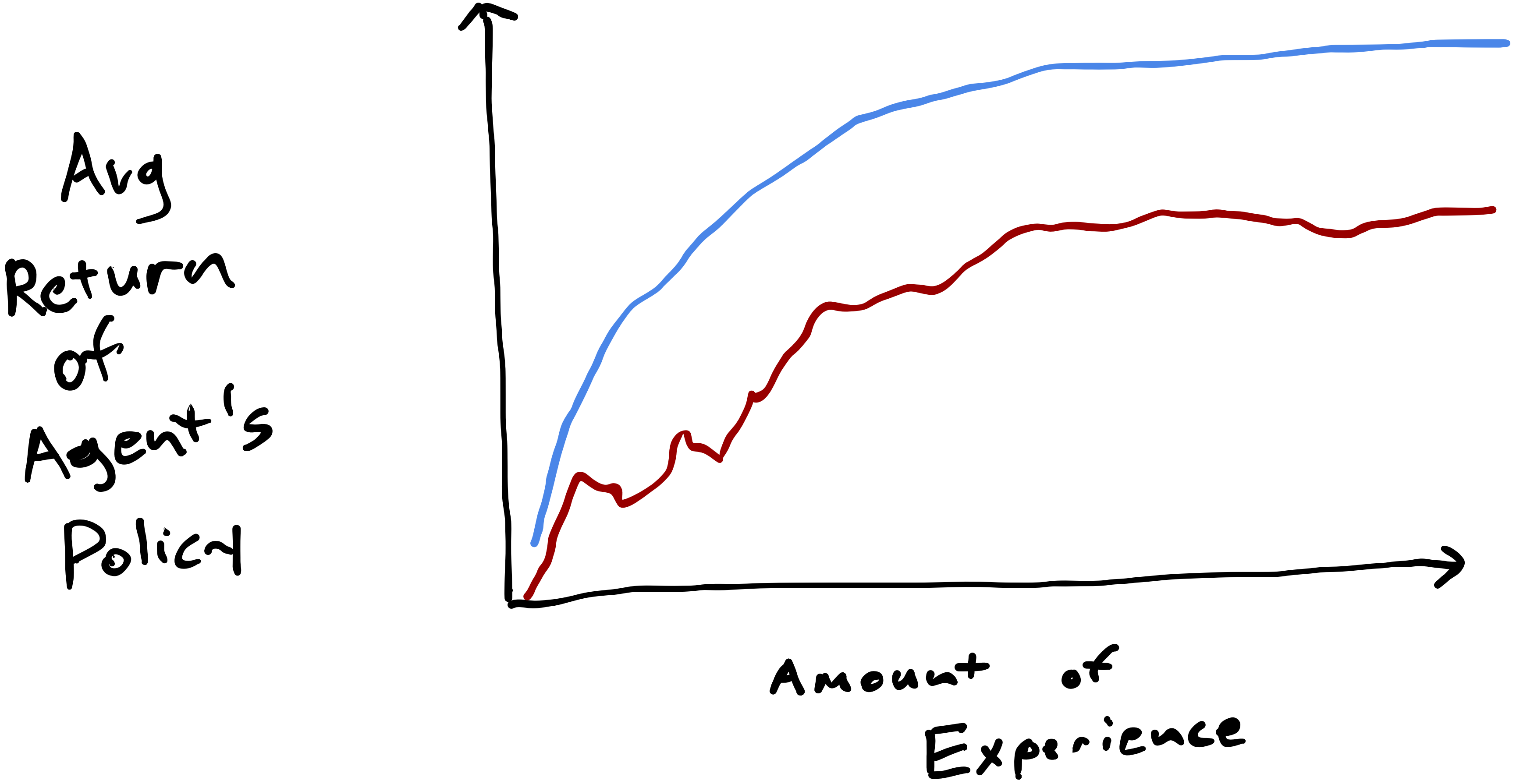
Learning Curve



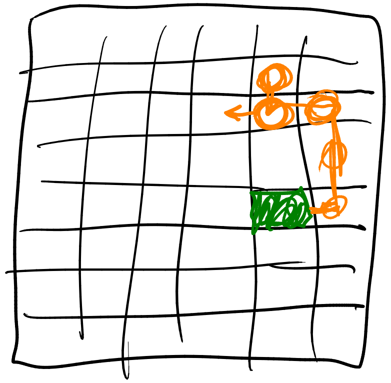
Learning Curve



Learning Curve



Break



know S, A

How should we interact with environment to maximize return given no knowledge of T and R.

model-based

$\hat{R}(s,a)$

$\hat{N}(s,a,s') \longrightarrow \hat{T}$

model-free

$\hat{U}(s)$

$\hat{Q}(s,a)$

Actor Critic

$\hat{\pi}(s)$

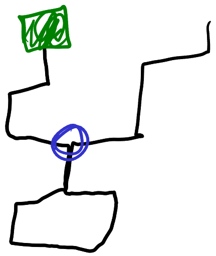
$$\pi(s) = \text{argmax}\left(\hat{R}(s,a) + \gamma \underset{\hat{T}}{E}\left[U(s)\right]\right)$$

$$\pi(s) = \underset{a}{\text{argmax}}\; \hat{Q}(s,a)$$

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

- **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

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

- **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
- **Deep**: Use a neural network to approximate learned values

Tabular Maximum Likelihood Model-Based RL

TMLMBRL

Given env, S, A

$N[s,a,s'] \leftarrow 0 \quad \forall s,a,s'$

$\rho[s,a] \leftarrow 0 \quad \forall s,a$

cumulative reward

$s \leftarrow$ observe(env)

$\pi \leftarrow$ random policy

loop

$a \leftarrow \begin{cases} \text{rand}(A) & \text{w.p. } \varepsilon \\ \pi(s) & \text{w.p. } 1-\varepsilon \end{cases}$

$r \leftarrow$ act!(env, a)

$s' \leftarrow$ observe(env)

$N[s,a,s'] \mathrel{+}= 1$

$\rho[s,a] \mathrel{+}= r$

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

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

expensive

$\pi \leftarrow$ solve(T, R, γ)

$s \leftarrow s'$

Guiding Questions

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