Lecture 7: Policy Gradients and Imitation learning

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2026

- Monotonic improvement slides and several PPO slides from Joshua Achiam

Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a family of methods that approximately enforce KL constraint **without computing natural gradients**. Two variants:

- Adaptive KL Penalty
  - Policy update solves unconstrained optimization problem
    $$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k)$$
  - Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint
- Clipped Objective
  - New objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then
    $$\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$
    where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)
  - Policy update is $\theta_{k+1} = \arg\max_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$

Check Your Understanding: Proximal Policy Optimization

- Clipped Objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

↑ is better

old policy

$$\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

$\pi_\theta(a|s) = 1$ ← .025

- where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)

$\pi_{\theta_k}(a|s) = 0.01$ ←

- Policy update is $\theta_{k+1} = \arg\max_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$.

$A > 0$

Consider the figure[1]. Select all that are true. $\epsilon \in (0, 1)$.

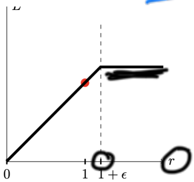
1. The left graph shows the L^{CLIP} objective when the advantage function $A > 0$ and the right graph shows when $A < 0$
2. The right graph shows the L^{CLIP} objective when the advantage function $A > 0$ and the left graph shows when $A < 0$
3. It depends on the value of ϵ
4. Not sure

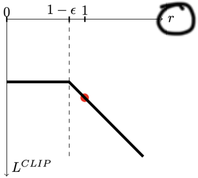
$A < 0$

$A > 0$ $\qquad$ $A < 0$

$A > 0$

$\dfrac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$





$\pi_{\theta_k}(a|s) = .5$

$\pi_\theta(a|s) \to 0$

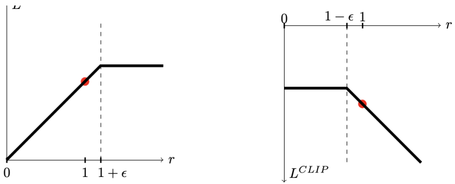
[1] Schulman, Wolski, Dhariwal, Radford, Klimov, 2017

- Clipped Objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

$$\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

- where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)
- Policy update is $\theta_{k+1} = \arg\max_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$.

Consider the figure[2]. Select all that are true. $\epsilon \in (0, 1)$.
The left graph shows the L^{CLIP} objective when the advantage function $A > 0$ and the right graph shows when $A < 0$



[2] Schulman, Wolski, Dhariwal, Radford, Klimov, 2017

Class Structure

- Last time: Advanced Policy Search
- This time: Policy search continued and Imitation Learning

Today

- Proximal policy optimization (PPO) (will implement in homework)
  - Generalized Advantage Estimation (GAE)
  - Theory: Monotonic Improvement Theory
- Imitation Learning
  - Behavior cloning
  - DAGGER
  - Max entropy inverse RL

Recall Problems with Policy Gradients

Policy gradient algorithms try to solve the optimization problem

$$\max_{\theta} J(\pi_\theta) \doteq \mathop{\mathrm{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

by taking stochastic gradient ascent on the policy parameters θ, using the *policy gradient*

$$g = \nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t) \right].$$

Limitations of policy gradients:

- Sample efficiency is poor
- Distance in parameter space $\neq$ distance in policy space!
  - What is policy space? For tabular case, set of matrices

$$\Pi = \left\{ \pi \; : \; \pi \in \mathbb{R}^{|S| \times |A|}, \; \sum_a \pi_{sa} = 1, \; \pi_{sa} \geq 0 \right\}$$

  - Policy gradients take steps in parameter space
  - Step size is hard to get right as a result

Recall Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a family of methods that approximately enforce KL constraint Two variants:

- Adaptive KL Penalty
  - Policy update solves unconstrained optimization problem

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta||\theta_k)$$

  - Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint
- Clipped Objective
  - New objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

$$\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

    where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)
  - Policy update is $\theta_{k+1} = \arg\max_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$

Recall Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a family of methods that approximately enforce KL constraint

- Adaptive KL Penalty
  - Policy update solves unconstrained optimization problem

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k)$$

  - Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint
- Clipped Objective
  - New objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

$$\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(\underline{r_t(\theta)}\hat{A}^{\pi_k}_t, \text{clip}\left(\underline{r_t(\theta)}, 1-\epsilon, 1+\epsilon\right)\hat{A}^{\pi_k}_t)\right]\right]$$

exact

where ϵ is a hyperparameter (maybe $\epsilon = 0.2$)
  - Policy update is $\theta_{k+1} = \arg\max_\theta \mathcal{L}^{CLIP}_{\theta_k}(\theta)$
- **How do we estimate the advantage function inside the policy update?**

but we don't know the "true" advantage

Recall N-step estimators

$$\nabla_\theta V(\theta) \quad \approx \quad (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} A_{ti} \nabla_\theta \log \pi_\theta(a_{ti}|s_{ti})$$

- Recall the N-step advantage estimators

$$\hat{A}_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t) \qquad (*)$$
$$\hat{A}_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t)$$
$$\hat{A}_t^{(\text{inf})} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots - V(s_t)$$

- Define $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$. Then

$r_t + \gamma V(s_{t+1}) - V(s_t) + \gamma\,(r_{t+1} + \gamma V(s_{t+2}) - V(s_{t+1}))$

$(*) \rightarrow$

$$\hat{A}_t^{(1)} = \delta_t^V \qquad = r_t + \gamma V(s_{t+1}) - V(s_t)$$
$$\hat{A}_t^{(2)} = \delta_t^V + \gamma \delta_{t+1}^V \qquad = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t)$$
$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V \qquad = \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V(s_{t+k}) - V(s_t)$$

- Note the above is an instance of a **telescoping sum**

Generalized Advantage Estimator (GAE)

GAE is often used in PPO

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V(s_{t+k}) - V(s_t) \tag{1}$$

- GAE is an exponentially-weighted average of k-step estimators

$$\begin{aligned} \hat{A}_t^{GAE(\gamma,\lambda)} &= (1-\lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \ldots) \\ &= (1-\lambda)(\underline{\delta_t^V} + \lambda(\underline{\delta_t^V} + \gamma\delta_{t+1}^V) + \lambda^2(\underline{\delta_t^V} + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \ldots) \end{aligned}$$

$\delta_t^V (1-\lambda)(1+\lambda+\lambda^2+\lambda^3+\ldots)$

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V(s_{t+k}) - V(s_t) \tag{2}$$

- GAE is an exponentially-weighted average of k-step estimators

note typo fixed post lecture

$$\begin{aligned}
\hat{A}_t^{GAE(\gamma,\lambda)} &= (1-\lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots) \\
&= (1-\lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots) \\
&= (1-\lambda)(\delta_t^V(1 + \lambda + \lambda^2 + \dots) + \gamma\delta_{t+1}^V(\lambda + \lambda^2 + \dots) \\
&\quad + \gamma^2\delta_{t+2}^V(\lambda^2 + \lambda^3 + \dots) + \dots) \\
&= (1-\overset{\lambda}{\cancel{\gamma}})(\delta_t^V \underbrace{\frac{1}{1-\lambda}}_{\text{geometric series}} + \gamma\lambda\delta_{t+1}^V \frac{1}{1-\lambda} + \gamma^2\lambda^2\delta_{t+2}^V \frac{1}{1-\lambda} + \dots) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\end{aligned}$$

- Introduced in "High-Dimensional Continuous Control Using Generalized Advantage Estimation" ICLR 2016 by Schulman et al.
- Our derivation follows the derivation presented in the paper

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V(s_{t+k}) - V(s_t) \tag{3}$$

$$A^{\pi}(a,s) = Q^{\pi}(a,s) - V^{\pi}(s)$$

- GAE is an exponentially-weighted average of k-step estimators

$$\begin{aligned}
\hat{A}_t^{GAE(\gamma,\lambda)} &= (1-\lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots) \\
&= (1-\lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots) \\
&= (1-\lambda)(\delta_t^V(1+\lambda+\lambda^2+\dots) + \gamma\delta_{t+1}^V(\lambda+\lambda^2+\dots) \\
&\quad +\gamma^2\delta_{t+2}^V(\lambda^2+\lambda^3+\dots) + \dots) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\end{aligned}$$

- What are the properties of GAE(γ,$\lambda = 0$) and GAE(γ,$\lambda = .99$)? (select all)
- (a) GAE(γ,$\lambda = .99$) is the advantage function using a TD(0) return
- (b) GAE(γ,0) is the advantage function using a TD(0) return
- (c) The bias of GAE(γ,0) is likely to be larger than GAE(γ,$\lambda = .99$)
- (d) Not sure

Check Your Understanding L7N2: GAE Solution

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V(s_{t+k}) - V(s_t) \tag{4}$$

- GAE is an exponentially-weighted average of k-step estimators

$$\begin{aligned}
\hat{A}_t^{GAE(\gamma,\lambda)} &= (1-\lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots) \\
&= (1-\lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots) \\
&= (1-\lambda)(\delta_t^V(1 + \lambda + \lambda^2 + \dots) + \gamma\delta_{t+1}^V(\lambda + \lambda^2 + \dots) \\
&\quad + \gamma^2\delta_{t+2}^V(\lambda^2 + \lambda^3 + \dots) + \dots) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\end{aligned}$$

- What are the properties of GAE(γ,0) and GAE(γ,1)? (select all)
- (a) GAE(γ,1) is the advantage function using a TD(0) return
- (b) GAE(γ,0) is the advantage function using a TD(0) return
- (c) The variance of GAE(γ,0) is likely to be larger than GAE(γ,1)
- (d) The bias of GAE(γ,0) is likely to be larger than GAE(γ,1)
- (e) Not sure

b and d are true

Generalized Advantage Estimator (GAE) Balance

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V(s_{t+k}) - V(s_t) \tag{5}$$

- GAE is an exponentially-weighted average of k-step estimators

$$\begin{aligned}
\hat{A}_t^{GAE(\gamma,\lambda)} &= (1-\lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \ldots) \\
&= (1-\lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \ldots) \\
&= (1-\lambda)(\delta_t^V(1+\lambda+\lambda^2+\ldots) + \gamma\delta_{t+1}^V(\lambda+\lambda^2+\ldots) \\
&\quad +\gamma^2\delta_{t+2}^V(\lambda^2+\lambda^3+\ldots)+\ldots) \\
&= (1-\gamma)(\delta_t^V \frac{1}{1-\lambda} + \gamma\lambda\delta_{t+1}^V \frac{1}{1-\lambda} + \gamma^2\lambda^2\delta_{t+2}^V \frac{1}{1-\lambda} + \ldots) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\end{aligned}$$

- Introduced in "High-Dimensional Continuous Control Using Generalized Advantage Estimation" ICLR 2016 by Schulman et al.
- In general will prefer $\lambda \in (0,1)$ to balance bias and variance

Generalized Advantage Estimator (GAE) in PPO

- GAE is an exponentially-weighted average of k-step estimators

$$
\begin{aligned}
\hat{A}_t^{(k)} &= \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V(s_{t+k}) - V(s_t) \\
\delta_t^V &= r_t + \gamma V(s_{t+1}) - V(s_t) \\
\hat{A}_t^{GAE(\gamma,\lambda)} &= (1-\lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\end{aligned}
$$

- PPO uses a truncated version of a GAE

$$
\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}^V
$$

- Benefits: Only have to run policy in environment for T timesteps before updating, improved estimate of gradient

Monotonic Improvement Theory

no data from π'

In last lecture used $d^{\pi'}$ as approximation of d^{π} (Why?)

$$J(\pi') - J(\pi) \approx \frac{1}{1-\gamma} \underset{\substack{s \sim d^{\pi} \\ a \sim \pi}}{\mathbb{E}} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^{\pi}(s,a) \right]$$
$$\doteq \underline{\mathcal{L}_{\pi}(\pi')}$$

This approximation is good when π' and π are close in KL-divergence

Relative policy performance bounds: [3]

$$\left| J(\pi') - \left(J(\pi) + \mathcal{L}_{\pi}(\pi') \right) \right| \leq C \sqrt{\underset{s \sim d^{\pi}}{\mathbb{E}} \left[D_{KL}(\pi' || \pi)[s] \right]} \tag{6}$$

constant

[3] Achiam, Held, Tamar, Abbeel, 2017

Monotonic Improvement Theory

RHS is a lower bound on how much better π' is than π

max wrt π'

From the bound on the previous slide, we get

$$\boxed{J(\pi') - J(\pi)} \geq \underbrace{\mathcal{L}_\pi(\pi') - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^\pi}[D_{KL}(\pi'||\pi)[s]]}}_{\text{RHS}}.$$

* [

- If we maximize the right hand side (RHS) with respect to π', we are **guaranteed to improve over π**.

  L wrt '

  - This is a *majorize-maximize* algorithm w.r.t. the true objective, the LHS.
- And $\mathcal{L}_\pi(\pi')$ & the KL-divergence term *can both be estimated with samples from* π!

trying to show $V(\pi_{k+1}) - V(\pi_k) > 0$

$J(\pi_{k+1}) - J(\pi_k) > 0$

Monotonic Improvement Theory

Proof of improvement guarantee: Suppose π_{k+1} and π_k are related by

①

$$\pi_{k+1} = \arg\max_{\pi'} \overbrace{\mathcal{L}_{\pi_k}(\pi') ① - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^{\pi_k}}\left[D_{KL}(\pi'||\pi_k)[s]\right]}}^{RHS}. \quad ②$$

$$\mathcal{L}_{\pi_k}(\pi_k) = \frac{1}{1-\gamma} E_{s\sim d^{\pi_k}, a\sim \pi_k}\left[\frac{\pi_k(a|s)}{\pi_k(a|s)} A^{\pi_k}(s,a)\right]$$

$$= \frac{1}{1-\gamma} E_{s\sim d^{\pi_k}} \left[\sum_a \pi_k(a|s) A^{\pi_k}(s,a)\right]$$

$$= \frac{1}{1-\gamma} E_{s\sim d^{\pi_k}} \sum_a \pi_k(a|s)\left(Q^{\pi_k}(s,a) - V^{\pi_k}(s)\right)$$

$$= \text{''} \quad \text{''} \quad \left[\left(\underbrace{\sum_a \pi_k(a|s) Q^{\pi_k}(s,a)}_{V^{\pi_k}(s)}\right) - V^{\pi_k}(s)\right]$$

$$= 0$$

② $D_{KL}(\pi_k | \pi_k) = 0$

RHS = ① - ② = 0 if $\pi_{k+1} = \pi_k$

$\arg\max_{\pi'}$ RHS ≥ 0 because π_k is at least as good as π_k

$V(\pi_{k+1}) - V(\pi_k) \geq$ ① - ② ≥ 0

$\square$

Proof of improvement guarantee: Suppose π_{k+1} and π_k are related by

$$\pi_{k+1} = \arg\max_{\pi'} \mathcal{L}_{\pi_k}(\pi') - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^{\pi_k}}[D_{KL}(\pi'||\pi_k)[s]]}.$$

- π_k is a feasible point, and the objective at π_k is equal to 0.
  - $\mathcal{L}_{\pi_k}(\pi_k) \propto \mathop{\mathrm{E}}_{s,a\sim d^{\pi_k},\pi_k}[A^{\pi_k}(s,a)] = 0$
  - $D_{KL}(\pi_k||\pi_k)[s] = 0$
- $\Longrightarrow$ optimal value ≥ 0
- $\Longrightarrow$ by the performance bound, $J(\pi_{k+1}) - J(\pi_k) \geq 0$

This proof works even if we restrict the domain of optimization to an arbitrary class of parametrized policies Π_θ, as long as $\pi_k \in \Pi_\theta$.

$$\pi_{k+1} = \arg\max_{\pi'} \mathcal{L}_{\pi_k}(\pi') - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^{\pi_k}}\left[D_{KL}(\pi'||\pi_k)[s]\right]}. \qquad 1 \tag{7}$$

Problem:

- C provided by theory is quite high when γ is near 1
- $\Longrightarrow$ steps from Equation ($\cancel{??}$ 1) are too small.

Potential Solution:

- Tune the KL penalty ($\Longrightarrow$ PPO)
- Use KL constraint (called **trust region**).

PPO Summary

- Improves data efficiency: can take several gradient steps before gathering more data from new policy
- Uses clipping (or KL constraint) to help increase likelihood of monotonic improvement
  - Conservative policy updating is an influential idea in RL, stemming at least from early 2000s
- Converges to local optima
- Very popular method, easy to implement, used in ChatGPT tuning

Policy Gradient Summary

- Extremely popular and useful algorithms, many beyond this class
- Can be used when the reward function is not differentiable
- Often used in conjunction with model-free value methods: actor-critic methods

Today

- Proximal policy optimization (PPO) (will implement in homework)
  - Generalized Advantage Estimation (GAE)
  - Theory: Monotonic Improvement Theory
- **Imitation Learning**[4]
  - Behavior cloning
  - DAGGER
  - Max entropy inverse RL

[4] With slides from Katerina Fragkiadaki and slides from Pieter Abbeel

Learning from Past Decisions and Outcomes

In some settings there exist very good decision policies and we would like to automate them

- One idea: humans provide reward signal when RL algorithm makes decisions
- Good: simple, cheap form of supervision
- Bad: High sample complexity

Alternative: imitation learning

Reward Shaping

Rewards that are **dense in time** closely guide the agent. How can we supply these rewards?

- **Manually design them**: often brittle
- **Implicitly specify them through demonstrations**

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010

Examples

- Simulated highway driving [Abbeel and Ng, ICML 2004; Syed and Schapire, NIPS 2007; Majumdar et al., RSS 2017]
- Parking lot navigation [Abbeel, Dolgov, Ng, and Thrun, IROS 2008]

Learning from Demonstrations

- Expert provides a set of **demonstration trajectories**: sequences of states and actions no reward
- Imitation learning is useful when it is easier for the expert to demonstrate the desired behavior rather than:
  - Specifying a reward that would generate such behavior,
  - Specifying the desired policy directly

Problem Setup

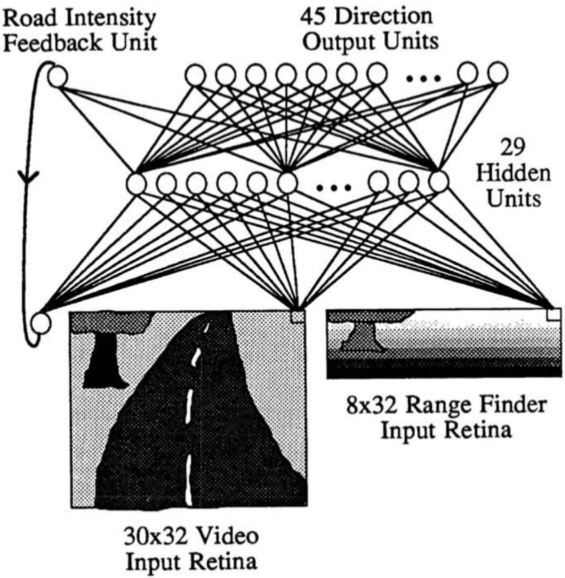
- Input:
  - State space, action space
  - Transition model $P(s' \mid s, a)$
  - No reward function R
  - Set of one or more expert's demonstrations $(s_0, a_0, s_1, s_0, \dots)$ (actions drawn from expert's policy π^*)
- Behavioral Cloning:
  - Can we directly learn the expert's policy using supervised learning?
- Inverse RL:
  - Can we recover R?
- Apprenticeship learning via Inverse RL:
  - Can we use R to generate a good policy?

Behavioral Cloning

Behavioral Cloning

- Reduce problem to a standard supervised machine learning problem:
  - Fix a policy class (e.g. neural network, decision tree, etc.)
  - Estimate a policy from training examples $(s_0, a_0), (s_1, a_1), (s_2, a_2), \ldots$
- Two early notable success stories: $\pi : s \rightarrow a$
  - Pomerleau, NIPS 1989: ALVINN
  - Summut et al., ICML 1992: Learning to fly in flight simulator

ALVINN



Road Intensity Feedback Unit
45 Direction Output Units
29 Hidden Units
8x32 Range Finder Input Retina
30x32 Video Input Retina

Behavioral cloning

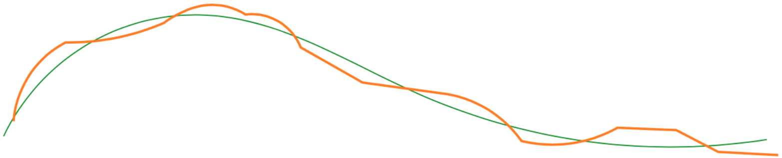
behavioral cloning
recursive NN

- Often behavior cloning in practice can work very well, especially if use BCRNN
- See What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. Mandlekar et al. CORL 2021
- Extensively used in practice

DAGGER

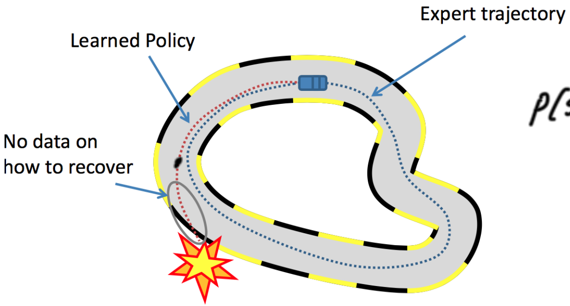
Potential Problem with Behavior Cloning: Compounding Errors

Supervised learning assumes iid. (s, a) pairs and ignores temporal structure
Independent in time errors:



Error at time t with probability $\leq \epsilon$
$\mathbb{E}$[Total errors] $\leq \epsilon T$

Problem: Compounding Errors



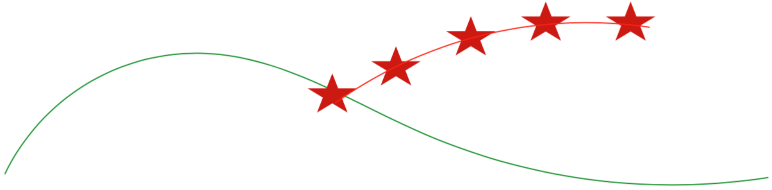
$p(s'|s,a)$

Data distribution mismatch!
In supervised learning, $(x, y) \sim D$ during train **and** test. In MDPs:

- Train: $s_t \sim D_{\pi^*}$
- Test: $s_t \sim D_{\pi_\theta}$

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011

Problem: Compounding Errors



- Error at time t with probability ϵ
- Approximate intuition: $\mathbb{E}[\text{Total errors}] \leq \epsilon(T + (T-1) + (T-2) \ldots + 1) \propto \epsilon T^2$
- Real result requires more formality. See Theorem 2.1 in http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats10-paper.pdf with proof in supplement: http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats10-sup.pdf

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011

DAGGER: Dataset Aggregation

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

- Idea: Get more labels of the expert action along the path taken by the policy computed by behavior cloning
- Obtains a stationary deterministic policy with good performance under its induced state distribution
- Key limitation?

Reward Learning

Feature Based Reward Function

- Given state space, action space, transition model $P(s' \mid s, a)$
- No reward function R
- Set of one or more expert's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from expert's policy π^*)
- Goal: infer the reward function R
- Assume that the expert's policy is optimal. What can be inferred about R?

Check Your Understanding L7N3: Feature Based Reward Function

- Given state space, action space, transition model $P(s' \mid s, a)$
- No reward function R
- Set of one or more expert's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from expert's policy π^*)
- Goal: infer the reward function R
- Assume that the expert's policy is optimal.

1. There is a single unique R that makes expert's policy optimal
2. There are many possible R that makes expert's policy optimal
3. It depends on the MDP
4. Not sure

Check Your Understanding L7N3: Feature Based Reward Function

- Given state space, action space, transition model $P(s' \mid s, a)$
- No reward function R
- Set of one or more expert's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from expert's policy π^*)
- Goal: infer the reward function R
- Assume that the expert's policy is optimal.

1. There is a single unique R that makes expert's policy optimal
2. There are many possible R that makes expert's policy optimal
3. It depends on the MDP
4. Not sure

Answer: There are an infinite set of R .

Linear Feature Reward Inverse RL

- Linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $\boldsymbol{w} \in \mathbb{R}^n, x : S \rightarrow \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi(s_0) = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|s_0]$$

Linear Feature Reward Inverse RL

- Related to linear value function approximation
- Consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \to \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi(s_0) = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0] \quad = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{w}^T x(s_t) \mid s_0]$$

$$= w^T \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t x(s_t) \mid s_0]$$

$$= \boldsymbol{w}^T \mu(\pi)$$

- where $\mu(\pi)(s)$ is defined as the discounted weighted frequency of state features under policy π, starting in state s_0.

Relating Frequencies to Optimality

- Assume $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \to \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- $V^\pi = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] = \boldsymbol{w}^T \mu(\pi)$ where $\mu(\pi)(s) =$ discounted weighted frequency of state s under policy π.

$$V^* \geq V^\pi$$

Relating Frequencies to Optimality

- Recall linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \to \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi = \boldsymbol{w}^T \mu(\pi)$$

- $\mu(\pi)(s) =$ discounted weighted frequency of state s under policy π.

$$\mathbb{E}_{s\sim\pi^*}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^*] = V^* \geq V^\pi = \mathbb{E}_{s\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] \quad \forall\pi$$

- Therefore if the expert's demonstrations are from the optimal policy, to identify $\boldsymbol{w}$ it is sufficient to find w^* such that

$$w^{*T} \mu(\pi^*) \geq w^{*T} \mu(\pi), \forall\pi \neq \pi^*$$

Feature Matching

- Want to find a reward function such that the expert policy outperforms other policies.
- For a policy π to be guaranteed to perform as well as the expert policy π^*, sufficient if its discounted summed feature expectations match the expert's policy [Abbeel & Ng, 2004].
- More precisely, if

$$\|\mu(\pi) - \mu(\pi^*)\|_1 \leq \epsilon$$

then for all w with $\|w\|_\infty \leq 1$ (uses Holder's inequality):

$$|w^T \mu(\pi) - w^T \mu(\pi^*)| \leq \epsilon$$

Ambiguity

- There is an infinite number of reward functions with the same optimal policy.
- There are infinitely many stochastic policies that can match feature counts
- Which one should be chosen?

Learning from Demonstration / Imitation Learning Pointers

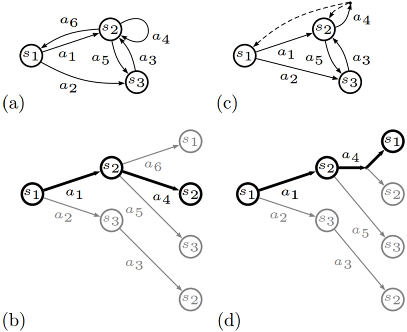
- Many different approaches
- Two of the key papers are:
  - Maximumum Entropy Inverse Reinforcement Learning (Ziebart et al. AAAI 2008)
  - Generative adversarial imitation learning (Ho and Ermon, NeurIPS 2016)

Max Entropy Inverse RL

Max Entropy Inverse RL

- Again assume a linear reward function $R(s) = \boldsymbol{w}^T x(s)$
- Define the total feature counts for a single trajectory τ_j as: $\mu_{\tau_j} = \sum_{s_i \in \tau_j} x(s_i)$
  - Note that this is a slightly different definition that we saw earlier
- The average feature counts over m trajectories is: $\tilde{\mu} = \frac{1}{m} \sum_{j=1}^{m} \mu_{\tau_j}$

Deterministic MDP Path Distributions



- Consider all possible H-step trajectories in a deterministic MDP
- For a linear reward model, a policy is completely specified by its distribution over trajectories
- Which policy/distribution should we choose given a set of m demonstrations?

Principle of Max Entropy

- Principle of max entropy: choose distribution with no additional preferences beyond matching the feature expectations in the demonstration dataset

$$\max_P - \sum_\tau P(\tau) \log P(\tau) s.t. \quad \sum_\tau P(\tau)\mu_\tau = \tilde{\mu} \qquad \sum_\tau P(\tau) = 1 \tag{8}$$

- In the linear reward case, this is equivalent to specifying the weights $\boldsymbol{w}$ that yield a policy with the max entropy constrained to matching the feature expectations

Ziebart et al., 2008

Max Entropy Principle

- Maximizing the entropy of the distribution over the paths subject to the feature constraints from observed data implies we maximize the likelihood of the observed data under the maximum entropy (exponential family) distribution[5].

$$P(\tau_j \mid w) = \frac{1}{Z(w)} \exp\left(w^T \mu_{\tau_j}\right) = \frac{1}{Z(w)} \exp\left(\sum_{s_i \in \tau_j} w^T x(s_i)\right)$$

$$Z(w, s) = \sum_{\tau_s} \exp\left(w^T \mu_{\tau_s}\right)$$

- Strong preference for low cost paths, equal cost paths are equally probable.

[5] Jaynes 1957

Stochastic MDPs

- Many MDPs of interest are stochastic
- For these the distribution over paths depends both on the reward weights and on the stochastic dynamics

$$P(\tau_j \mid w, P(s'|s,a)) \approx \frac{\exp\left(w^T \mu_{\tau_j}\right)}{Z(w, P(s'|s,a))} \prod_{s_i, a_i \in \tau_j} P(s_{i+1}|s_i, a_i)$$

Learning w

- Select $\boldsymbol{w}$ to maximize likelihood of data:

$$w^* = \arg\max_w L(w) = \arg\max_w \sum_{\text{examples}} \log P(\tau \mid w)$$

- The gradient is the difference between expected empirical feature counts and the learner's expected feature counts, which can be expressed in terms of expected state visitation frequencies

$$\nabla L(w) = \tilde{\mu} - \sum_{\tau} P(\tau \mid w)\mu_\tau = \tilde{\mu} - \sum_{s_i} D(s_i)x(s_i)$$

- where $D(s_i)$: state visitation frequency
- Do we need to know the transition model to compute the above?

MaxEnt IRL Algorithm for Frequencies

Backward pass

1. Set $Z_{s_i,0} = 1$
2. Recursively compute for N iterations

$$Z_{a_{i,j}} = \sum_k P(s_k|s_i, a_{i,j}) e^{\text{reward}(s_i|\theta)} Z_{s_k}$$

$$Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}}$$

Local action probability computation

3. $P(a_{i,j}|s_i) = \dfrac{Z_{a_{i,j}}}{Z_{s_i}}$

Forward pass

4. Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$
5. Recursively compute for $t = 1$ to N

$$D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$$

Summing frequencies

6. $D_{s_i} = \sum_t D_{s_i,t}$

Max Entropy IRL

- Max entropy approach has been hugely influential
- Provides a principled way for selecting among the (many) possible reward functions
- The original formulation requires knowledge of the transition model or the ability to simulate/act in the world to gather samples of the transition model
    - Check your understanding: was this needed in behavioral cloning?

From IRL to Policies

- Inverse RL approaches provide a way to learn a reward function
- Generally interested in using this reward function to compute a policy whose performance equals or exceeds the expert policy
- One approach: given learned reward function, use with regular RL
- Can we more directly learn the desired policy?

Summary

- Imitation learning can greatly reduce the amount of data need to learn a good policy
- Challenges remain and one exciting area is combining inverse RL / learning from demonstration and online reinforcement learning
- For a look into some of the theory between imitation learning and RL, see Sun, Venkatraman, Gordon, Boots, Bagnell (ICML 2017)
- Discussed learning rewards and using that
- Note often interested in learning rewards when only have preference pairs ($y_1 \succ y_2$)
  - "Dueling bandits"
  - We will see more on this setting shortly (and in homework 3)

- Define behavior cloning and how it differs from reinforcement learning

Importance Sampling for Off Policy, Policy Gradient

Importance Sampling

Importance sampling is a technique for estimating expectations using samples drawn from a different distribution.

$$\mathop{\mathrm{E}}_{x \sim P}[f(x)] =$$

Importance Sampling

Importance sampling is a technique for estimating expectations using samples drawn from a different distribution.

$$\mathop{\mathrm{E}}_{x \sim P}[f(x)] = \mathop{\mathrm{E}}_{x \sim Q}\left[\frac{P(x)}{Q(x)} f(x)\right] \approx \frac{1}{|D|} \sum_{x \in D} \frac{P(x)}{Q(x)} f(x), \quad D \sim Q$$

The ratio $P(x)/Q(x)$ is the **importance sampling weight** for x.

Importance Sampling

Importance sampling is a technique for estimating expectations using samples drawn from a different distribution.

$$\mathop{\mathrm{E}}_{x\sim P}[f(x)] = \mathop{\mathrm{E}}_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx \frac{1}{|D|}\sum_{x\in D}\frac{P(x)}{Q(x)}f(x), \quad D\sim Q$$

The ratio $P(x)/Q(x)$ is the **importance sampling weight** for x.

What is the variance of an importance sampling estimator?

$$\begin{aligned}
\mathrm{var}(\hat{\mu}_Q) &= \frac{1}{N}\mathrm{var}\left(\frac{P(x)}{Q(x)}f(x)\right) \\
&= \frac{1}{N}\left(\mathop{\mathrm{E}}_{x\sim Q}\left[\left(\frac{P(x)}{Q(x)}f(x)\right)^2\right] - \mathop{\mathrm{E}}_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right]^2\right) \\
&= \frac{1}{N}\left(\color{red}{\mathop{\mathrm{E}}_{x\sim P}\left[\frac{P(x)}{Q(x)}f(x)^2\right]} - \mathop{\mathrm{E}}_{x\sim P}[f(x)]^2\right)
\end{aligned}$$

The term in red is problematic—if $P(x)/Q(x)$ is large in the wrong places, the variance of the estimator explodes.

Importance Sampling for Policy Gradients

Here, we compress the notation π_θ down to θ in some places for compactness.

$$
\begin{aligned}
g = \nabla_\theta J(\theta) &= \mathop{\mathrm{E}}_{\tau \sim \theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t)\right] \\
&= \sum_{\tau} \sum_{t=0}^{\infty} \gamma^t P(\tau_t|\theta) \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t) \\
&= \mathop{\mathrm{E}}_{\tau \sim \theta'}\left[\sum_{t=0}^{\infty} \frac{P(\tau_t|\theta)}{P(\tau_t|\theta')} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t)\right]
\end{aligned}
$$

Importance Sampling for Policy Gradients

Here, we compress the notation π_θ down to θ in some places for compactness.

$$
\begin{aligned}
g = \nabla_\theta J(\theta) &= \mathop{\mathrm{E}}_{\tau\sim\theta}\left[\sum_{t=0}^{\infty}\gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t)\right] \\
&= \sum_{\tau}\sum_{t=0}^{\infty}\gamma^t P(\tau_t|\theta)\nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t) \\
&= \mathop{\mathrm{E}}_{\tau\sim\color{red}{\theta'}}\left[\sum_{t=0}^{\infty}\color{red}{\frac{P(\tau_t|\theta)}{P(\tau_t|\theta')}}\color{black}\gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t)\right]
\end{aligned}
$$

$\frac{P(\tau_t|\theta)}{P(\tau_t|\theta')} =$

Importance Sampling for Policy Gradients

Here, we compress the notation π_θ down to θ in some places for compactness.

$$
\begin{aligned}
g = \nabla_\theta J(\theta) &= \mathop{\mathrm{E}}_{\tau \sim \theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t)\right] \\
&= \sum_{\tau} \sum_{t=0}^{\infty} \gamma^t P(\tau_t|\theta) \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t) \\
&= \mathop{\mathrm{E}}_{\tau \sim \theta'}\left[\sum_{t=0}^{\infty} \frac{P(\tau_t|\theta)}{P(\tau_t|\theta')} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t)\right]
\end{aligned}
$$

Challenge? **Exploding or vanishing importance sampling weights.**

$$
\frac{P(\tau_t|\theta)}{P(\tau_t|\theta')} = \frac{\mu(s_0) \prod_{t'=0}^{t} P(s_{t'+1}|s_{t'}, a_{t'}) \pi_\theta(a_{t'}|s_{t'})}{\mu(s_0) \prod_{t'=0}^{t} P(s_{t'+1}|s_{t'}, a_{t'}) \pi_{\theta'}(a_{t'}|s_{t'})} = \prod_{t'=0}^{t} \frac{\pi_\theta(a_{t'}|s_{t'})}{\pi_{\theta'}(a_{t'}|s_{t'})}
$$

Even for policies only slightly different from each other, **many small differences multiply to become a big difference**.

Big question: how can we make efficient use of the data we already have from the old policy, while avoiding the challenges posed by importance sampling?

Advanced Policy Gradients

Theory:

1. Problems with Policy Gradient Methods
2. Policy Performance Bounds
3. Monotonic Improvement Theory

Proximal Policy Optimization:

1. Approximately constraints policy steps
2. Relatively simple to implement
3. Good empirical success and very widely used

Refresh Your Understanding L7N1

Which of the following are true about REINFORCE? In the following options, PG stands for policy gradient.

a. Adding a baseline term can help to reduce the variance of the PG updates
b. It will converge to a global optima
c. It can be initialized with a sub-optimal, deterministic policy and still converge to a local optima, given the appropriate step sizes
d. If we take one step of PG, it is possible that the resulting policy gets worse (in terms of achieved returns) than our initial policy

Refresh Your Understanding L7N1

Which of the following are true about REINFORCE? In the following options, PG stands for policy gradient.

(a) Adding a baseline term can help to reduce the variance of the PG updates
(b) It will converge to a global optima
(c) It can be initialized with a sub-optimal, deterministic policy and still converge to a local optima, given the appropriate step sizes
(d) If we take one step of PG, it is possible that the resulting policy gets worse (in terms of achieved returns) than our initial policy