Reinforcement Learning: Off-Policy Policy Gradients (ESE 3990)

Instructor:
Antonio Loquercio

Fall 2026

University of Pennsylvania

Part 2: From Abstraction to Algorithms

- **Abstractions**:
  - MDPs and definitions.
  - Optimality of convergence with error that shrinks at some rate.
  - Bias & Variance Trade-off; Sample Efficiency.

- **Algorithms**:
  - Can my method control a robot in open-world environments?
  - Can my method learn to bias an LLM towards human preferences?
  - Can I use existing data (e.g., internet data) to bootstrap my learning?

Project

- Approximately half of your grade.
  - Individual project.

- Intermediate presentation in class on Nov 12
  - Present the idea, approach, and preliminary results
  - An opportunity for feedback.

- Final presentation in class on Dec 3.
  - Grade will be based on how confidently you answer my questions.

Project Ideas

- Train a robot to do something interesting.
- Train a portfolio manager.
- Analyze the difference between two algorithms on a set of toy problems.
- You can use AI coding tools to help, but you are responsible for everything.
- Your code will be inspected to verify that it matches the claims made.

Categorizing RL agents

Axis 1: How to make decisions

- Value Based
  - No Policy (Implicit)
  - Value/Q Function
- Policy Based
  - Policy
  - No Value/Q Function
- Actor Critic
  - Policy
  - Value Function

Today

Axis 2: Environment Representation

- Model-based
  - Policy and/or Value/Q Function
  - Model
- Model-free
  - Policy and/or Value/Q Function
  - No Model

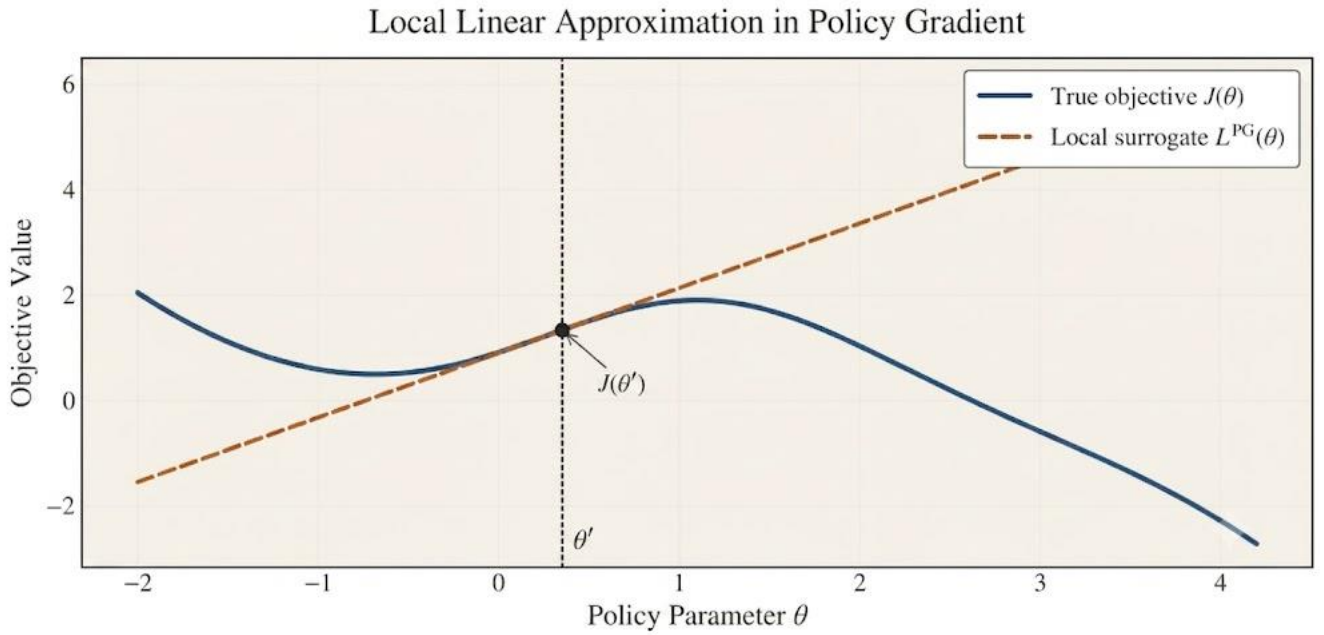
Reinforcement Learning by Policy Gradients

- True Objective

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[R(\tau)]$$

- Local Surrogate Objective
- $L(\theta) = \sum_{t=0}^{T-1}(\log \pi(a_t|s_t,\theta)\, G_t)$

- The local surrogate and true objective have the same gradient at the parameters θ' of the policy that collected the trajectories.



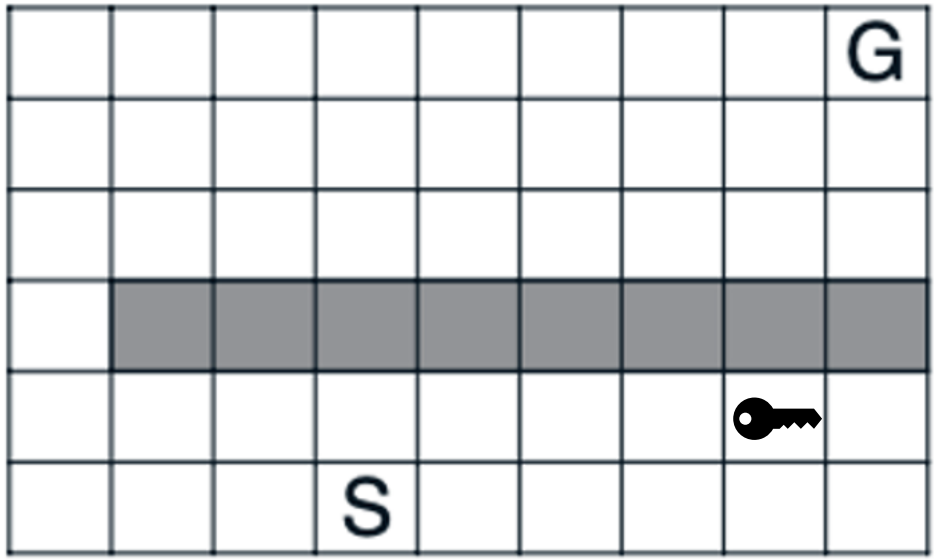
Advantage Actor-Critic (Revised)

- Init θ at random
- Run policy $\pi_\theta(a_t|s_t)$ and collect an episode in the environment.
- For each transition $(s_t, a_t, r_t, s_{t'})$
  - Compute the (bootstrapped) return G_t^λ
  - Compute the advantage $A_t = G_t^\lambda - \hat{V}(s_t)$

  Can do these two simultaneously with Generalized Advantage Estimation
- Compute the policy gradient estimate:
  - $\hat{g} = \frac{1}{T-1} \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) A_t$
- Update policy parameters
  - $\theta \leftarrow \theta + \alpha \hat{g}$
- Update the approximation of the value function

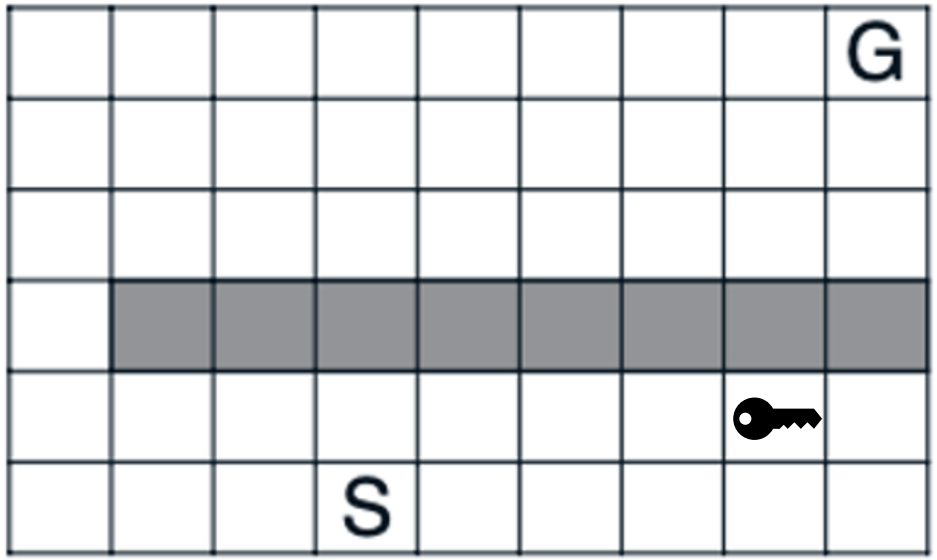
A2C is terribly inefficient

- Collect the key first.
- Then open the door to the goal.
- 100 reward upon reaching the goal.
- 0 otherwise.



A2C is terribly inefficient

- A random policy will hit the goal very rarely
- That's the only time the gradient is non-zero
  - $\hat{g} = \frac{1}{T-1} \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) A_t$
- After finally hitting the goal, A2C performs one gradient step.
- And then discards the trajectory.



A2C Cannot Use Other Sources of Data

- You have a lot of data from human drivers.
- You cannot use this data for policy learning.
- But it would be very useful to avoid catastrophic forgetting.



What we want

- Init θ at random
- Run policy $\pi_\theta(a_t|s_t)$ and collect an episode in the environment. Add data in the replay buffer D (can contain other data)
- For a few epochs:
  - Sample a batch of transitions $\{(s_t, a_t, r_t, s_{t'})^k\}$ from the replay buffer
  - For each transition $(s_t, a_t, r_t, s_{t'})^k$
    - Compute the (bootstrapped) return G_t^λ
    - Compute the advantage $A_t^k = G_t^\lambda - \hat{V}(s_t)$
  - Compute the policy gradient estimate: $\hat{g} \approx \frac{1}{K-1} \sum_k \nabla_\theta \log \pi_\theta\left(a_t^k \middle| s_t^k\right) A_t^k$
  - Update policy parameters: $\theta \leftarrow \theta + \alpha \hat{g}$
  - Update the approximation of the value function

The main problem

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\sum_{t=0}^{T-1} \left(\nabla_\theta \log \pi(a_t|s_t,\theta) \left(G_t - V^\pi(s_t) \right) \right) \right]$$

- I do a single gradient step: $\theta_{new} \leftarrow \theta_{\text{old}} + \alpha \hat{g}$
- If I reuse the same data for a second gradient step

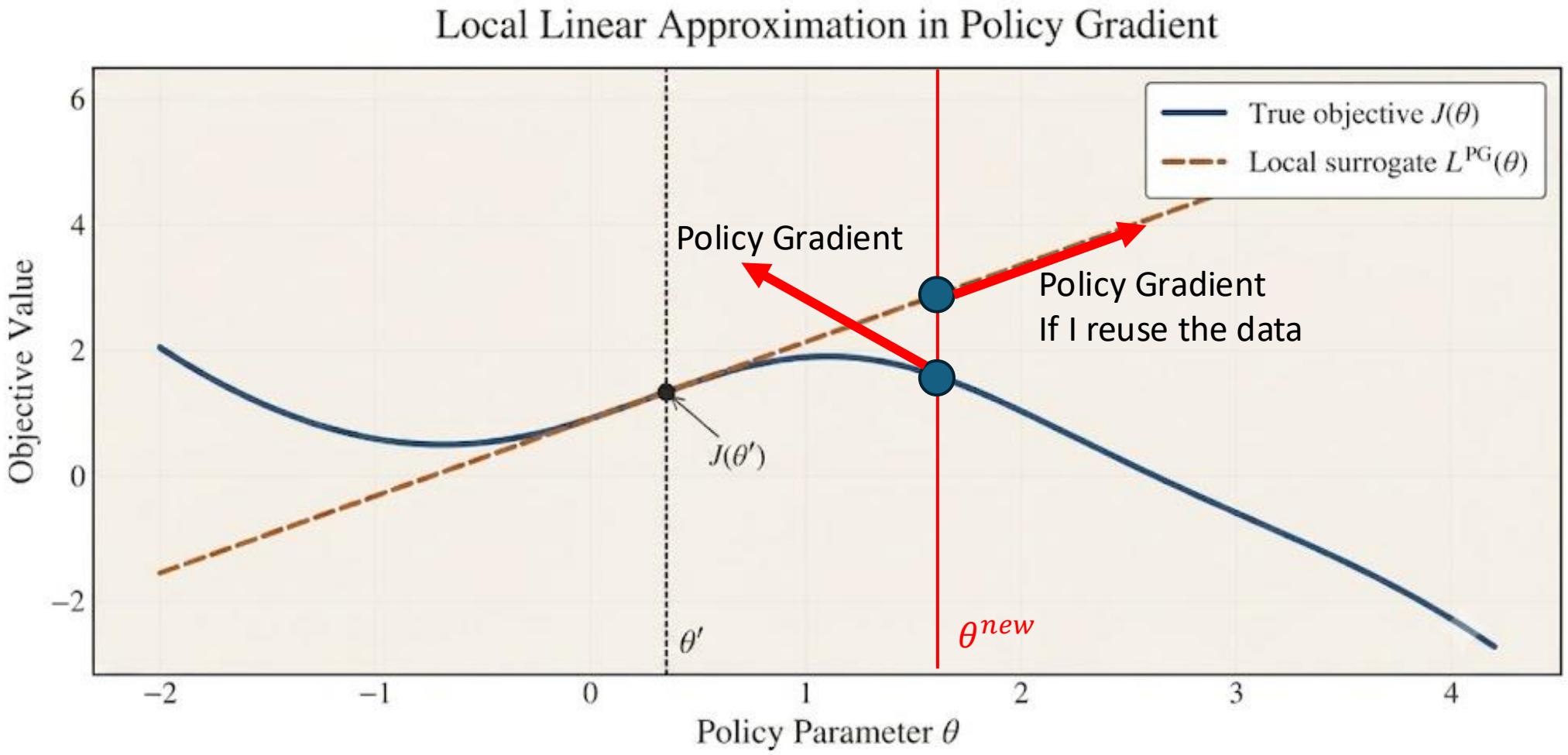
- $\nabla_\theta J(\theta^{new}) \neq \mathbb{E}_{\tau \sim p(\tau;\theta^{old})} \left[\sum_{t=0}^{T-1} \left(\nabla_\theta \log \pi(a_t|s_t,\theta^{new}) \left(G_t(\tau) - V^\pi(s_t) \right) \right) \right]$

The main problem

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\sum_{t=0}^{T-1} \left(\nabla_\theta \log \pi(a_t|s_t,\theta) \left(G_t - V^\pi(s_t) \right) \right) \right]$$

- As soon as I do a single gradient step: $\theta_{new} \leftarrow \theta_{\text{old}} + \alpha \hat{g}$
- I cannot do another step because:
  - The probability $\tau \sim p(\tau;\theta)$ changes
  - The (expected) returns G_t of the new policy changes

Reinforcement Learning by Policy Gradients



Importance Sampling To The Rescue

- Estimate the expectation of a different distribution

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(X) \\ &= \sum Q(X)\frac{P(X)}{Q(X)}f(X) \\ &= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]\end{aligned}$$

Estimating The Gradient with IS

- $\nabla_\theta J(\theta^{\text{new}}) = \mathbb{E}_{\tau \sim p(\tau;\theta^{\text{new}})}\left[\sum_{t=0}^{T-1}\left(\nabla_\theta \log \pi(a_t|s_t,\theta^{\text{new}})\left(G_t(\tau) - V^\pi(s_t)\right)\right)\right]$

- $\nabla_\theta J(\theta^{\text{new}}) =$

$$\mathbb{E}_{\tau \sim p(\tau;\theta^{old})}\left[\frac{p(\tau;\theta^{new})}{p(\tau;\theta^{old})}\sum_{t=0}^{T-1}\left(\nabla_\theta \log \pi(a_t|s_t,\theta^{\text{new}})\left(G_t(\tau) - V^\pi(s_t)\right)\right)\right]$$

Estimating the Gradient with IS

$$p(\tau;\theta^{new}) = \pi(a_t|s_t,\theta^{new})p(s_{t+1}|a_t,s_t)\pi(a_{t+1}|s_{t+1},\theta^{new})\cdots$$

$$p\left(\tau;\theta^{old}\right) = \pi\left(a_t\middle|s_t,\theta^{old}\right)p(s_{t+1}|a_t,s_t)\pi\left(a_{t+1}\middle|s_{t+1},\theta^{old}\right)\cdots$$

The correction weight of each return is the ratio of action probabilities:

$$W = \frac{P(\tau;\pi)}{Q(\tau;\mu)} = \frac{\pi(a_t|s_t,\theta^{new})}{\pi\left(a_t\middle|s_t,\theta^{old}\right)}\frac{\pi(a_{t+1}|s_{t+1},\theta^{new})}{\pi\left(a_{t+1}\middle|s_{t+1},\theta^{\mathrm{old}}\right)}\cdots$$

- Since a policy cannot affect the MDP transition probabilities.

Estimating The Gradient with IS

- $\nabla_\theta J(\theta^{\text{new}}) = \mathbb{E}_{\tau\sim p(\tau;\theta^{\text{new}})}\left[\sum_{t=0}^{T-1}\left(\nabla_\theta \log\pi(a_t|s_t,\theta^{\text{new}})\left(G_t(\tau) - V^\pi(s_t)\right)\right)\right]$
- $\nabla_\theta J(\theta^{\text{new}}) =$

$$\mathbb{E}_{\tau\sim p(\tau;\theta^{old})}\left[\rho_{0:T-1}\sum_{t=0}^{T-1}\left(\nabla_\theta \log\pi(a_t|s_t,\theta^{\text{new}})\left(G_t(\tau) - V^\pi(s_t)\right)\right)\right]$$

- $\rho_{0:T-1} = \prod_{t'=0}^{T-1}\frac{\pi(a_{t'}|s_{t'},\theta^{new})}{\pi\left(a_{t'}\middle|s_{t'},\theta^{old}\right)}$

Can we do better?

- $\rho_{0:T-1} = \Pi_{t'=0}^{T-1} \frac{\pi(a_{t'}|s_{t'}, \theta^{new})}{\pi\left(a_{t'}\middle|s_{t'}, \theta^{old}\right)}$

- $\rho_{0:T-1}$ can be very large or very small
- It adds a lot of variance to the gradient

- We can improve it using some of the techniques we saw ☺

Let's make this better

- $\nabla_\theta J(\theta^{\text{new}}) =$

$$\mathbb{E}_{\tau\sim p(\tau;\theta^{old})}\left[\sum_{t=0}^{T-1}\left(\rho_{0:t}\nabla_\theta \log\pi(a_t|s_t,\theta^{\text{new}})\rho_{t:T-1}\left(G_t(\tau)-V^\pi(s_t)\right)\right)\right]$$

- $\nabla_\theta J(\theta^{\text{new}}) =$

$$\mathbb{E}_{\tau\sim p(\tau;\theta^{old})}\left[\sum_{t=0}^{T-1}\left(\rho_{0:t}\nabla_\theta \log\pi(a_t|s_t,\theta^{\text{new}})A^{\theta^{new}}\right)\right]$$

Reducing Variance with Per-decision Importance Sampling

- $\nabla_\theta J(\theta^{\text{new}}) =$

$$\mathbb{E}_{\tau \sim p(\tau;\theta^{old})}\left[\sum_{t=0}^{T-1}\left(\rho_{0:t}\nabla_\theta \log \pi(a_t|s_t,\theta^{\text{new}})\left(\sum_{t'=t}^{T-1}\gamma^{t'-t}\rho_{t:t'}r_{t'} - V^\rho(s_t)\right)\right)\right]$$

- Where:

$$V^\rho(s_t) = \rho_{t:T-1}V^{\theta^{old}}(s_t)$$

- We can use other techniques presented in the off-policy learning class

Reducing Variance in Off-Policy Policy Gradients

- The general form is the best we can do when using policy gradients on *any kind* of off-policy data.
- However, we can do better by *restricting* the type of off-policy data.
- Specifically, we're interested in the case where the replay buffer only contains data from the *current* policy.
- And we want to do multiple epochs on that same data.

Example

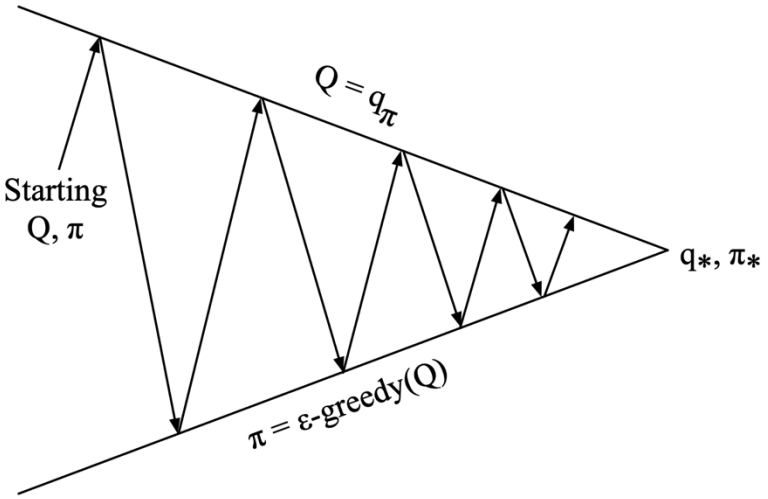
- I don’t use the data from human drivers.
- But I want to iterate many times on the data collected by the current policy.
- I can’t do this with A2C (data is “invalid” after a single gradient step).



The algorithm we want

- Init θ at random
- Run policy $\pi_\theta(a_t|s_t)$ and collect N episodes in the environment. Put them in a replay buffer D.
- For each transition $(s_t, a_t, r_t, s_{t'})^k$, compute the (bootstrapped) return G_t^λ and the advantage $A_t^k = G_t^\lambda - \hat{V}(s_t)$
- For a few epochs:
  - Sample a batch of K transitions $\{(s_t, a_t, r_t, s_{t'})^k\}$ from the replay buffer
  - Compute the policy gradient estimate: $\hat{g} \approx \frac{1}{K-1}\sum_k \nabla_\theta \log \pi_\theta\left(a_t^k \middle| s_t^k\right) A_t^k$
  - Update policy parameters: $\theta \leftarrow \theta + \alpha\hat{g}$
- Update the approximation of the value function
- Empty replay buffer

Reminder: Policy Iteration



- Model-free Policy Iteration Algorithm
  - Evaluate $Q^\pi(s, a)$
  - Set $\pi \leftarrow \pi'$ (using greedy actions)

- In policy iteration, we are using the Q value of the old policy π to find the new policy π'

- *What if we interpret off-policy policy gradients as a form of policy iteration?*

Policy Gradient as Policy Iteration

- Let's define $\theta^{old} = \theta; \theta^{new} = \theta'$
- Claim: $J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\theta}(s_t, a_t)\right] = \bar{A}(\theta')$

- Why do we want this?
  - $\theta' \leftarrow \arg\max_{\theta'} \bar{A}(\theta')$
  - $\bar{A}(\theta')$ uses only the advantage of the old policy.
  - However, $\bar{A}(\theta')$ still depends on the state distribution of the new policy (we will address this later)

Policy Gradient as Policy Iteration

- Claim: $J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\theta}(s_t, a_t)\right] = \bar{A}(\theta')$

- $J(\theta') - J(\theta) = J(\theta') - \mathbb{E}_{s_0 \sim p(s_0)}\left[V^{\theta}(s_0)\right]$
- $J(\theta') - \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[V^{\theta}(s_0)\right]$ (does not depend on τ)
- $J(\theta') - \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0} \gamma^t V^{\theta}(s_t) - \sum_{t=1} \gamma^t V^{\theta}(s_t)\right]$ (adding nothing)
- $J(\theta') + \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0} \gamma^t \left(\gamma V^{\theta}(s_{t+1}) - V^{\theta}(s_t)\right)\right]$ (rearrange terms)
- $\mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t r(s_t, a_t)\right] + \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0} \gamma^t \left(\gamma V^{\theta}(s_{t+1}) - V^{\theta}(s_t)\right)\right]$
- $\mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t \left(r(s_t, a_t) + \gamma V^{\theta}(s_{t+1}) - V^{\theta}(s_t)\right)\right]$
- $\mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\theta}(s_t, a_t)\right]$ (definition of advantage)

Policy Gradient As Policy Iteration

- $J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t \cancel{\gamma^t} A^{\theta}(s_t, a_t)\right] = \bar{A}(\theta')$

- This proof justifies using the old policy's advantage to compute the new policy.
- Since it does not matter *how* we get to θ' (one step or multiple steps)
- However, the expectation is on the state distribution of the new policy θ'
- This is a problem since we need to update the distribution after every gradient step.
- We will address this problem with importance sampling.

A useful identity

- $\mathbb{E}_{\tau \sim \mathrm{p}_\theta(\tau)}[\sum_{t=0}^{T-1} f(s_t, a_t)]$
- $= \sum_{t=0}^{T-1} \left[\mathbb{E}_{\tau \sim \mathrm{p}_\theta(\tau)}[f(s_t, a_t)] \right]$ (linearity of Exp.)
- $= \sum_{t=0}^{T-1} \left[\mathbb{E}_{\tau \sim (s_0, a_0, s_1, \dots)}[f(s_t, a_t)] \right]$
- $= \sum_{t=0}^{T-1} \left[\mathbb{E}_{\tau \sim (s_0, a_0, s_1, \dots, s_t, a_t)}[f(s_t, a_t)] \right]$ (law of total exp.)
- $= \sum_{t=0}^{T-1} \left[\mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)}[f(s_t, a_t)] \right]$ (holds for any joint distribution)

Importance Sampling on the State Distribution

- $J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_t A^{\theta}(s_t, a_t)\right]$
- $\sum_{t=0}^{T-1} \mathbb{E}_{s_t, a_t \sim p_{\theta'}(s_t, a_t)}\left[\sum_t A^{\theta}(s_t, a_t)\right]$ (from previous slide)

- Let's use importance sampling to change the distribution to $p_{\theta}(s_t, a_t)$
- $\sum_{t=0}^{T-1} \mathbb{E}_{s_t, a_t \sim p_{\theta'}(s_t, a_t)}\left[\sum_t A^{\theta}(s_t, a_t)\right] =$
- $\sum_{t=0}^{T-1} \mathbb{E}_{s_t, a_t \sim p_{\theta}(s_t, a_t)}\left[\frac{p_{\theta'}(s_t, a_t)}{p_{\theta}(s_t, a_t)} A^{\theta}(s_t, a_t)\right]$ (IS definition)
- $\sum_{t=0}^{T-1} \mathbb{E}_{s_t, a_t \sim p_{\theta}(s_t, a_t)}\left[\frac{p_{\theta'}(s_t)\, \pi_{\theta'}(a_t|s_t)}{p_{\theta}(s_t)\, \pi_{\theta}(a_t|s_t)} A^{\theta}\right]$ (decompose prob.)

An extra assumption

- **Assumption:** The probability distribution of states does not change much between updates

$$J(\theta') - J(\theta) = \sum_{t=0}^{T-1} \mathbb{E}_{s_t, a_t \sim p_\theta(s_t, a_t)} \left[\frac{\cancel{p_{\theta'}(s_t)}\, \pi_{\theta'}(a_t|s_t)}{\cancel{p_\theta(s_t)}\, \pi_\theta(a_t|s_t)} A^\theta \right]$$

- Seems reasonable if the policy does not change much, but we need to add constraints to ensure this happens.

Summary

- We can productively interpret policy gradients as a policy iteration algorithm
  - This enables us to reuse the advantage of the old policy

- We can remove the dependence on large products of probabilities
  - If we assume that the probability of states does not change much

- In the next class, we will see how different algorithms constrain changes to the probability distribution of states between updates
  - E.g., TRPO, PPO.