Lecture 6: Policy Gradient II. Advanced policy gradient section slides from Joshua Achiam (OpenAI)'s slides, with minor modifications

Emma Brunskill

CS234 Reinforcement Learning.

Spring 2024

Refresh Your Knowledge L6N1

- Select all that are true about policy gradients:
  1. $\nabla_\theta V(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)]$
  2. θ is always increased in the direction of $\nabla_\theta \ln(\pi(S_t, A_t, \theta)$.
  3. State-action pairs with higher estimated Q values will increase in probability on average
  4. Are guaranteed to converge to the global optima of the policy class
  5. Not sure

Refresh Your Knowledge L6N1 Solutions

- Select all that are true about policy gradients:
  1. $\nabla_\theta V(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)]$
  2. θ is always increased in the direction of $\nabla_\theta \ln(\pi(S_t, A_t, \theta)$.
  3. State-action pairs with higher estimated Q values will increase in probability on average
  4. Are guaranteed to converge to the global optima of the policy class
  5. Not sure

Class Structure

- Last time: Policy Search
- This time: Policy search continued.

Today

- Likelihood ratio / score function policy gradient
  - Baseline
  - Alternative targets
- Advanced policy gradient methods
  - Proximal policy optimization (PPO) (will implement in homework)

Likelihood Ratio / Score Function Policy Gradient

$$\nabla_\theta V(\theta) \approx (1/m) \sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

- Unbiased but very noisy
- Fixes that can make it practical
  - Temporal structure
  - **Baseline**
  - Alternatives to using Monte Carlo returns $R(\tau^{(i)})$ as targets

Desired Properties of a Policy Gradient RL Algorithm

- Goal: Converge as quickly as possible to a local optima
  - To obtain data that use to learn, have to make actual decisions which may be suboptimal
  - Aim: minimize number of iterations / time steps until reach a good policy

Policy Gradient Algorithms and Reducing Variance

Table of Contents

Policy Gradient: Introduce Baseline

- Reduce variance by introducing a *baseline* $b(s)$

$$\nabla_\theta \mathbb{E}_\tau[R] = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right]$$

- For any choice of b, gradient estimator is unbiased.
- Near optimal choice is the expected return,

$$b(s_t) \approx \mathbb{E}[r_t + r_{t+1} + \cdots + r_{T-1}]$$

- Interpretation: increase logprob of action a_t proportionally to how much returns $\sum_{t'=t}^{T-1} r_{t'}$ are better than expected

Baseline $b(s)$ Does Not Introduce Bias–Derivation

$$
\begin{aligned}
&\mathbb{E}_\tau[\nabla_\theta \log \pi(a_t|s_t;\theta) b(s_t)] \\
&= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}}\left[\mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t;\theta) b(s_t)]\right]
\end{aligned}
$$

Baseline $b(s)$ Does Not Introduce Bias–Derivation

$$
\begin{aligned}
&\mathbb{E}_\tau[\nabla_\theta \log \pi(a_t|s_t;\theta) b(s_t)] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[\mathbb{E}_{s_{(t+1):T},a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t;\theta) b(s_t)]\right] \text{ (break up expectation)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\mathbb{E}_{s_{(t+1):T},a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t;\theta)]\right] \text{ (pull baseline term out)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}[b(s_t)\mathbb{E}_{a_t}[\nabla_\theta \log \pi(a_t|s_t;\theta)]] \text{ (remove irrelevant variables)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\sum_a \pi_\theta(a_t|s_t)\frac{\nabla_\theta \pi(a_t|s_t;\theta)}{\pi_\theta(a_t|s_t)}\right] \text{ (likelihood ratio)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\sum_a \nabla_\theta \pi(a_t|s_t;\theta)\right] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\nabla_\theta \sum_a \pi(a_t|s_t;\theta)\right] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}[b(s_t)\nabla_\theta 1] \\
&= \mathbb{E}_{s_0:t,a_0:(t-1)}[b(s_t)\cdot 0] = 0
\end{aligned}
$$

"Vanilla" Policy Gradient Algorithm

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **do**
  Collect a set of trajectories by executing the current policy
  At each timestep t in each trajectory τ^i, compute
    Return $G_t^i = \sum_{t'=t}^{T-1} r_{t'}^i$, and
    Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t^i)$.
  Re-fit the baseline, by minimizing $\sum_i \sum_t |b(s_t^i) - G_t^i|^2$,
  Update the policy, using a policy gradient estimate $\hat{g}$,
    Which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.
    (Plug $\hat{g}$ into SGD or ADAM)
endfor

Other Choices for Baseline?

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \cdots$ **do**
  Collect a set of trajectories by executing the current policy
  At each timestep t in each trajectory τ^i, compute
    Return $G_t^i = \sum_{t'=t}^{T-1} r_{t'}^i$, and
    Advantage estimate $\hat{A}_t^i = G_t^i - b(s_t^i)$.
  Re-fit the baseline, by minimizing $\sum_i \sum_t |b(s_t^i) - G_t^i|^2$,
  Update the policy, using a policy gradient estimate $\hat{g}$,
    Which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.
    (Plug $\hat{g}$ into SGD or ADAM)
endfor

Choosing the Baseline: Value Functions

- Recall Q-function / state-action-value function:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[r_0 + \gamma r_1 + \gamma^2 r_2 \cdots | s_0 = s, a_0 = a\right]$$

- State-value function can serve as a great baseline

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}\left[r_0 + \gamma r_1 + \gamma^2 r_2 \cdots | s_0 = s\right] \\ &= \mathbb{E}_{a\sim\pi}[Q^{\pi}(s,a)] \end{aligned}$$

Policy Gradient Algorithms and Reducing Variance

Table of Contents

Likelihood Ratio / Score Function Policy Gradient

- Policy gradient:

$$\nabla_\theta \mathbb{E}[R] \approx (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t)(G_t^{(i)} - b(s_t))$$

- Fixes that improve simplest estimator
  - Temporal structure (shown in above equation)
  - Baseline (shown in above equation)
  - **Alternatives to using Monte Carlo returns G_t^i as estimate of expected discounted sum of returns for the policy parameterized by θ?**

Choosing the Target

- G_t^i is an estimation of the value function at s_t from a single roll out
- Unbiased but high variance
- Reduce variance by introducing bias using bootstrapping and function approximation
  - Just like we saw for TD vs MC, and value function approximation

Actor-critic Methods

- Estimate of V/Q is done by a **critic**
- **Actor-critic** methods maintain an explicit representation of policy and the value function, and update both
- A3C (Mnih et al. ICML 2016) is a very popular actor-critic method

Policy Gradient Formulas with Value Functions

- Recall:

$$\nabla_\theta \mathbb{E}_\tau[R] = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right]$$

$$\nabla_\theta \mathbb{E}_\tau[R] \approx \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta) \left(Q(s_t, a_t; \mathbf{w}) - b(s_t) \right) \right]$$

- Letting the baseline be an estimate of the value V, we can represent the gradient in terms of the state-action advantage function

$$\nabla_\theta \mathbb{E}_\tau[R] \approx \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta) \hat{A}^\pi(s_t, a_t) \right]$$

- where the advantage function $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

Choosing the Target: N-step estimators

$$\nabla_\theta V(\theta) \quad \approx \quad (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

- Note that critic can select any blend between TD and MC estimators for the target to substitute for the true state-action value function.

Choosing the Target: N-step estimators

$$\nabla_\theta V(\theta) \quad \approx \quad (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

- Note that critic can select any blend between TD and MC estimators for the target to substitute for the true state-action value function.

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

- If subtract baselines from the above, get advantage estimators

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

L6N2 Check Your Understanding: Blended Advantage Estimators

$$\nabla_\theta V(\theta) \quad \approx \quad (1/m)\sum_{i=1}^{m}\sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

- If subtract baselines from the above, get advantage estimators

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

- Select all that are true
- $\hat{A}_t^{(1)}$ has low variance & low bias.
- $\hat{A}_t^{(1)}$ has high variance & low bias.
- $\hat{A}_t^{(\infty)}$ low variance and high bias.
- $\hat{A}_t^{(\infty)}$ high variance and low bias.
- Not sure

LN6N2 Check Your Understanding: Blended Advantage Estimators Answers

$$\nabla_\theta V(\theta) \quad \approx \quad (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

- If subtract baselines from the above, get advantage estimators

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

Choosing the Target

- G_t^i is an estimation of the value function at s_t from a single roll out
- Unbiased but high variance
- Reduce variance by introducing bias using bootstrapping and function approximation
  - Just like in we saw for TD vs MC, and value function approximation

Actor-critic Methods

- Estimate of V/Q is done by a **critic**
- **Actor-critic** methods maintain an explicit representation of policy and the value function, and update both
- A3C (Mnih et al. ICML 2016) is a very popular actor-critic method

Policy Gradient Formulas with Value Functions

- Recall:

$$\nabla_\theta \mathbb{E}_\tau[R] = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right]$$

$$\nabla_\theta \mathbb{E}_\tau[R] \approx \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta) \left(Q(s_t, a_t; \boldsymbol{w}) - b(s_t) \right) \right]$$

- Letting the baseline be an estimate of the value V, we can represent the gradient in terms of the state-action advantage function

$$\nabla_\theta \mathbb{E}_\tau[R] \approx \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t;\theta) \hat{A}^\pi(s_t, a_t) \right]$$

- where the advantage function $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

Advanced Policy Gradients

Outline

Theory:

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

Algorithms:

1. Proximal Policy Optimization

The 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

Sample Efficiency in Policy Gradients

- Sample efficiency for vanilla policy gradient methods is poor
- Discard each batch of data immediately after **just one gradient step**
- Why? PG is an **on-policy expectation**.
- Two main approaches to obtaining an unbiased estimate of the policy gradient
  - Collect sample trajectories from policy, then form sample estimate. (More stable)
  - Use trajectories from other policies (Less stable)
- Opportunity: use old data to take **multiple gradient steps** before using the resulting new policy to gather more data
- Challenge: even if this is possible to use old data to estimate multiple gradients, how many steps should be taken?

Choosing a Step Size for Policy Gradients

Policy gradient algorithms are stochastic gradient ascent:

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$$

with step $\Delta_k = \alpha_k \hat{g}_k$.

- If the step is too large, **performance collapse** is possible (Why?)

Policy gradient algorithms are stochastic gradient ascent:

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$$

with step $\Delta_k = \alpha_k \hat{g}_k$.

- If the step is too large, **performance collapse** is possible (Why?)
- If the step is too small, progress is unacceptably slow
- "Right" step size changes based on θ

Automatic learning rate adjustment like advantage normalization, or Adam-style optimizers, can help. But does this solve the problem?

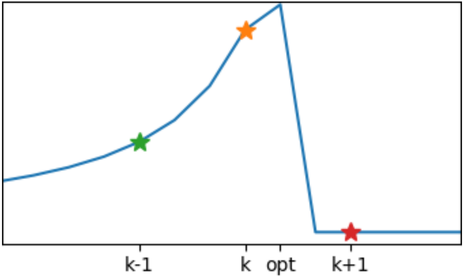


Figure: Policy parameters on x-axis and performance on y-axis. A bad step can lead to performance collapse, which may be hard to recover from.

The Problem is More Than Step Size

Consider a family of policies with parametrization:

$$\pi_\theta(a) = \begin{cases} \sigma(\theta) & a = 1 \\ 1 - \sigma(\theta) & a = 2 \end{cases}$$

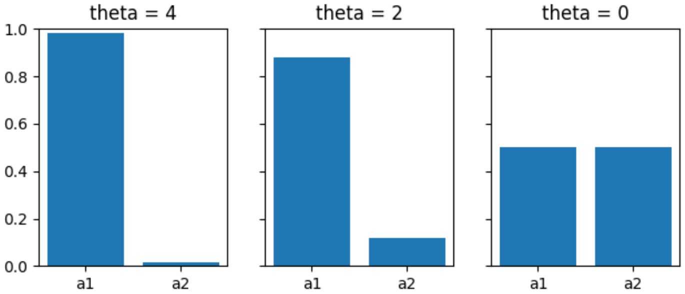


Figure: Small changes in the policy parameters can unexpectedly lead to **big** changes in the policy.

Big question: how do we come up with an update rule that doesn't ever change the policy more than we meant to?

Policy Performance Bounds

Relative Performance of Two Policies

In a policy optimization algorithm, we want an update step that

- uses rollouts collected from the most recent policy as efficiently as possible,
- and takes steps that respect **distance in policy space** as opposed to distance in parameter space.

To figure out the right update rule, we need to exploit relationships between the performance of two policies.

Performance difference lemma: In CS234 HW2 we ask you to prove that for any policies π, π'

$$J(\pi') - J(\pi) = \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right] \tag{1}$$

$$= \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'}} \left[A^{\pi}(s, a) \right] \tag{2}$$

where

$$d^{\pi}(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi)$$

What is it good for?

Can we use this for policy improvement, where π' represents the new policy and π represents the old one?

$$\begin{aligned}
\max_{\pi'} J(\pi') &= \max_{\pi'} J(\pi') - J(\pi) \\
&= \max_{\pi'} \mathop{\mathrm{E}}_{\tau \sim \pi'} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right]
\end{aligned}$$

This is suggestive, but not useful yet.

Nice feature of this optimization problem: defines the performance of π' in terms of the advantages from π!

But, problematic feature: still requires trajectories sampled from π'...

Looking at it from another angle...

In terms of the **discounted future state distribution** d^π, defined by

$$d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi),$$

we can rewrite the relative policy performance identity:

$$J(\pi') - J(\pi) = \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t, a_t)\right]$$

Note: Instance of Importance Sampling

In terms of the **discounted future state distribution** d^π, defined by

$$d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi),$$

we can rewrite the relative policy performance identity:

$$\begin{aligned}
J(\pi') - J(\pi) &= \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t, a_t)\right] \\
&= \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}}\left[A^\pi(s,a)\right] \\
&= \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi}}\left[\frac{\pi'(a|s)}{\pi(a|s)}A^\pi(s,a)\right]
\end{aligned}$$

Last step is an instance of **importance sampling** (more on this next time)

Problem: State Distribution

In terms of the **discounted future state distribution** d^π, defined by

$$d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi),$$

we can rewrite the relative policy performance identity:

$$\begin{aligned} J(\pi') - J(\pi) &= \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t A^\pi(s_t, a_t)\right] \\ &= \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}}\left[A^\pi(s,a)\right] \\ &= \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi}}\left[\frac{\pi'(a|s)}{\pi(a|s)}A^\pi(s,a)\right] \end{aligned}$$

...almost there! Only problem is $s \sim d^{\pi'}$.

What if we just said $d^{\pi'} \approx d^{\pi}$ and didn't worry about it?

$$\begin{aligned} J(\pi') - J(\pi) &\approx \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s \sim d^{\pi} \\ a \sim \pi}} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^{\pi}(s,a) \right] \\ &\doteq \mathcal{L}_{\pi}(\pi') \end{aligned}$$

Turns out: this approximation is pretty good when π' and π are close! But why, and how close do they have to be?

Relative policy performance bounds: [1]

$$\left| J(\pi') - \left(J(\pi) + \mathcal{L}_{\pi}(\pi') \right) \right| \leq C \sqrt{\mathop{\mathrm{E}}_{s \sim d^{\pi}} \left[D_{KL}(\pi' || \pi)[s] \right]} \tag{3}$$

If policies are close in KL-divergence—the approximation is good!

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

What is KL-divergence?

For probability distributions P and Q over a discrete random variable,

$$D_{KL}(P||Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$$

Properties:

- $D_{KL}(P||P) = 0$
- $D_{KL}(P||Q) \geq 0$
- $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ — Non-symmetric!

What is KL-divergence between policies?

$$D_{KL}(\pi'||\pi)[s] = \sum_{a \in \mathcal{A}} \pi'(a|s) \log \frac{\pi'(a|s)}{\pi(a|s)}$$

A Useful Approximation

What did we gain from making that approximation?

$$
\begin{aligned}
J(\pi') - J(\pi) &\approx \mathcal{L}_\pi(\pi') \\
\mathcal{L}_\pi(\pi') &= \frac{1}{1-\gamma} \underset{\substack{s\sim d^\pi \\ a\sim\pi}}{\mathrm{E}} \left[\frac{\pi'(a|s)}{\pi(a|s)} A^\pi(s,a) \right] \\
&= \underset{\tau\sim\pi}{\mathrm{E}} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)} A^\pi(s_t,a_t) \right]
\end{aligned}
$$

- This is something we can optimize using trajectories sampled from the old policy π!
- Similar to using importance sampling, but because weights only depend on current timestep (and not preceding history), they don't vanish or explode.

Recommended Reading

- "Approximately Optimal Approximate Reinforcement Learning," Kakade and Langford, 2002 [2]
- "Trust Region Policy Optimization," Schulman et al. 2015 [3]
- "Constrained Policy Optimization," Achiam et al. 2017 [4]

[2] https://people.eecs.berkeley.edu/ pabbeel/cs287-fa09/readings/KakadeLangford-icml2002.pdf
[3] https://arxiv.org/pdf/1502.05477.pdf
[4] https://arxiv.org/pdf/1705.10528.pdf

Algorithms

Proximal Policy Optimization (PPO) is a family of methods that approximately penalize policies from changing too much between steps. Two variants:

- Adaptive KL Penalty
  - Policy update solves unconstrained optimization problem

$$\begin{aligned} \theta_{k+1} &= \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k) \qquad (4) \\ \bar{D}_{KL}(\theta || \theta_k) &= E_{s \sim d^{\pi_k}} D_{KL}(\theta_k(\cdot|s), \pi_\theta(\cdot|s)) \qquad (5) \end{aligned}$$

  - Penalty coefficient β_k changes between iterations to approximately enforce KL-divergence constraint

Proximal Policy Optimization with Adaptive KL Penalty

Algorithm PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

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

  by taking K steps of minibatch SGD (via Adam)
  if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  else if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \leq \delta / 1.5$ **then**
    $\beta_{k+1} = \beta_k / 2$
  end if
end for

- Initial KL penalty not that important—it adapts quickly
- Some iterations may violate KL constraint, but most don't

PPO with Adaptive KL Penalty: **Multiple Gradient Steps**

Algorithm PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

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

  by taking K steps of minibatch SGD (via Adam)
  if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  else if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \leq \delta/1.5$ **then**
    $\beta_{k+1} = \beta_k/2$
  end if
end for

- Initial KL penalty not that important—it adapts quickly
- Some iterations may violate KL constraint, but most don't

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

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

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

L6N3 Check Your Understanding: Proximal Policy Optimization

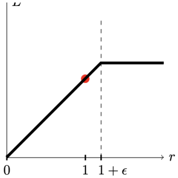
- Clipped Objective function: let $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_k}(a_t|s_t)$. Then

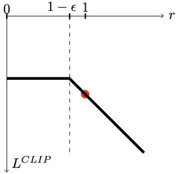
$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

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

Consider the figure[5]. 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





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

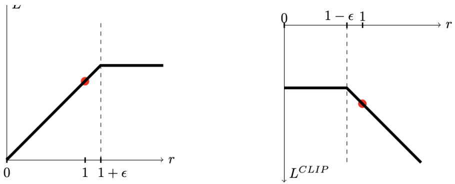
L6N3: Check Your Understanding L6N2 Proximal Policy Optimization Solutions

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

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

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

Consider the figure[6]. Select all that are true. $\epsilon \in (0, 1)$.



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

Proximal Policy Optimization with Clipped Objective

But *how* does clipping keep policy close? By making objective as pessimistic as possible about performance far away from θ_k:

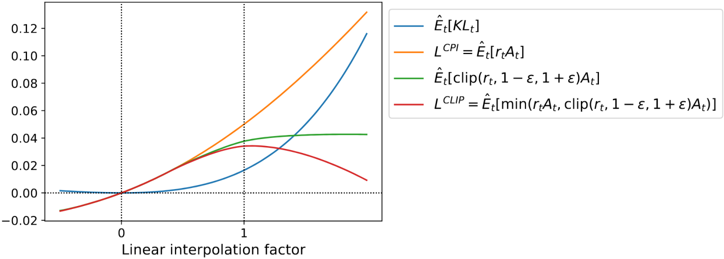


Figure: Various objectives as a function of interpolation factor α between θ_{k+1} and θ_k after one update of PPO-Clip [7]

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

Proximal Policy Optimization with Clipped Objective

Algorithm PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ

for $k = 0, 1, 2, \ldots$ **do**

Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Compute policy update

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

by taking K steps of minibatch SGD (via Adam), where

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

end for

- Clipping prevents policy from having incentive to go far away from θ_{k+1}
- Clipping seems to work at least as well as PPO with KL penalty, but is simpler to implement

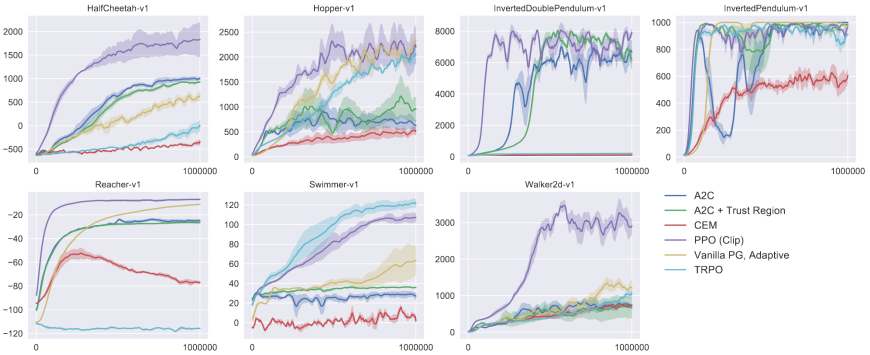


Figure: Performance comparison between PPO with clipped objective and various other deep RL methods on a slate of MuJoCo tasks. [8]

- Wildly popular, and key component of ChatGPT

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

PPO

- "Proximal Policy Optimization Algorithms," Schulman et al. 2017 [9]
- OpenAI blog post on PPO, 2017 [10]

[9] https://arxiv.org/pdf/1707.06347.pdf

[10] https://blog.openai.com/openai-baselines-ppo/

PPO: Algorithm and Code Implementation Details

- Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation Matters in Deep RL: A Case Study on PPO and TRPO. ICLR 2020
  `https://openreview.net/forum?id=r1etN1rtPB`
- Reward scaling, learning rate annealing, etc. can make a significant difference

Today

- Likelihood ratio / score function policy gradient
  - Baseline
  - Alternative targets
- Advanced policy gradient methods
  - Proximal policy optimization (PPO) algorithm (will implement in homework)

Class Structure

- Last time: Policy Search
- This time: Policy search continued.
- Next time: Proximal Policy Optimization (PPO) cont (theory and additional discussion)

Class Structure

- SLIDES FOR NEXT CLASS (LIKELY)

Monotonic Improvement Theory

From the bound on the previous slide, we get

$$J(\pi') - J(\pi) \geq \mathcal{L}_\pi(\pi') - C\sqrt{\underset{s \sim d^\pi}{\mathrm{E}}\left[D_{KL}(\pi' || \pi)[s]\right]}.$$

- If we maximize the RHS with respect to π', we are **guaranteed to improve over $\boldsymbol{\pi}$**.
  - This is a *majorize-maximize* algorithm w.r.t. the true objective, the LHS.
- **And** $\mathcal{L}_\pi(\pi')$ and the KL-divergence term *can both be estimated with samples from* π!

Monotonic Improvement Theory

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

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

Monotonic Improvement Theory

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

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

- π_k is a feasible point, and the objective at π_k is equal to 0.
  - $\mathcal{L}_{\pi_k}(\pi_k) \propto \underset{s,a\sim d^{\pi_k},\pi_k}{\mathrm{E}}\left[A^{\pi_k}(s,a)\right] = 0$
  - $D_{KL}(\pi_k||\pi_k)[s] = 0$
- $\Longrightarrow$ 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 $\pi_k \in \Pi_\theta$.

Approximate Monotonic Improvement

$$\pi_{k+1} = \arg\max_{\pi'} \mathcal{L}_{\pi_k}(\pi') - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^{\pi_k}}[D_{KL}(\pi'||\pi_k)[s]]}. \tag{6}$$

Problem:

- C provided by theory is quite high when γ is near 1
- $\implies$ steps from (6) are too small.

Potential Solution:

- Tune the KL penalty
- Use KL constraint (called **trust region**).

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.

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.

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

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

Importance Sampling for Policy Gradients

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

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

$$= \sum_{\tau} \sum_{t=0}^{\infty} \gamma^t P(\tau_t|\theta) \nabla_\theta \log \pi_\theta(a_t|s_t) A^\theta(s_t, a_t)$$

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

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