Policy Search Methods

(INF8250AE: Introduction to Reinforcement Learning)

Amir-massoud Farahmand

Polytechnique Montréal & Mila

Adaptive Agents Lab (Adage)

Table of Contents

Goal

We study how to parametrize the policy and update it in order to improve the performance.

- How to directly represent the policy and measure its performance
- How to improve the policy using zero-order methods such as random search and evolutionary algorithms
- How to improve the policy using the gradient information (first-order)
    - How to estimate the policy gradient
    - How to use the policy gradient to update the policy

Learning Objectives

You need to

- Remember: Policy parametrization, policy performance, evolutionary algorithm, policy gradient
- Understand: Why we parametrize the policy? How the policy gradient is derived?
- Apply: Policy gradient to improve the policy.

Introduction

- So far, we have focused on value-based methods to obtain the optimal policy
- Only the value function was explicitly represented.
  - The policy could be computed based on it. (Q: How?)
- There are also methods based on explicit representation of the policy and optimizing the performance of the agent by searching in the space of policies.
- We call them policy search methods.
- Hybrid methods: explicit representation of both value and policy.

Policy Parametrization

Policy Parametrization

- Consider a stochastic policy $\pi_\theta : \mathcal{X} \to \mathcal{M}(\mathcal{A})$ that is parameterized by a $\theta \in \Theta$.
- The set Θ is the parameter space, e.g., a subset of $\mathbb{R}^p$.
- The space of all parameterized policies:

$$\Pi_\Theta = \{ \pi_\theta : \mathcal{X} \to \mathcal{M}(\mathcal{A}) \; : \; \theta \in \Theta \} \,. \tag{1}$$

- This space depends on the mapping π_θ and Θ.

Policy Parametrization: Examples

- Many choices for how we can parameterize a policy π_θ.
- A generic example is based on the Boltzmann (or softmax) distribution.
- Given a function $f_\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ (e.g., a DNN or decision tree parameterized by θ), the density of choosing action a at state x is

$$\pi_\theta(a|x) = \frac{\exp(f_\theta(x,a))}{\int \exp(f_\theta(x,a'))\mathrm{d}a'}.$$

Policy Parametrization: Examples

- A special case would be when $f_\theta(x,a) = \phi(x,a)^\top \theta$ for some features $\phi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^p$ and $\theta \in \mathbb{R}^p$:

$$\pi_\theta(a|x) = \frac{\exp(\phi(x,a)^\top \theta)}{\int \exp(\phi(x,a')^\top \theta) \mathrm{d}a'}.$$

- When the action space $\mathcal{A}$ is discrete, $\pi_\theta(a|x)$ denotes the probability of choosing action a at state x (instead of its density):

$$\pi_\theta(a|x) = \frac{\exp(\phi(x,a)^\top \theta)}{\sum_{a' \in \mathcal{A}} \exp(\phi(x,a')^\top \theta)}.$$

Policy Parametrization: Examples

- Another example: $\pi_\theta(\cdot|x)$ defining a Normal distribution over action space with θ parameterization its mean and covariance:

$$\pi_\theta(\cdot|x) = \mathcal{N}\left(\mu_\theta(x), \Sigma_\theta(x)\right).$$

- If the action space is $d_{\mathcal{A}}$-dimensional:
  - Mean: $\mu_\theta : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{A}}}$
  - Covariance: $\Sigma_\theta : \mathcal{X} \to S_+^{d_{\mathcal{A}}}$. Here $S_+^{d_{\mathcal{A}}}$ refers to the set of $d_{\mathcal{A}} \times d_{\mathcal{A}}$ positive semi-definite matrices.

Ease of Work in Continuous Action Spaces

- Explicit parameterization of policy allows us to easily choose a continuous action
- For value-based methods, this can be challenging:
  - Even if we know Q^*, computing the optimal policy

    $$\pi^*(x) = \pi_g(x; Q^*) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(x, a)$$

    requires an optimization problem in $\mathcal{A}$.
  - Challenging if $\mathcal{A}$ is a high-dimensional space.
  - VI and PI requires repeated calculation of the greedy policy.
- Sure, action selection might be easy, but ...
- Question: How can we optimize the performance of a parametrized policy after all?
- This is the topic of this lecture!

Performance Measure

Performance Measure

- Before optimizing the performance of policy, we should clearly define what performance we want to optimize.
- The performance can be measured in various ways.
- We focus on the expected return of following π_θ, the value function.
    - This is the same as what we have done in this course.
    - We can also incorporate the variance or some other risk measures, relatively easily.
- Goal: Find a policy that maximizes this performance measure.
- Constraint: We are restricted to choosing policies within Π_Θ (1).

Performance Measure on a Single State

- Assume that we only care about the performance at state $x \in \mathcal{X}$.
- The goal of policy search:

$$\operatorname*{argmax}_{\pi \in \Pi_\Theta} V^{\pi}(x) = \operatorname*{argmax}_{\theta \in \Theta} V^{\pi_\theta}(x). \tag{2}$$

- Interpretation: We are interested in finding a policy such that if the agent starts at this particular state x, its performance, measured according to its expected return, is maximized.

Performance Measure on a Single State

$$\operatorname*{argmax}_{\pi \in \Pi_\Theta} V^{\pi}(x) = \operatorname*{argmax}_{\theta \in \Theta} V^{\pi_\theta}(x).$$

- The optimal policy π^* not only maximizes the value function at this particular x, but also at any other $x' \in \mathcal{X}$.
- But the optimal policy may not be in Π_Θ.
- If $\pi^* \notin \Pi_\Theta$, we will not be able to find a policy that maximizes the value at all states.
  - In that case, we may want to find a policy that is only good at our starting state x, and ignore the performance at other states.
  - The obtained policy is going to be initial-state-dependent. If we change x to another state $x' \neq x$, the optimal policy within Π_Θ might change.

Performance Measure with $X_1 \sim \rho$

- Instead of the extreme case of considering a single initial state x, we can consider when the initial state is distributed according to some distribution $\rho \in \mathcal{M}(\mathcal{X})$.
- The performance measure would be the average of following π_θ with the initial state $X_1 \sim \rho$.

$$J(\pi_\theta) = J_\rho(\pi_\theta) \triangleq \int V^{\pi_\theta}(x) \mathrm{d}\rho(x) = \mathbb{E}_{X \sim \rho}\left[V^{\pi_\theta}(X)\right]. \quad (3)$$

Performance Measure with $X_1 \sim \rho$

$$J(\pi_\theta) = J_\rho(\pi_\theta) \triangleq \int V^{\pi_\theta}(x) \mathrm{d}\rho(x) = \mathbb{E}_{X \sim \rho}\left[V^{\pi_\theta}(X)\right].$$

- The optimal policy maximizes J_ρ.
- $J_\rho(\pi^*) \geq J_\rho(\pi_\theta)$ for any $\pi_\theta \in \Pi_\Theta$.
- If $\pi^* \notin \Pi_\Theta$, the inequality is strict if the support of ρ is the whole state space $\mathcal{X}$.

Performance Measure with $X_1 \sim \rho$

$$J(\pi_\theta) = J_\rho(\pi_\theta) \triangleq \int V^{\pi_\theta}(x)\mathrm{d}\rho(x) = \mathbb{E}_{X\sim\rho}\left[V^{\pi_\theta}(X)\right].$$

- In policy search methods, we aim to find the maximizer of the performance measure within Π_Θ.

$$\bar{\pi} \leftarrow \underset{\pi_\theta \in \Pi_\Theta}{\operatorname{argmax}} J_\rho(\pi_\theta). \tag{4}$$

- The corresponding policy is identified by its parameter $\bar{\theta}$, i.e., $\bar{\pi} = \pi_{\bar{\theta}}$.
- For different ρ, we may get different optimizers.
- To emphasize the dependence of the maximizer on ρ, we may use $\bar{\pi}_\rho$.
- We may sometimes denote $J(\pi_\theta)$ or $J_\rho(\pi_\theta)$ simply by $J_\rho(\theta)$.

Policy Search as an Optimization Problem

- Question: How can we solve the optimization problem (4) to find π_θ that maximizes the performance measure J_ρ?
- This is an optimization problem, so we can benefit from the arsenal of optimization algorithms.
- Being an RL problem, however, brings both challenges and opportunities.
    - Challenge: The value of J_ρ is not readily available, and has to be estimated through interaction with the environment.
    - Opportunity: The special structure of the RL problem, such as the value function satisfying the Bellman equation.

Policy Search as an Optimization Problem

- Optimization methods, broadly speaking, can be categorized based on the information they need about their objective.
- Zero-order methods only use the value of the objective at various query points.
  - They compute $J_\rho(\theta)$ at various θs in order to guide the optimization process.
- First-order methods use the derivative of the objective instead of, or in addition to, the value of the objective.
  - They use $\nabla_\theta J_\rho(\theta)$ in order to guide the search.
  - The quintessential first-order optimization method is the gradient descent (and its stochastic variant).

Zero-Order Methods for Policy Optimization

Zero-Order Methods

- We first consider the case when the policy parameter space Θ is finite.
- This is not realistic, but helps us understand some of the challenges.
- We then extend our discussion to case when Θ is continuous space.

Zero-Order Methods: Finite Policy Parameter Space

- Assume that we are given a finite $\Theta = \{\theta_1, \dots, \theta_m\}$ policy parameters.
- This defines the finite policy space $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$.
- The goal of policy optimization: Find the policy $\pi_\theta \in \Pi_\Theta$ such that $J_\rho(\pi_\theta)$ is maximized, see (4).
- If we can easily compute $J_\rho(\pi_\theta)$ for each $\theta \in \Theta$, this is an easy problem, at least in principle.
- So the main issue is how to compute $J_\rho(\pi_\theta)$.

Zero-Order Methods: Finite Policy Parameter Space

- The performance measure $J_\rho(\pi_\theta) = \mathbb{E}_{X \sim \rho}\left[V^{\pi_\theta}(X)\right]$, i.e., the expectation of $V^{\pi_\theta}(X)$ w.r.t. $X \sim \rho$.
- We can try to compute $V^{\pi_\theta}(x)$ for all $x \in \mathcal{X}$, using any of the PE methods that we have developed, and take the weighted average according to ρ.
- If $\mathcal{X}$ is discrete, this would be

$$J_\rho(\pi_\theta) = \sum_{x \in \mathcal{X}} \rho(x) V^{\pi_\theta}(x).$$

- If $\mathcal{X}$ is large:
  - Computing V^{π_θ} itself is not going to be easy.
  - Computing the integral $\int V^{\pi_\theta}(x) \mathrm{d}\rho(x)$ is going to be challenging.

Zero-Order Methods: Finite Policy Parameter Space

- Alternative: Computing an unbiased estimate of $J_\rho(\pi_\theta)$ instead, using MC estimation.
- We derive this in two steps.
  - Assume that we know V^{π_θ}, estimate $J_\rho(\pi_\theta)$.
  - Replace $V^{\pi_\theta}(x)$ with the return $G^{\pi_\theta}(x)$.

Zero-Order Methods: Finite Policy Parameter Space

- We assume that we know V^{π_θ}, and we want to estimate $J_\rho(\pi_\theta)$.
- If we sample $X \sim \rho$, we have that $V^{\pi_\theta}(X)$ is an unbiased estimate of $J_\rho(\pi_\theta)$ as

$$\mathbb{E}\left[V^{\pi_\theta}(X)\right] = \int V^{\pi_\theta}(x) \mathrm{d}\rho(x) = J_\rho(\pi_\theta).$$

- If we draw n independent samples $X_1, \ldots, X_n \sim \rho$, the estimator

$$\frac{1}{n} \sum_{i=1}^{n} V^{\pi_\theta}(X_i)$$

would be unbiased as well.

Zero-Order Methods: Finite Policy Parameter Space

$$\frac{1}{n}\sum_{i=1}^{n} V^{\pi_\theta}(X_i)$$

- Variance:

$$\frac{\mathrm{Var}\left[V^{\pi_\theta}(X)\right]}{n}.$$

- This variance goes to 0 as n increases.
- The variance $\mathrm{Var}\left[V^{\pi_\theta}(X)\right]$ is a measure of dispersion of the value function across states samples according to ρ.
  - If the value function is constant, it will be zero.
  - If it is changing slowly as a function of the state, it would be small.
  - If the value function varies greatly, the variance is large.
- The variance is a function of the policy π_θ, so for each $\pi_\theta \in \Pi_\Theta$, we get a different variance.

Zero-Order Methods: Finite Policy Parameter Space

- The second step is to replace $V^{\pi_\theta}(x)$ with the return $G^{\pi_\theta}(x)$.
- The return $G^{\pi_\theta}(x)$ is an unbiased estimate of $V^{\pi_\theta}(x)$.
- Computation of $G^{\pi_\theta}(x)$ requires starting the agent from state x and following π_θ (i.e., performing a rollout from x) until the end of episode for episodic tasks, or until infinity for continual tasks.
- If $X \sim \rho$, $G^{\pi_\theta}(X)$ is an unbiased estimate of $J_\rho(\pi_\theta)$ as

$$\begin{aligned}\mathbb{E}_{X\sim\rho}\left[G^{\pi_\theta}(X)\right] &= \mathbb{E}_{X\sim\rho}\left[\mathbb{E}\left[G^{\pi_\theta}(X) \mid X\right]\right] \\ &= \mathbb{E}_{X\sim\rho}\left[V^{\pi_\theta}(X)\right] = J_\rho(\pi_\theta).\end{aligned}$$

Zero-Order Methods: Finite Policy Parameter Space

- If we draw n independently selected $X_1, \ldots, X_n \sim \rho$, we can form

$$\hat{J}_n(\pi_\theta) = \frac{1}{n} \sum_{i=1}^{n} G^{\pi_\theta}(X_i), \tag{5}$$

as an unbiased estimate of $J_\rho(\pi_\theta)$.

Zero-Order Methods: Finite Policy Parameter Space

Proposition

The estimator $\hat{J}_n(\pi_\theta)$ (5) *is an unbiased estimator for* $J_\rho(\pi_\theta)$ *and has the variance of*

$$\mathrm{Var}\left[\hat{J}_n(\pi_\theta)\right] = \frac{1}{n}\left(\mathbb{E}\left[\mathrm{Var}\left[G^{\pi_\theta}(X) \mid X\right]\right] + \mathrm{Var}\left[V^{\pi_\theta}(X)\right]\right).$$

- If we have a finite number of parameters in Θ, we can estimate

$$J_\rho(\pi_{\theta_i}) \approx \hat{J}_n(\pi_{\theta_i}) \pm O_P(\frac{1}{\sqrt{n}})$$

for each $\theta_i \in \Theta$. (Here $O_P(\cdot)$ is an order notation that hides quantities related to "this statement holds with probability at least $1-\delta$".)

Zero-Order Methods: Finite Policy Parameter Space

- We can use these estimates to select the best among them:

$$\hat{\pi} = \pi_{\hat{\theta}} \leftarrow \operatorname*{argmax}_{\theta \in \Theta} \hat{J}_n(\pi_\theta) \left[\approx J_\rho(\pi_\theta) \pm O_P\left(\frac{1}{\sqrt{n}}\right) \right] \qquad (6)$$

- This can be done with $n|\Theta|$ rollouts.
- As there is an $O_P(\frac{1}{\sqrt{n}})$ error in estimation of each $J_\rho(\pi_\theta)$, the selected policy $\hat{\pi}$ may not be the same as the maximizer $\bar{\pi}$ of (4).
- A mistake in the choice of optimal policy happens if

$$\hat{J}_n(\hat{\pi}) > \hat{J}_n(\bar{\pi}),$$

which leads to preferring $\hat{\pi}$ to $\bar{\pi}$ according to the empirical performance measure, even though

$$J_\rho(\hat{\pi}) < J_\rho(\bar{\pi}).$$

Zero-Order Methods: Finite Policy Parameter Space

- Even if we make an error in selecting the best policy, the gap in their performance is within $O_P(\frac{1}{\sqrt{n}})$.
- As we increase n, the error in estimating $J_\rho(\pi_\theta)$ decreases and the probability of selecting an optimal policy increases.
- This increased accuracy, however, is at the cost of increased sample and computational complexity, which would be $n|\Theta|$ rollouts.

Zero-Order Methods: Finite Policy Parameter Space

Proposition

Consider $\hat{\pi} = \pi_{\hat{\theta}} \leftarrow argmax_{\theta \in \Theta} \hat{J}_n(\pi_\theta)$ (6). *Assume that the returns* $G^{\pi_\theta}(x)$ *are all* Q_{max}*-bounded for any* $\theta \in \Theta$ *and* $x \in \mathcal{X}$*. Furthermore, suppose that* $|\Theta| < \infty$*. For any* $\delta > 0$*, we have that*

$$J_\rho(\hat{\theta}) \geq \max_{\theta \in \Theta} J_\rho(\theta) - 2Q_{max}\sqrt{\frac{2\ln\left(\frac{2|\Theta|}{\delta}\right)}{n}},$$

with probability at least $1 - \delta$*.*

Zero-Order Methods: Random Search

- If Θ is not finite, we cannot evaluate $\hat{J}_n(\pi_\theta)$ for all $\theta \in \Theta$.
- There are several generic methods for searching in a large parameter space:
  - Random Search (RS)
  - Simulated Annealing
  - Evolutionary Algorithms (many different variations)

Zero-Order Methods: Random Search

- We randomly pick m policy parameters $\theta_1, \ldots, \theta_m \in \Theta$.
- Evaluate $\hat{J}_n(\pi_{\theta_i})$
- Pick the one with the highest value.
- Intuition of why this works:
  - With large enough m, one of θ_i might hit close to the optimal

$$\hat{\theta} \leftarrow \operatorname*{argmax}_{\theta \in \Theta} \hat{J}_n(\pi_\theta).$$

  - If n is large enough, the difference between $\hat{J}_n(\theta)$ and $J_\rho(\theta)$ would be small for all randomly selected θ.

Zero-Order Methods: Random Search

```
Require: Distribution $\nu \in \mathcal{M}(\Theta)$; Number of rollouts $n$;
    Maximum number of iterations $K$
 1: Draw a parameter $\theta_1 = \theta'_1 \sim \nu$
 2: Evaluate $\hat{J}_n(\pi_{\theta_1})$
 3: for $k = 2, 3, \dots, K$ do
 4:     Draw a parameter $\theta'_k \sim \nu$
 5:     Evaluate $\hat{J}_n(\pi_{\theta'_k})$
 6:     if $\hat{J}_n(\pi_{\theta'_k}) > \hat{J}_n(\pi_{\theta_{k-1}})$ then
 7:         $\theta_k \leftarrow \theta'_k$
 8:     else
 9:         $\theta_k \leftarrow \theta_{k-1}$
10:     end if
11: end for
12: return $\pi_{\theta_K}$
```

Zero-Order Methods: Random Search

- We can provide guarantee that RS finds the optimal point, asymptotically.
- RS is not the most efficient way to search a parameter space.
- The way it is presented here does not benefit from all previous evaluations of the function $\hat{J}_n$ when suggesting a new θ'_k.
- That knowledge can be useful by helping us focus on more promising regions of the search space, instead of blindly sampling from the same distribution ν.
- This can be achieved by adaptively changing the policy parameter sampling distribution ν_k to be a function of previous evaluations.

Evolutionary Algorithms

- A large class of optimization methods are inspired by the process of evolution.
- Heritable characteristics of individuals in a population change over generations due to processes such as natural selection.
- The evolution leads to the adaptation of individuals, which means that they become better to live in their habitat.

Evolutionary Process for Optimization

- Identifying a solution to an optimization problem as an individual in a population
- The value of the function to be optimized for a particular solution as the fitness of that individual
- Emulate the evolution:
  - Mutation
  - Reproduction (cross-over)
  - Selection
- There are many variations in how we can do this:
  - Genetic Algorithms
  - Genetic Programming
  - Evolutionary Strategy

Evolutionary Strategy (ES) $(1+1)$

```
Require: Initial point $\theta_0 \in \Theta$; Rollouts $n$; Iterations $K$
Require: Initial standard deviation of mutation operator: $\sigma_1 > 0$
Require: Adaptation parameters: $c_+ > 0$ and $c_- < 0$.
 1: Evaluate $\hat{J}_n(\pi_{\theta_0})$
 2: for $k = 1, 2, \ldots, K$ do
 3:     Draw a perturbation $\eta \sim \mathcal{N}(0, \mathbf{I})$
 4:     $\theta'_k \leftarrow \theta_k + \sigma_k \eta$                          ▷ Mutation
 5:     Evaluate $\hat{J}_n(\pi_{\theta'_k})$
 6:     if $\hat{J}_n(\pi_{\theta'_k}) > \hat{J}_n(\pi_{\theta_k})$ then           ▷ Selection
 7:         $\theta_{k+1} \leftarrow \theta'_k$
 8:         $\sigma_{k+1} \leftarrow \sigma_k e^{c_+}$
 9:     else
10:         $\theta_{k+1} \leftarrow \theta_{k-1}$
11:         $\sigma_{k+1} \leftarrow \sigma_k e^{c_-}$
12:     end if
13: end for
14: return $\pi_{\theta_K}$
```

Evolutionary Strategy (ES) $(1+1)$ and Beyond

- Evolutionary Strategy (ES) $(1+1)$ is one of the simplest evolutionary algorithms
  - Similar to RS, but guided choice of randomness
  - Has some theoretical analysis
- A modification of this algorithm is called ES$(1, \lambda)$ with $\lambda > 1$ being an integer number.
  - The parent θ_k generates λ offsprings:

$$\theta'_{k,j} = \theta_k + \sigma_k \eta_j, \qquad j = 1, \ldots, \lambda.$$

  - The competition would only be between the offsprings $\{\theta'_{k,j}\}_{j=1}^{\lambda}$, and not with the parent. Only one of the λ offsprings gets to the next generation.

Beyond Evolutionary Strategy

- ES does not have any sexual reproduction: each new θ' is based on only a single parent's θ, and not two parents θ_1 and θ_2.
- There are other evolutionary algorithms that have the reproduction component too, e.g., Genetic Algorithm (GA).
- Evolutionary algorithms can be quite complicated algorithms.
  - Many heuristics, inspired by nature.
  - Their performance is often evaluated only empirically.
  - Analyzing them theoretically can be quite complicated.
  - Current available results are limited to simple algorithms, such as ES$(1+1)$, which may not be the best performing ones in practice.

Evolutionary Algorithms and RL

- Studying evolutionary algorithms to solve RL problems is a relatively niche area in the RL community.
- Sometimes (often?), they are not considered as a part of the RL research proper, though there has been a minor shift in the past decade.
- Knowing about them is useful!
- Both evolution and learning have been crucial adaptation mechanisms to get to the point where we have relatively smart species.
- Building AI agents with comparable capabilities to animals may require borrowing ideas from both learning and evolution.
  - Learning: Within the lifespan of the agent
  - Evolution: Across generations of the agents

First-Order Methods for Policy Optimization, and the Policy Gradient

First-Order Methods and the Policy Gradient

- The gradient of $J_\rho(\pi_\theta)$ w.r.t. θ allows us to design first-order optimization methods.
- General recipe: Repeatedly compute

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \nabla_\theta J_\rho(\pi_{\theta k}),$$

  with $\alpha_k > 0$ being the learning rate/step size.
  Q: Why do we have $+$ instead of $-$ in the update rule?
- Potentially more efficient in finding an optimum of the performance compared to zero-order methods.
- Not obvious how to compute the gradient $\nabla_\theta J_\rho(\pi_\theta)$:
  - The performance $J_\rho(\pi_\theta)$ depends on V^{π_θ}.
  - Not a simple function of π_θ.
  - The value function is a complicated function of the policy, reward distribution $\mathcal{R}$ and the transition dynamics $\mathcal{P}$.

Finite Difference Approximation of the Policy Gradient

- Use Finite Difference (FD) approximation of the policy gradient.
- Can be computed using the value of the performance objective itself.
- Recall that given a function $f : \mathbb{R} \to \mathbb{R}$, the FD approximation of the derivative $f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}(x)$ is

$$f'_{\mathsf{FD}}(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x}, \tag{7}$$

where Δx is a small number. This is called the forward difference approximation.

Finite Difference Approximation of the Policy Gradient

- By the Taylor's theorem, assuming twice differentiability, we have

$$f(x+\Delta x) = f(x) + f'(x)\Delta x + f''(z)|_{x<z<x+\Delta x}\frac{(\Delta x)^2}{2!}.$$

  Therefore,

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} - f''(z)|_{x<z<x+\Delta x}\frac{(\Delta x)^2}{2!}.$$

- The error between the FD approximation (7) and $f'(x)$ is

$$\left| f''(z)\Big|_{x<z<x+\Delta x}\frac{(\Delta x)^2}{2!} \right|,$$

  that is, $O((\Delta x)^2)$.

Finite Difference Approximation of the Policy Gradient

- Central difference approximation:

$$f'_{\mathsf{FD}}(x) = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}.$$

- Error is $O((\Delta x)^3)$.

Finite Difference Approximation of the Policy Gradient

To compute the gradient of $J_\rho(\pi_\theta)$ w.r.t. $\theta \in \mathbb{R}^p$, we need to compute $2p$ evaluations of J_ρ:

$$\nabla_\theta J_\rho(\pi_\theta) \approx \nabla_\theta^{\text{(FD)}} J_\rho(\pi_\theta) = \begin{bmatrix} \frac{J_\rho(\theta+\varepsilon e_1)-J_\rho(\theta-\varepsilon e_1)}{2\varepsilon} \\ \vdots \\ \frac{J_\rho(\theta+\varepsilon e_i)-J_\rho(\theta-\varepsilon e_i)}{2\varepsilon} \\ \vdots \\ \frac{J_\rho(\theta+\varepsilon e_p)-J_\rho(\theta-\varepsilon e_p)}{2\varepsilon} \end{bmatrix},$$

where e_i is a unit vector along dimension i of $\mathbb{R}^p$.

Finite Difference Approximation of the Policy Gradient

- We cannot directly compute $J_\rho(\pi_\theta)$.
- We can only compute $\hat{J}_n(\pi_\theta)$ using rollouts.
- Replace each J_ρ above with their corresponding $\hat{J}_n$.
- This requires $2pn$ rollouts in total.
- Given the approximated gradient, which has error caused by both the FD approximation and using $\hat{J}_n$ instead of J_ρ, we may use the gradient ascent to move towards higher value of $J_\rho(\pi_\theta)$:

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \nabla_\theta^{(\mathsf{FD})} \hat{J}_n(\pi_{\theta_k}). \tag{8}$$

- Even though this is a feasible approach, we can compute the gradient more elegantly.

Policy Gradient for the Immediate Reward Problem

- Suppose that we want to find a policy $\pi_\theta : \mathcal{X} \to \mathcal{M}(\mathcal{A})$ with $\theta \in \mathbb{R}^p$ that maximizes the performance for the immediate reward problem.
- Recall that the interaction protocol is
  - At episode t, $X_t \sim \rho \sim \mathcal{M}(\mathcal{X})$
  - The agent chooses action $A_t \sim \pi_\theta(\cdot|X_t)$
  - The agent receives reward $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$.
  - The agent starts the new (independent) episode $t + 1$.
- This is an RL setting as ρ and $\mathcal{R}$ are not directly available to the agent, but only through samples.

Performance Measure

- The performance measure is
$$J_\rho(\pi_\theta) = \int V^{\pi_\theta}(x)\mathrm{d}\rho(x) = \int r^{\pi_\theta}(x)\mathrm{d}\rho(x)$$
$$= \int r(x,a)\pi_\theta(a|x)\mathrm{d}\rho(x)\mathrm{d}a,$$
as the value function V^{π_θ} for the immediate reward problem is the same as r^{π_θ}.
- We consider the action space to be continuous and we assume that $\pi_\theta(\cdot|x)$ provides a density over the state space.
- If we had a discrete action space, we would have
$$\int_{\mathcal{X}} \mathrm{d}\rho(x) \sum_{a\in\mathcal{A}} r(x,a)\pi_\theta(a|x).$$
- We may switch back and forth between continuous and discrete action spaces.

Policy Gradient

- The gradient of $J_\rho(\pi_\theta)$ w.r.t. θ:

$$\begin{aligned}\nabla_\theta J_\rho(\pi_\theta) &= \int r(x,a)\nabla_\theta \pi_\theta(a|x)\mathrm{d}\rho(x)\mathrm{d}a \\ &= \int \mathrm{d}\rho(x) \int r(x,a)\nabla_\theta \pi_\theta(a|x)\mathrm{d}a \\ &= \mathbb{E}_{X\sim\rho}\left[\int r(X,a)\nabla_\theta \pi_\theta(a|X)\mathrm{d}a\right]. \qquad (9)\end{aligned}$$

- For discrete action spaces, the inner integral becomes $\sum_{a\in\mathcal{A}} r(x,a)\nabla_\theta \pi_\theta(a|x)$.
- We call $\nabla_\theta J_\rho(\pi_\theta)$ the Policy Gradient (PG).

Improving Performance Measure using Policy Gradient

If we can compute PG, we can update the policy parameters, using the gradient ascent method:

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \nabla_\theta J_\rho(\pi_{\theta_k}), \tag{10}$$

similar to what we have done using the FD approximation (8).

Computing the Policy Gradient

$$\nabla_\theta J_\rho(\pi_\theta) = \int r(x,a) \nabla_\theta \pi_\theta(a|x) \mathrm{d}\rho(x) \mathrm{d}a$$

- How can we compute this gradient?
- We build this gradually in several steps.
- At each step, we relax some assumptions until we get to a procedure that can use the data available by the interaction protocol above.

Computing the Policy Gradient – Known ρ and r

$$\nabla_\theta J_\rho(\pi_\theta) = \int r(x,a) \nabla_\theta \pi_\theta(a|x) \mathrm{d}\rho(x) \mathrm{d}a$$

- If we know ρ and r, we have all the necessary information for computing the gradient.
  - For each $x \in \mathcal{X}$, we compute the summation (or integral) over all $a \in \mathcal{A}$ of $r(x,a)\nabla_\theta \pi_\theta(a|x)$.
  - We weigh that term proportional to $\rho(x)$.
  - Take average over all x.
- But this is not the RL setting described as the interaction protocol at the beginning of the section.

Computing the Policy Gradient – Known r, unknown ρ

- Assume that r is known, but ρ can only be sampled.
- Approximately solve this problem by sampling $X_i \sim \rho$ $(i = 1, \ldots, n)$ and computing

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{a \in \mathcal{A}} r(X_i, a) \nabla_\theta \pi_\theta(a|X_i). \tag{11}$$

or

$$\frac{1}{n} \sum_{i=1}^{n} \int r(X_i, a) \nabla_\theta \pi_\theta(a|X_i) \mathrm{d}a.$$

As $X_i \sim \rho$, this is an unbiased estimate of

$$\nabla_\theta J_\rho(\pi_\theta) = \mathbb{E}_{X \sim \rho}\left[\sum_{a \in \mathcal{A}} r(X, a) \nabla_\theta \pi_\theta(a|X)\right]$$

or $\mathbb{E}_{X \sim \rho}\left[\int r(x, a) \nabla_\theta \pi_\theta(a|x) \mathrm{d}a\right]$ (continuous).

Computing the Policy Gradient – Known r, unknown ρ

- As $X_i \sim \rho$, this is an unbiased estimate of

$$\nabla_\theta J_\rho(\pi_\theta) = \mathbb{E}_{X\sim\rho}\left[\sum_{a\in\mathcal{A}} r(X,a)\nabla_\theta \pi_\theta(a|X)\right]$$

or

$$\mathbb{E}_{X\sim\rho}\left[\int r(x,a)\nabla_\theta \pi_\theta(a|x)\mathrm{d}a\right].$$

- This is still not feasible if r is unknown in our interaction protocol:
  - the agent is initialized at state x
  - it has to choose its action according to $A \sim \pi_\theta(\cdot|x)$.

Computing the Policy Gradient – Unknown r and ρ

- The term
$$\sum_{a \in \mathcal{A}} r(x,a) \nabla_\theta \pi_\theta(a|x)$$
can be interpreted as the expectation of
$$r(x,A) \nabla_\theta \pi_\theta(A|x)$$
when A is coming from a uniform distribution with $q(a) = \frac{1}{|\mathcal{A}|}$ (for $a \in \mathcal{A}$).
- We have
$$\sum_{a \in \mathcal{A}} r(x,a) \nabla_\theta \pi_\theta(a|x) = |\mathcal{A}| \sum_{a \in \mathcal{A}} q(a) r(x,a) \nabla_\theta \pi_\theta(a|x). \quad (12)$$
- Similar for the continuous case.

Computing the Policy Gradient – Unknown r and ρ

- If the actions were coming from a uniform distribution, we could easily form an empirical estimate of these terms.
- But the actions in the interaction protocol comes from distribution $\pi_\theta(\cdot|x)$, which in general is different distribution than a uniform one.
- It is as if we have some form of off-policy sampling scenario in the distribution of actions.
- Some approaches to deal with it:
  - Estimate $\hat{r} \approx r$ using data (model-based approach).
  - Modify $r(x, A)\nabla_\theta \pi_\theta(A|x)$ to a quantity that can be estimated from data.

Computing the Policy Gradient – Unknown r and ρ

- We need a mathematical fact from calculus.
- For a function $f : \mathbb{R} \to \mathbb{R}$, we have

$$\frac{\mathrm{d} \log f(x)}{\mathrm{d}x} = \frac{\frac{\mathrm{d}f}{\mathrm{d}x}(x)}{f(x)},$$

or more generally, for a function $f : \mathbb{R}^p \to \mathbb{R}$,

$$\nabla_x \log f(x) = \frac{\nabla_x f(x)}{f(x)}.$$

- This means that $\nabla_x f(x) = f(x) \nabla_x \log f(x)$.

Computing the Policy Gradient – Unknown r and ρ

- Using this fact, we get

$$\begin{aligned}\int r(x,a)\nabla_\theta \pi_\theta(a|x)\mathrm{d}a &= \int r(x,a)\nabla_\theta \log \pi_\theta(a|x) \pi_\theta(a|x)\mathrm{d}a \\ &= \mathbb{E}_{A\sim\pi_\theta(\cdot|x)}\left[r(x,A)\nabla_\theta \log \pi_\theta(A|x)\right].\end{aligned}$$

- The desired quantity can be written as the expectation of

$$r(x,A)\nabla_\theta \log \pi_\theta(A|x)$$

  when $A \sim \pi_\theta(\cdot|x)$.
- Interestingly, the sampling distribution is the same as the one agent uses to choose its actions.
- We are in the on-policy sampling scenario over the choice of actions.

Computing the Policy Gradient – Unknown r and ρ

- If $X \sim \rho$ and $A \sim \pi_\theta(\cdot|X)$, the random variable
$$r(X, A)\nabla_\theta \log \pi_\theta(A|X) \tag{13}$$
is an unbiased estimate of $\nabla_\theta J_\rho(\pi_\theta)$, i.e.,
$$\begin{aligned}\nabla_\theta J_\rho(\pi_\theta) &= \mathbb{E}\left[r(X, A)\nabla_\theta \log \pi_\theta(A|X)\right] \\ &= \mathbb{E}_{X\sim\rho}\left[\mathbb{E}_{A\sim\pi_\theta(\cdot|X)}\left[r(X, A)\nabla_\theta \log \pi_\theta(A|X) \mid X\right]\right].\end{aligned} \tag{14}$$
- We can estimate the gradient of the performance w.r.t. the parameters of the policy using data available through the interaction of the agent with its environment.
- We may use this estimate in (10) to update the policy parameters using unbiased but noisy estimate of the gradient.
- This makes it a stochastic gradient ascent.

Policy Gradient: Sources of Variance

Two Sources of Variance

$$r(X, A)\nabla_\theta \log \pi_\theta(A|X)$$

vs.

$$\mathbb{E}_{X\sim\rho}\left[\mathbb{E}_{A\sim\pi_\theta(\cdot|X)}\left[r(X, A)\nabla_\theta \log \pi_\theta(A|X) \mid X\right]\right]$$

- The r.v. is an unbiased estimate of the gradient,
- but it has variance due to two sources of randomness:
  - Variance of estimating

    $$g(x;\theta) \triangleq \mathbb{E}_{A\sim\pi_\theta(\cdot|X)}\left[r(X, A)\nabla_\theta \log \pi_\theta(A|X) \mid X = x\right]$$

    with a single sample $r(X, A)\nabla_\theta \log \pi_\theta(A|X)$.
  - Variance of estimating $\mathbb{E}_{X\sim\rho}\left[g(X;\theta)\right]$ using a single sample.

Two Sources of Variance

$$r(X, A) \nabla_\theta \log \pi_\theta(A|X)$$

- The variance along the i-th dimension of this r.v. is

$$\mathrm{Var}\left[r(X, A) \frac{\partial \log \pi_\theta(A|X)}{\partial \theta_i} \right] =$$
$$\mathbb{E}_{X \sim \rho}\left[\mathrm{Var}\left[r(X, A) \frac{\partial \log \pi_\theta(A|X)}{\partial \theta_i} \mid X \right] \right] + \mathrm{Var}_{X \sim \rho}\left[g_i(X; \theta) \right]. \tag{15}$$

Two Sources of Variance

- Let us define

$$g(x;\theta) = \mathbb{E}_{A \sim \pi_\theta(\cdot|x)} \left[r(x, A) \nabla_\theta \log \pi_\theta(A|x) \right]. \qquad (16)$$

- The function $g : \mathcal{X} \times \Theta \to \mathbb{R}^p$ is the gradient of r^{π_θ} w.r.t. θ at state x, and is a p-dimensional vector.
- If we knew $r(x, a)$ and we could compute $g(x;\theta)$, we wouldn't have the first source of variance, but we still would have the second one. In that case, the variance would be

$$\mathrm{Var}_{X \sim \rho} \left[g_i(X;\theta) \right].$$

- These two sources of variance make our estimate of the gradient inaccurate.
- There are ways to reduce them.

Variance Reduction – Randomness of States

- Suppose we can compute $g(x;\theta)$ exactly for any given $x \in \mathcal{X}$.
- Instead of a single sample $g(X_1;\theta)$, we use multiple independent samples $X_1, \dots, X_n \sim \rho$ to estimate the PG:

$$\begin{aligned}\nabla_\theta J_\rho(\pi_\theta) &\approx \frac{1}{n}\sum_{i=1}^{n} g(X_i;\theta) \\ &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{A\sim\pi_\theta(\cdot|X_i)}\left[r(X_i, A)\nabla_\theta \log \pi_\theta(A|X_i)\right].\end{aligned}$$

- The variance of this estimator, along dimension i, is

$$\frac{1}{n}\mathrm{Var}_{X\sim\rho}\left[g_i(X;\theta)\right].$$

  As $n \to \infty$, the variance goes to zero. This leads to more accurate estimate of the PG, hence more accurate update of the policy.

Variance Reduction – Randomness of Actions

Consider the variance of estimating $g(x;\theta)$ (16) using a single sample $r(x,A)\nabla_\theta \log \pi_\theta(A|x)$ with $A \sim \pi_\theta(\cdot|x)$.
We make an important observation: Let $b : \mathcal{X} \to \mathbb{R}$ be a state-dependent function. For each dimension i, we have

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} b(x) \mid x\right] &= \int \pi_\theta(a|x) \frac{\partial \log \pi_\theta(a|x)}{\partial \theta_i} b(x) \mathrm{d}a \\
&= \int \frac{\partial \pi_\theta(a|x)}{\partial \theta_i} b(x) \mathrm{d}a \\
&= b(x) \int \frac{\partial \pi_\theta(a|x)}{\partial \theta_i} \mathrm{d}a \\
&= b(x) \frac{\partial}{\partial \theta_i} \underbrace{\int \pi_\theta(a|x) \mathrm{d}a}_{=1} = 0.
\end{aligned}$$

Variance Reduction – Randomness of Actions

- This shows that

$$\mathbb{E}\left[\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} r(x, A) \mid x\right] = \mathbb{E}\left[\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i}\left(r(x, A) + b(x)\right) \mid x\right]. \tag{17}$$

- Adding a state-dependent function $b : \mathcal{X} \to \mathbb{R}$ to $r(x, a)$ does not change the expectation.
- But it may change the variance!

Variance Reduction – Randomness of Actions

- For each dimension i of the PG, we can use a different state-dependent function.
- For any state-dependent function $b : \mathcal{X} \to \mathbb{R}^p$, the PG (14) is

$$\nabla_\theta J_\rho(\pi_\theta) = \mathbb{E}\left[r(X,A)\nabla_\theta \log \pi_\theta(A|X)\right] = \mathbb{E}\left[(r(X,A)\mathbf{1} + b(X)) \odot \nabla_\theta \log \pi_\theta(A|X)\right],$$

  where $\mathbf{1}$ is a p-dimensional vector with all components equal to 1, and $\odot$ is a pointwise (Hadamard) product of two vectors, i.e., for $u, v \in \mathbb{R}^p$, $[u \odot v]_i = u_i v_i$.
- If we simply choose a scalar function b, which is often the case in practice, we have

$$\nabla_\theta J_\rho(\pi_\theta) = \mathbb{E}\left[(r(X,A) + b(X))\,\nabla_\theta \log \pi_\theta(A|X)\right].$$

- The function b is called the baseline.

Variance Reduction – Randomness of Actions – Baseline

- The baseline can be used in order to minimize the variation of p-dimensional random vector.
- We would like to find a function $b : \mathcal{X} \to \mathbb{R}^p$ such that for all $x \in \mathcal{X}$,

$$\min_b \sum_{i=1}^{p} \operatorname{Var}\left[\left(r(x, A) + b_i(x)\right) \frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} \mid x \right] = \operatorname{Tr} \operatorname{Cov}\left(\left(r(x, A)\mathbf{1} + b(x)\right) \odot \nabla_\theta \log \pi_\theta(A|x) \mid x \right)$$

Variance Reduction – Randomness of Actions – Baseline

After some derivations, we get that the minimizer of the variance is

$$b_i(x) = \frac{-\mathbb{E}\left[r(x, A) \left(\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} \right)^2 \mid x \right]}{\mathbb{E}\left[\left(\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} \right)^2 \mid x \right]}. \tag{18}$$

We could choose a single scalar function $b : \mathcal{X} \to \mathbb{R}$ instead. In that case, the solution would be

$$b(x) = \frac{-\mathbb{E}\left[r(x, A) \left\| \nabla_\theta \log \pi_\theta(A|x) \right\|_2^2 \mid x \right]}{\mathbb{E}\left[\left\| \nabla_\theta \log \pi_\theta(A|x) \right\|_2^2 \mid x \right]}.$$

Variance Reduction – Randomness of Actions – Baseline

- Practitioners often choose a baseline that is the average reward at that state:

  $$b(x) = -r^{\pi_\theta}(x) = -\mathbb{E}_{A \sim \pi_\theta(\cdot|x)}\left[r(x, A)\right].$$

  This is not a minimum variance baseline.
- The conventional wisdom is that it is better to have a lower variance estimator for PG.
- But is this actually correct? Recent works suggest that this may not be the case. See [Chung et al., 2021; Mei et al., 2022].

Policy Gradient for Continuing Tasks

Policy Gradient for Continuing Tasks

- We derive the PG for continuing tasks.
- The difference with the immediate reward case is that the performance $J_\rho(\pi_\theta)$ depends on the dynamics $\mathcal{P}^{\pi_\theta}$ too.
- As we change θ, the dynamics $\mathcal{P}^{\pi_\theta}$ changes as well.
- This seems to complicate the gradient computation.
- It turns out that despite this challenge, the PG can be written in an elegant, and relatively easy to compute, form.

Discounted Future-State Distribution

- New notations to present the results more compactly.
- Recall that $\mathcal{P}^\pi(\cdot|x;k) = \mathcal{P}^\pi(\cdot|x)^k$ is the probability distribution of following policy π for $k \geq 0$ steps.
- We introduce discounted future-state distribution of starting from $x \in \mathcal{X}$ and following π as

$$\rho_\gamma^\pi(\cdot|x) = \rho_\gamma(\cdot|x;\mathcal{P}^\pi) \triangleq (1-\gamma)\sum_{k\geq 0}\gamma^k \mathcal{P}^\pi(\cdot|x;k). \qquad (19)$$

- It is easy to verify that $\rho_\gamma^\pi(\cdot|x)$ is a valid probability distribution (for example, $\rho_\gamma^\pi(\mathcal{X}|x) = 1$).

Discounted Future-State Distribution

- The relevant of this distribution becomes more clear if we note that

$$\begin{aligned} V^\pi(x) &= \mathbb{E}\left[\sum_{t\geq 0} \gamma^t R_t | X_0 = x\right] \\ &= \sum_{t\geq 0} \gamma^t \mathbb{E}\left[R_t | X_0 = x\right] \\ &= \sum_{t\geq 0} \gamma^t \int \mathcal{P}^\pi(\mathrm{d}x'|x;t) r(x') \\ &= \frac{1}{1-\gamma} \int \rho_\gamma^\pi(\mathrm{d}x'|x) r(x') = \frac{1}{1-\gamma} \mathbb{E}_{X' \sim \rho_\gamma^\pi(\cdot|x)}\left[r(X')\right]. \end{aligned}$$

- The value function at a state x is the expected reward when X' is distributed according to $\rho_\gamma^\pi(\cdot|x)$.

Discounted Future-State Distribution

- Interpretation: The agent starts from state x and at each time step, it decides to follow π with probability γ or terminates the episode with probability $1 - \gamma$.
- We can also define discounted future-state distribution of starting from ρ and following π as

$$\rho_\gamma^\pi(\cdot) = \rho_\gamma(\cdot|\mathcal{P}^\pi) \triangleq \int \rho_\gamma(\cdot|x; \mathcal{P}^\pi)\mathrm{d}\rho(x).$$

- The performance measure $J(\pi_\theta)$ (3) is

$$J(\pi_\theta) = \mathbb{E}_{X\sim\rho}\left[V^{\pi_\theta}(X)\right] = \frac{1}{1-\gamma}\mathbb{E}_{X\sim\rho_\gamma^\pi}\left[r(X)\right].$$

Policy Gradient Theorem

Theorem (Policy Gradient Theorem – Sutton et al. 2000)

Assume that π_θ is differentiable w.r.t. $\theta \in \Theta$. We have

$$\begin{aligned}
&\nabla_\theta J_\rho(\pi_\theta) = \\
&\sum_{k \geq 0} \gamma^k \int \mathrm{d}\rho(x) \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x;k) \int \nabla_\theta \pi_\theta(a'|x') Q^{\pi_\theta}(x',a') \mathrm{d}a' = \\
&\frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x) \int \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a) \mathrm{d}a = \\
&\frac{1}{1-\gamma} \mathbb{E}_{X \sim \rho_\gamma^{\pi_\theta}, A \sim \pi_\theta(\cdot|X)} \left[\nabla_\theta \log \pi_\theta(A|X) Q^{\pi_\theta}(X,A) \right].
\end{aligned}$$

Policy Gradient Theorem – Proof

We write the value function at state $x \in \mathcal{X}$ as the expected value of the action-value function, i.e.,

$$V^{\pi_\theta}(x) = \int \pi_\theta(a|x) Q^{\pi_\theta}(x,a) \mathrm{d}a.$$

We take its derivative w.r.t. θ and use the product rule of differentiation to get

$$\nabla_\theta V^{\pi_\theta}(x) = \int \left[\nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a) + \pi_\theta(a|x) \nabla_\theta Q^{\pi_\theta}(x,a) \right] \mathrm{d}a. \tag{20}$$

Policy Gradient Theorem – Proof

As $Q^{\pi_\theta}(x,a) = r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)V^{\pi_\theta}(x')$,

$$\nabla_\theta Q^{\pi_\theta}(x,a) = \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\nabla_\theta V^{\pi_\theta}(x').$$

This alongside with (20) gives us the recursive Bellman-like equation for the gradient of $V^{\pi_\theta}(x)$:

$$\nabla_\theta V^{\pi_\theta}(x) = \int \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a)\mathrm{d}a + \gamma \int \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x)\nabla_\theta V^{\pi_\theta}(x'). \tag{21}$$

Policy Gradient Theorem – Proof

Expanding $\nabla_\theta V^{\pi_\theta}(x')$, we get that

$$\nabla_\theta V^{\pi_\theta}(x) = \int \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a)\mathrm{d}a + \gamma \int \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x) \Bigg[\nabla_\theta \pi_\theta(a'|x') Q^{\pi_\theta}(x',a')\mathrm{d}a' + \gamma \int \mathcal{P}^{\pi_\theta}(\mathrm{d}x''|x') \nabla_\theta V^{\pi_\theta}(x'') \Bigg].$$

Following this pattern recursively, we get that

$$\begin{aligned}\nabla_\theta V^{\pi_\theta}(x) &= \sum_{k\geq 0} \gamma^k \int \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x;k) \int \nabla_\theta \pi_\theta(a'|x') Q^{\pi_\theta}(x',a')\mathrm{d}a' \\ &= \frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x'|x) \int \nabla_\theta \pi_\theta(a'|x') Q^{\pi_\theta}(x',a')\mathrm{d}a'.\end{aligned}$$

Policy Gradient Theorem – Proof

Also since $\nabla_\theta \pi_\theta(a'|x') = \pi_\theta(a'|x') \nabla_\theta \log \pi_\theta(a'|x')$, we can write the gradient as

$$
\begin{aligned}
&\nabla_\theta V^{\pi_\theta}(x) = \\
&\frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x'|x) \int \pi_\theta(a'|x') \nabla_\theta \log \pi_\theta(a'|x') Q^{\pi_\theta}(x', a') \mathrm{d}a' = \\
&\frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x'|x) \mathbb{E}_{A' \sim \pi_\theta(\cdot|X')} \left[\nabla_\theta \log \pi_\theta(A'|X') Q^{\pi_\theta}(X', A') \right].
\end{aligned}
$$

Policy Gradient Theorem – Proof

As $J_\rho(\pi_\theta) = \int V^{\pi_\theta}(x) \mathrm{d}\rho(x)$, taking the average of x w.r.t. ρ, we get that

$$\begin{aligned}\nabla_\theta J_\rho(\pi_\theta) &= \frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x) \int \pi_\theta(a|x) \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a) \mathrm{d}a \\ &= \frac{1}{1-\gamma} \mathbb{E}_{\substack{X \sim \rho_\gamma^{\pi_\theta} \\ A \sim \pi_\theta(\cdot|X)}} \left[\nabla_\theta \log \pi_\theta(A|X) Q^{\pi_\theta}(X,A) \right],\end{aligned}$$

which is the desired result.

Policy Gradient Theorem

- This theorem provides an elegant formula for the PG.
- It relates the PG to the discounted future-state distribution $\rho_\gamma^{\pi_\theta}$, the action-value function $Q^{\pi_\theta}(x, a)$, and the gradient of π_θ.
- We can now improve the policy by performing gradient ascent

$$\begin{aligned}\theta_{k+1} \leftarrow & \theta_k + \alpha_k \nabla_\theta J_\rho(\pi_{\theta k}) = \\ & \theta_k + \alpha_k \mathbb{E}_{\substack{X \sim \rho_\gamma^{\pi_\theta} \\ A \sim \pi_\theta(\cdot|X)}} \left[\nabla_\theta \log \pi_\theta(A|X) Q^{\pi_\theta}(X, A) \right].\end{aligned}$$

- Two questions:
  - How can we estimate the PG using data available in the RL setting?
  - Is performing gradient ascent the best way to update the policy? How should we choose the learning rate/step size α_k?

Policy Gradient Theorem

- To compute the PG in the RL setting, we have to estimate it using samples. If we get
  - a state X sampled from $\rho_\gamma^{\pi_\theta}$,
  - an action A sampled from $\pi_\theta(\cdot|X)$, and
  - know action-value $Q^{\pi_\theta}(X, A)$,

  the random variables

$$\nabla_\theta \log \pi_\theta(A|X) Q^{\pi_\theta}(X, A)$$

  is an unbiased estimate of the PG (cf. (13)).
- We can then use it in an SGD scheme to improve the policy.

Sampling X from $\rho_\gamma^{\pi_\theta}$

Sampling from $\rho_\gamma^{\pi_\theta}$ is relatively straightforward in the on-policy sampling scenario when the agent follows π_θ.

- The agent starts an episode from $X_0 \sim \rho$ and follows π_θ.
- We get a sequence of states $X_0, X_1, \dots$.
- These would be samples from $\int \mathrm{d}\rho(x)\mathcal{P}^{\pi_\theta}(\cdot|x;k)$ for $k = 0, 1, \dots$.
- The distribution $\rho_\gamma^{\pi_\theta}$, however, has a γ^k factor for the k-th time step, see (19).
- Its effect is that the contribution to the gradient from X_k, which is

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(A|X) Q^{\pi_\theta}(X,A)\right] = \int \pi_\theta(a|x) \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a) \mathrm{d}a,$$

should be weighted by γ^k.

Sampling X from $\rho_\gamma^{\pi_\theta}$

- Another way to directly sample from $\rho_\gamma^{\pi_\theta}$ is to follow π, but at each step terminate the episode with probability $1 - \gamma$.

Sampling A from $\pi_\theta(\cdot|X)$

An action A sampled from $\pi_\theta(\cdot|X)$ is automatically generated when the agent follows policy π_θ (on-policy).

Computation of $Q^{\pi_\theta}(X, A)$

- The remaining issue is the computation of $Q^{\pi_\theta}(X, A)$ for $X \sim \rho_\gamma^{\pi_\theta}$ and $A \sim \pi_\theta(\cdot|X)$ using data.
- This is essentially a PE problem, and we may use various action-value function estimators that we have developed so far.

Computation of $Q^{\pi_\theta}(X, A)$

- A simple approach: Use the MC estimate of $Q^{\pi_\theta}(X, A)$.
- In the on-policy setting when the agent follows π_θ, it generates the sequence $X_0, A_0, R_0, X_1, A_1, R_1, \ldots$ with $A_t \sim \pi_\theta(\cdot|X_t)$.
- The return $G_t^\pi = \sum_{k \geq t} \gamma^{k-t} R_k$ is an unbiased estimate of $Q^{\pi_\theta}(X_t, A_t)$.
- We replace the action-value function at that state-action with this return from time t onward.

Computation of $Q^{\pi_\theta}(X, A)$

- The return, however, is a high variance estimate of the action-value function.
- We can use a baseline to reduce the variance of this MC estimate.
- This approach is known as the REINFORCE algorithm by Williams [1992].[1]

[1] REINFORCE stands for REward Increment $\times$ Nonnegative Factor $\times$ Offset Reinforcement $\times$ Characteristic Eligibility.

Computation of $Q^{\pi_\theta}(X, A)$ – Actor-Critic Methods

- Another approach is to use an action-value function estimator instead.
  - TD methods
  - LSTD
  - Fitted Value Iteration (for PE, and not for Control)
- Such a method is called actor-critic method
  - The actor refers to the policy (and often PG method to improve it)
  - The critic refers to the value function estimator used to criticize the policy (actor).

Computation of $Q^{\pi_\theta}(X, A)$ – Actor-Critic Methods

- The use of a critic, however, may induce a bias as $\mathbb{E}\left[\hat{Q}^{\pi_\theta}(X, A) | X, A\right]$ may be different from $Q^{\pi_\theta}(X, A)$, especially if we use a TD method (which introduces bias because of bootstrapping) or a function approximator (for large state-action spaces).
- Such a method would explicitly represent both policy and value function.
- Actor-critic methods bring together some of the benefits of both value-based and policy search methods.

Summary

- Policy Search Methods: Explicit representation of the policy and searching within the policy space
- The search might be guided by the zero-order or first-order methods
- We may sometimes constraint the change of the policy update
    - Policy gradient is only a local information. We should not make a large move.
- We will have a guest lecturer to discuss the question "Is performing gradient ascent the best way to update the policy?", asked a few slides ago.

References

Wesley Chung, Valentin Thomas, Marlos C Machado, and Nicolas Le Roux. Beyond variance reduction: Understanding the true impact of baselines on policy optimization. In *International Conference on Machine Learning (ICML)*, 2021.

Jincheng Mei, Wesley Chung, Valentin Thomas, Bo Dai, Csaba Szepesvari, and Dale Schuurmans. The role of baselines in policy gradient optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS - 12)*, 2000.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8 (3-4):229–256, 1992.