Policy Gradient Algorithms

Ashwin Rao

ICME, Stanford University

Overview

Why do we care about Policy Gradient (PG)?

Why do we care about Policy Gradient (PG)?

- Let us review how we got here

Why do we care about Policy Gradient (PG)?

- Let us review how we got here
- We started with Markov Decision Processes and Bellman Equations

Why do we care about Policy Gradient (PG)?

- Let us review how we got here
- We started with Markov Decision Processes and Bellman Equations
- Next we studied several variants of DP and RL algorithms

Why do we care about Policy Gradient (PG)?

- Let us review how we got here
- We started with Markov Decision Processes and Bellman Equations
- Next we studied several variants of DP and RL algorithms
- We noted that the idea of *Generalized Policy Iteration* (GPI) is key

Why do we care about Policy Gradient (PG)?

- Let us review how we got here
- We started with Markov Decision Processes and Bellman Equations
- Next we studied several variants of DP and RL algorithms
- We noted that the idea of *Generalized Policy Iteration* (GPI) is key
- Policy Improvement step: $\pi(s, a)$ derived from $\text{argmax}_a Q(s, a)$

Why do we care about Policy Gradient (PG)?

- Let us review how we got here
- We started with Markov Decision Processes and Bellman Equations
- Next we studied several variants of DP and RL algorithms
- We noted that the idea of *Generalized Policy Iteration* (GPI) is key
- Policy Improvement step: $\pi(s, a)$ derived from $\text{argmax}_a Q(s, a)$
- How do we do argmax when action space is large or continuous?

Why do we care about Policy Gradient (PG)?

- Let us review how we got here
- We started with Markov Decision Processes and Bellman Equations
- Next we studied several variants of DP and RL algorithms
- We noted that the idea of *Generalized Policy Iteration* (GPI) is key
- Policy Improvement step: $\pi(s, a)$ derived from $\text{argmax}_a\, Q(s, a)$
- How do we do argmax when action space is large or continuous?
- Idea: Do Policy Improvement step with a Gradient Ascent instead

"Policy Improvement with a Gradient Ascent??"

“Policy Improvement with a Gradient Ascent??”

- We want to find the Policy that fetches the “Best Expected Returns”

"Policy Improvement with a Gradient Ascent??"

- We want to find the Policy that fetches the "Best Expected Returns"
- Gradient Ascent on "Expected Returns" w.r.t params of Policy func

“Policy Improvement with a Gradient Ascent??”

- We want to find the Policy that fetches the “Best Expected Returns”
- Gradient Ascent on “Expected Returns” w.r.t params of Policy func
- So we need a func approx for (stochastic) Policy Func: $\pi(s, a; \boldsymbol{\theta})$

"Policy Improvement with a Gradient Ascent??"

- We want to find the Policy that fetches the "Best Expected Returns"
- Gradient Ascent on "Expected Returns" w.r.t params of Policy func
- So we need a func approx for (stochastic) Policy Func: $\pi(s, a; \boldsymbol{\theta})$
- In addition to the usual func approx for Action Value Func: $Q(s, a; \boldsymbol{w})$

"Policy Improvement with a Gradient Ascent??"

- We want to find the Policy that fetches the "Best Expected Returns"
- Gradient Ascent on "Expected Returns" w.r.t params of Policy func
- So we need a func approx for (stochastic) Policy Func: $\pi(s, a; \boldsymbol{\theta})$
- In addition to the usual func approx for Action Value Func: $Q(s, a; \boldsymbol{w})$
- $\pi(s, a; \boldsymbol{\theta})$ called *Actor* and $Q(s, a; \boldsymbol{w})$ called *Critic*

"Policy Improvement with a Gradient Ascent??"

- We want to find the Policy that fetches the "Best Expected Returns"
- Gradient Ascent on "Expected Returns" w.r.t params of Policy func
- So we need a func approx for (stochastic) Policy Func: $\pi(s, a; \boldsymbol{\theta})$
- In addition to the usual func approx for Action Value Func: $Q(s, a; \boldsymbol{w})$
- $\pi(s, a; \boldsymbol{\theta})$ called *Actor* and $Q(s, a; \boldsymbol{w})$ called *Critic*
- Critic parameters $\boldsymbol{w}$ are optimized w.r.t $Q(s, a; \boldsymbol{w})$ loss function min

"Policy Improvement with a Gradient Ascent??"

- We want to find the Policy that fetches the "Best Expected Returns"
- Gradient Ascent on "Expected Returns" w.r.t params of Policy func
- So we need a func approx for (stochastic) Policy Func: $\pi(s, a; \boldsymbol{\theta})$
- In addition to the usual func approx for Action Value Func: $Q(s, a; \boldsymbol{w})$
- $\pi(s, a; \boldsymbol{\theta})$ called *Actor* and $Q(s, a; \boldsymbol{w})$ called *Critic*
- Critic parameters $\boldsymbol{w}$ are optimized w.r.t $Q(s, a; \boldsymbol{w})$ loss function min
- Actor parameters $\boldsymbol{\theta}$ are optimized w.r.t Expected Returns max

“Policy Improvement with a Gradient Ascent??”

- We want to find the Policy that fetches the “Best Expected Returns”
- Gradient Ascent on “Expected Returns” w.r.t params of Policy func
- So we need a func approx for (stochastic) Policy Func: $\pi(s, a; \boldsymbol{\theta})$
- In addition to the usual func approx for Action Value Func: $Q(s, a; \boldsymbol{w})$
- $\pi(s, a; \boldsymbol{\theta})$ called *Actor* and $Q(s, a; \boldsymbol{w})$ called *Critic*
- Critic parameters $\boldsymbol{w}$ are optimized w.r.t $Q(s, a; \boldsymbol{w})$ loss function min
- Actor parameters $\boldsymbol{\theta}$ are optimized w.r.t Expected Returns max
- We need to formally define “Expected Returns”

"Policy Improvement with a Gradient Ascent??"

- We want to find the Policy that fetches the "Best Expected Returns"
- Gradient Ascent on "Expected Returns" w.r.t params of Policy func
- So we need a func approx for (stochastic) Policy Func: $\pi(s, a; \boldsymbol{\theta})$
- In addition to the usual func approx for Action Value Func: $Q(s, a; \boldsymbol{w})$
- $\pi(s, a; \boldsymbol{\theta})$ called *Actor* and $Q(s, a; \boldsymbol{w})$ called *Critic*
- Critic parameters $\boldsymbol{w}$ are optimized w.r.t $Q(s, a; \boldsymbol{w})$ loss function min
- Actor parameters $\boldsymbol{\theta}$ are optimized w.r.t Expected Returns max
- We need to formally define "Expected Returns"
- But we already see that this idea is appealing for continuous actions

"Policy Improvement with a Gradient Ascent??"

- We want to find the Policy that fetches the "Best Expected Returns"
- Gradient Ascent on "Expected Returns" w.r.t params of Policy func
- So we need a func approx for (stochastic) Policy Func: $\pi(s, a; \boldsymbol{\theta})$
- In addition to the usual func approx for Action Value Func: $Q(s, a; \boldsymbol{w})$
- $\pi(s, a; \boldsymbol{\theta})$ called *Actor* and $Q(s, a; \boldsymbol{w})$ called *Critic*
- Critic parameters $\boldsymbol{w}$ are optimized w.r.t $Q(s, a; \boldsymbol{w})$ loss function min
- Actor parameters $\boldsymbol{\theta}$ are optimized w.r.t Expected Returns max
- We need to formally define "Expected Returns"
- But we already see that this idea is appealing for continuous actions
- GPI with Policy Improvement done as **Policy Gradient (Ascent)**

Value Function-based and Policy-based RL

Value Function-based and Policy-based RL

- Value Function-based

Value Function-based and Policy-based RL

- Value Function-based
  - Learn Value Function (with a function approximation)

Value Function-based and Policy-based RL

- Value Function-based
  - Learn Value Function (with a function approximation)
  - Policy is implicit - readily derived from Value Function (eg: ϵ-greedy)

Value Function-based and Policy-based RL

- Value Function-based
  - Learn Value Function (with a function approximation)
  - Policy is implicit - readily derived from Value Function (eg: ϵ-greedy)
- Policy-based

Value Function-based and Policy-based RL

- Value Function-based
    - Learn Value Function (with a function approximation)
    - Policy is implicit - readily derived from Value Function (eg: ϵ-greedy)
- Policy-based
    - Learn Policy (with a function approximation)

Value Function-based and Policy-based RL

- Value Function-based
  - Learn Value Function (with a function approximation)
  - Policy is implicit - readily derived from Value Function (eg: ϵ-greedy)
- Policy-based
  - Learn Policy (with a function approximation)
  - No need to learn a Value Function

Value Function-based and Policy-based RL

- Value Function-based
  - Learn Value Function (with a function approximation)
  - Policy is implicit - readily derived from Value Function (eg: ϵ-greedy)
- Policy-based
  - Learn Policy (with a function approximation)
  - No need to learn a Value Function
- Actor-Critic

Value Function-based and Policy-based RL

- Value Function-based
  - Learn Value Function (with a function approximation)
  - Policy is implicit - readily derived from Value Function (eg: ϵ-greedy)
- Policy-based
  - Learn Policy (with a function approximation)
  - No need to learn a Value Function
- Actor-Critic
  - Learn Policy (Actor)

Value Function-based and Policy-based RL

- Value Function-based
  - Learn Value Function (with a function approximation)
  - Policy is implicit - readily derived from Value Function (eg: ϵ-greedy)
- Policy-based
  - Learn Policy (with a function approximation)
  - No need to learn a Value Function
- Actor-Critic
  - Learn Policy (Actor)
  - Learn Value Function (Critic)

Advantages and Disadvantages of Policy Gradient approach

Advantages and Disadvantages of Policy Gradient approach

Advantages:

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation
- Effective in high-dimensional or continuous action spaces

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation
- Effective in high-dimensional or continuous action spaces
- Small changes in $\boldsymbol{\theta} \Rightarrow$ small changes in π, and in state distribution

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation
- Effective in high-dimensional or continuous action spaces
- Small changes in $\boldsymbol{\theta} \Rightarrow$ small changes in π, and in state distribution
- This avoids the convergence issues seen in argmax-based algorithms

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation
- Effective in high-dimensional or continuous action spaces
- Small changes in $\boldsymbol{\theta} \Rightarrow$ small changes in π, and in state distribution
- This avoids the convergence issues seen in argmax-based algorithms

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation
- Effective in high-dimensional or continuous action spaces
- Small changes in $\boldsymbol{\theta} \Rightarrow$ small changes in π, and in state distribution
- This avoids the convergence issues seen in argmax-based algorithms

Disadvantages:

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation
- Effective in high-dimensional or continuous action spaces
- Small changes in $\boldsymbol{\theta} \Rightarrow$ small changes in π, and in state distribution
- This avoids the convergence issues seen in argmax-based algorithms

Disadvantages:

- Typically converge to a local optimum rather than a global optimum

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation
- Effective in high-dimensional or continuous action spaces
- Small changes in $\boldsymbol{\theta} \Rightarrow$ small changes in π, and in state distribution
- This avoids the convergence issues seen in argmax-based algorithms

Disadvantages:

- Typically converge to a local optimum rather than a global optimum
- Policy Evaluation is typically inefficient and has high variance

Advantages and Disadvantages of Policy Gradient approach

Advantages:

- Finds the best *Stochastic* Policy (Optimal Deterministic Policy, produced by other RL algorithms, can be unsuitable for POMDPs)
- Naturally *explores* due to Stochastic Policy representation
- Effective in high-dimensional or continuous action spaces
- Small changes in $\boldsymbol{\theta} \Rightarrow$ small changes in π, and in state distribution
- This avoids the convergence issues seen in argmax-based algorithms

Disadvantages:

- Typically converge to a local optimum rather than a global optimum
- Policy Evaluation is typically inefficient and has high variance
- Policy Improvement happens in small steps $\Rightarrow$ slow convergence

Notation

Notation

- Assume episodic with $0 \leq \gamma \leq 1$ or non-episodic with $0 \leq \gamma < 1$

Notation

- Assume episodic with $0 \leq \gamma \leq 1$ or non-episodic with $0 \leq \gamma < 1$
- Usual notation of discrete-time, countable-spaces, stationary MDPs

Notation

- Assume episodic with $0 \leq \gamma \leq 1$ or non-episodic with $0 \leq \gamma < 1$
- Usual notation of discrete-time, countable-spaces, stationary MDPs
- We lighten $\mathcal{P}(s, a, s')$ notation to $\mathcal{P}_{s,s'}^a$ and $\mathcal{R}(s, a)$ notation to $\mathcal{R}_s^a$

Notation

- Assume episodic with $0 \leq \gamma \leq 1$ or non-episodic with $0 \leq \gamma < 1$
- Usual notation of discrete-time, countable-spaces, stationary MDPs
- We lighten $\mathcal{P}(s, a, s')$ notation to $\mathcal{P}_{s,s'}^a$ and $\mathcal{R}(s, a)$ notation to $\mathcal{R}_s^a$
- Initial State Probability Distribution denoted as $p_0 : \mathcal{N} \rightarrow [0, 1]$

Notation

- Assume episodic with $0 \leq \gamma \leq 1$ or non-episodic with $0 \leq \gamma < 1$
- Usual notation of discrete-time, countable-spaces, stationary MDPs
- We lighten $\mathcal{P}(s, a, s')$ notation to $\mathcal{P}_{s,s'}^a$ and $\mathcal{R}(s, a)$ notation to $\mathcal{R}_s^a$
- Initial State Probability Distribution denoted as $p_0 : \mathcal{N} \rightarrow [0, 1]$
- Policy Function Approximation $\pi(s, a; \boldsymbol{\theta}) = \mathbb{P}[A_t = a | S_t = s; \boldsymbol{\theta}]$

Notation

- Assume episodic with $0 \leq \gamma \leq 1$ or non-episodic with $0 \leq \gamma < 1$
- Usual notation of discrete-time, countable-spaces, stationary MDPs
- We lighten $\mathcal{P}(s, a, s')$ notation to $\mathcal{P}_{s,s'}^a$ and $\mathcal{R}(s, a)$ notation to $\mathcal{R}_s^a$
- Initial State Probability Distribution denoted as $p_0 : \mathcal{N} \rightarrow [0, 1]$
- Policy Function Approximation $\pi(s, a; \boldsymbol{\theta}) = \mathbb{P}[A_t = a | S_t = s; \boldsymbol{\theta}]$

Notation

- Assume episodic with $0 \leq \gamma \leq 1$ or non-episodic with $0 \leq \gamma < 1$
- Usual notation of discrete-time, countable-spaces, stationary MDPs
- We lighten $\mathcal{P}(s, a, s')$ notation to $\mathcal{P}_{s,s'}^a$ and $\mathcal{R}(s, a)$ notation to $\mathcal{R}_s^a$
- Initial State Probability Distribution denoted as $p_0 : \mathcal{N} \rightarrow [0, 1]$
- Policy Function Approximation $\pi(s, a; \boldsymbol{\theta}) = \mathbb{P}[A_t = a | S_t = s; \boldsymbol{\theta}]$

PG coverage is quite similar for non-discounted non-episodic, by considering average-reward objective (we won't cover it)

“Expected Returns” Objective

"Expected Returns" Objective

Now we formalize the "Expected Returns" Objective $J(\boldsymbol{\theta})$

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1}]$$

“Expected Returns” Objective

Now we formalize the “Expected Returns” Objective $J(\boldsymbol{\theta})$

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1}]$$

Value Function $V^\pi(s)$ and Action Value function $Q^\pi(s,a)$ defined as:

$$V^\pi(s) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} \cdot R_{k+1} | S_t = s] \text{ for all } t = 0, 1, 2, \ldots$$

$$Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} \cdot R_{k+1} | S_t = s, A_t = a] \text{ for all } t = 0, 1, 2, \ldots$$

"Expected Returns" Objective

Now we formalize the "Expected Returns" Objective $J(\boldsymbol{\theta})$

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1}]$$

Value Function $V^\pi(s)$ and Action Value function $Q^\pi(s,a)$ defined as:

$$V^\pi(s) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} \cdot R_{k+1} | S_t = s] \text{ for all } t = 0, 1, 2, \ldots$$

$$Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} \cdot R_{k+1} | S_t = s, A_t = a] \text{ for all } t = 0, 1, 2, \ldots$$

Advantage Function $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

"Expected Returns" Objective

Now we formalize the "Expected Returns" Objective $J(\boldsymbol{\theta})$

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1}]$$

Value Function $V^\pi(s)$ and Action Value function $Q^\pi(s,a)$ defined as:

$$V^\pi(s) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} \cdot R_{k+1} | S_t = s] \text{ for all } t = 0, 1, 2, \ldots$$

$$Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} \cdot R_{k+1} | S_t = s, A_t = a] \text{ for all } t = 0, 1, 2, \ldots$$

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

Also, $p(s \to s', t, \pi)$ will be a key function for us - it denotes the probability of going from state s to s' in t steps by following policy π

Discounted-Aggregate State-Visitation Measure

Discounted-Aggregate State-Visitation Measure

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t \cdot R_{t+1}] = \sum_{t=0}^\infty \gamma^t \cdot \mathbb{E}_\pi[R_{t+1}]$$

Discounted-Aggregate State-Visitation Measure

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t \cdot R_{t+1}] = \sum_{t=0}^\infty \gamma^t \cdot \mathbb{E}_\pi[R_{t+1}]$$

$$= \sum_{t=0}^\infty \gamma^t \cdot \sum_{s \in \mathcal{N}} (\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi)) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

Discounted-Aggregate State-Visitation Measure

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1}] = \sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{E}_\pi[R_{t+1}]$$

$$= \sum_{t=0}^{\infty} \gamma^t \cdot \sum_{s \in \mathcal{N}} (\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot p(S_0 \to s, t, \pi)) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

$$= \sum_{s \in \mathcal{N}} (\sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \to s, t, \pi)) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

Discounted-Aggregate State-Visitation Measure

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1}] = \sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{E}_\pi[R_{t+1}]$$

$$= \sum_{t=0}^{\infty} \gamma^t \cdot \sum_{s \in \mathcal{N}} (\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi)) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

$$= \sum_{s \in \mathcal{N}} (\sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi)) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

Definition

$$J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

Discounted-Aggregate State-Visitation Measure

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1}] = \sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{E}_\pi[R_{t+1}]$$

$$= \sum_{t=0}^{\infty} \gamma^t \cdot \sum_{s \in \mathcal{N}} (\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi)) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

$$= \sum_{s \in \mathcal{N}} (\sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi)) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

Definition

$$J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathcal{R}_s^a$$

where $\rho^\pi(s) = \sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi)$ is the key function (for PG) we'll refer to as *Discounted-Aggregate State-Visitation Measure*.

Policy Gradient Theorem (PGT)

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^\pi(s, a)$$

- Note: $\rho^\pi(s)$ depends on $\boldsymbol{\theta}$, but there's no $\nabla_{\boldsymbol{\theta}} \rho^\pi(s)$ term in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

- Note: $\rho^{\pi}(s)$ depends on $\boldsymbol{\theta}$, but there's no $\nabla_{\boldsymbol{\theta}} \rho^{\pi}(s)$ term in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Note: $\nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) = \pi(s, a; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^\pi(s, a)$$

- Note: $\rho^\pi(s)$ depends on $\boldsymbol{\theta}$, but there's no $\nabla_{\boldsymbol{\theta}} \rho^\pi(s)$ term in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Note: $\nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) = \pi(s, a; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$
- So we can simply generate sampling traces, and at each time step, calculate $(\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})) \cdot Q^\pi(s, a)$ (probabilities implicit in paths)

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

- Note: $\rho^{\pi}(s)$ depends on $\boldsymbol{\theta}$, but there's no $\nabla_{\boldsymbol{\theta}} \rho^{\pi}(s)$ term in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Note: $\nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) = \pi(s, a; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$
- So we can simply generate sampling traces, and at each time step, calculate $(\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})) \cdot Q^{\pi}(s, a)$ (probabilities implicit in paths)
- Note: $\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$ is Score function (Gradient of log-likelihood)

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

- Note: $\rho^{\pi}(s)$ depends on $\boldsymbol{\theta}$, but there's no $\nabla_{\boldsymbol{\theta}} \rho^{\pi}(s)$ term in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Note: $\nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) = \pi(s, a; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$
- So we can simply generate sampling traces, and at each time step, calculate $(\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})) \cdot Q^{\pi}(s, a)$ (probabilities implicit in paths)
- Note: $\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$ is Score function (Gradient of log-likelihood)
- We will estimate $Q^{\pi}(s, a)$ with a function approximation $Q(s, a; \boldsymbol{w})$

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

- Note: $\rho^{\pi}(s)$ depends on $\boldsymbol{\theta}$, but there's no $\nabla_{\boldsymbol{\theta}} \rho^{\pi}(s)$ term in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Note: $\nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) = \pi(s, a; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$
- So we can simply generate sampling traces, and at each time step, calculate $(\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})) \cdot Q^{\pi}(s, a)$ (probabilities implicit in paths)
- Note: $\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$ is Score function (Gradient of log-likelihood)
- We will estimate $Q^{\pi}(s, a)$ with a function approximation $Q(s, a; \boldsymbol{w})$
- We will later show how to avoid the estimate bias of $Q(s, a; \boldsymbol{w})$

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

- Note: $\rho^{\pi}(s)$ depends on $\boldsymbol{\theta}$, but there's no $\nabla_{\boldsymbol{\theta}} \rho^{\pi}(s)$ term in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Note: $\nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) = \pi(s, a; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$
- So we can simply generate sampling traces, and at each time step, calculate $(\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})) \cdot Q^{\pi}(s, a)$ (probabilities implicit in paths)
- Note: $\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$ is Score function (Gradient of log-likelihood)
- We will estimate $Q^{\pi}(s, a)$ with a function approximation $Q(s, a; \boldsymbol{w})$
- We will later show how to avoid the estimate bias of $Q(s, a; \boldsymbol{w})$
- This numerical estimate of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ enables **Policy Gradient Ascent**

Policy Gradient Theorem (PGT)

Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

- Note: $\rho^{\pi}(s)$ depends on $\boldsymbol{\theta}$, but there's no $\nabla_{\boldsymbol{\theta}} \rho^{\pi}(s)$ term in $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Note: $\nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) = \pi(s, a; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$
- So we can simply generate sampling traces, and at each time step, calculate $(\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})) \cdot Q^{\pi}(s, a)$ (probabilities implicit in paths)
- Note: $\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$ is Score function (Gradient of log-likelihood)
- We will estimate $Q^{\pi}(s, a)$ with a function approximation $Q(s, a; \boldsymbol{w})$
- We will later show how to avoid the estimate bias of $Q(s, a; \boldsymbol{w})$
- This numerical estimate of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ enables **Policy Gradient Ascent**
- Let us look at the score function of some canonical $\pi(s, a; \boldsymbol{\theta})$

Canonical $\pi(s, a; \theta)$ for finite action spaces

Canonical $\pi(s, a; \boldsymbol{\theta})$ for finite action spaces

- For finite action spaces, we often use Softmax Policy

Canonical $\pi(s, a; \boldsymbol{\theta})$ for finite action spaces

- For finite action spaces, we often use Softmax Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for finite action spaces

- For finite action spaces, we often use Softmax Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- Features vector $\boldsymbol{\phi}(s, a) = (\phi_1(s, a), \ldots, \phi_m(s, a))$ for all $s \in \mathcal{N}, a \in \mathcal{A}$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for finite action spaces

- For finite action spaces, we often use Softmax Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- Features vector $\boldsymbol{\phi}(s, a) = (\phi_1(s, a), \ldots, \phi_m(s, a))$ for all $s \in \mathcal{N}, a \in \mathcal{A}$
- Weight actions using linear combinations of features: $\boldsymbol{\phi}(s, a)^T \cdot \boldsymbol{\theta}$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for finite action spaces

- For finite action spaces, we often use Softmax Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- Features vector $\boldsymbol{\phi}(s, a) = (\phi_1(s, a), \ldots, \phi_m(s, a))$ for all $s \in \mathcal{N}, a \in \mathcal{A}$
- Weight actions using linear combinations of features: $\boldsymbol{\phi}(s, a)^T \cdot \boldsymbol{\theta}$
- Action probabilities proportional to exponentiated weights:

$$\pi(s, a; \boldsymbol{\theta}) = \frac{e^{\boldsymbol{\phi}(s,a)^T \cdot \boldsymbol{\theta}}}{\sum_{b \in \mathcal{A}} e^{\boldsymbol{\phi}(s,b)^T \cdot \boldsymbol{\theta}}} \text{ for all } s \in \mathcal{N}, a \in \mathcal{A}$$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for finite action spaces

- For finite action spaces, we often use Softmax Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- Features vector $\phi(s, a) = (\phi_1(s, a), \ldots, \phi_m(s, a))$ for all $s \in \mathcal{N}, a \in \mathcal{A}$
- Weight actions using linear combinations of features: $\phi(s, a)^T \cdot \boldsymbol{\theta}$
- Action probabilities proportional to exponentiated weights:

$$\pi(s, a; \boldsymbol{\theta}) = \frac{e^{\phi(s,a)^T \cdot \boldsymbol{\theta}}}{\sum_{b \in \mathcal{A}} e^{\phi(s,b)^T \cdot \boldsymbol{\theta}}} \text{ for all } s \in \mathcal{N}, a \in \mathcal{A}$$

- The score function is:

$$\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) = \phi(s, a) - \sum_{b \in \mathcal{A}} \pi(s, b; \boldsymbol{\theta}) \cdot \phi(s, b) = \phi(s, a) - \mathbb{E}_\pi[\phi(s, \cdot)]$$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for continuous action spaces

Canonical $\pi(s, a; \boldsymbol{\theta})$ for continuous action spaces

- For continuous action spaces, we often use Gaussian Policy

Canonical $\pi(s, a; \boldsymbol{\theta})$ for continuous action spaces

- For continuous action spaces, we often use Gaussian Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for continuous action spaces

- For continuous action spaces, we often use Gaussian Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- State features vector $\boldsymbol{\phi}(s) = (\phi_1(s), \ldots, \phi_m(s))$ for all $s \in \mathcal{N}$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for continuous action spaces

- For continuous action spaces, we often use Gaussian Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- State features vector $\boldsymbol{\phi}(s) = (\phi_1(s), \ldots, \phi_m(s))$ for all $s \in \mathcal{N}$
- Gaussian Mean is a linear combination of state features $\boldsymbol{\phi}(s)^T \cdot \boldsymbol{\theta}$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for continuous action spaces

- For continuous action spaces, we often use Gaussian Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- State features vector $\boldsymbol{\phi}(s) = (\phi_1(s), \ldots, \phi_m(s))$ for all $s \in \mathcal{N}$
- Gaussian Mean is a linear combination of state features $\boldsymbol{\phi}(s)^T \cdot \boldsymbol{\theta}$
- Variance may be fixed σ^2, or can also be parameterized

Canonical $\pi(s, a; \boldsymbol{\theta})$ for continuous action spaces

- For continuous action spaces, we often use Gaussian Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- State features vector $\boldsymbol{\phi}(s) = (\phi_1(s), \ldots, \phi_m(s))$ for all $s \in \mathcal{N}$
- Gaussian Mean is a linear combination of state features $\boldsymbol{\phi}(s)^T \cdot \boldsymbol{\theta}$
- Variance may be fixed σ^2, or can also be parameterized
- Policy is Gaussian, $a \sim \mathcal{N}(\boldsymbol{\phi}(s)^T \cdot \boldsymbol{\theta}, \sigma^2)$ for all $s \in \mathcal{N}$

Canonical $\pi(s, a; \boldsymbol{\theta})$ for continuous action spaces

- For continuous action spaces, we often use Gaussian Policy
- $\boldsymbol{\theta}$ is an m-vector $(\theta_1, \ldots, \theta_m)$
- State features vector $\boldsymbol{\phi}(s) = (\phi_1(s), \ldots, \phi_m(s))$ for all $s \in \mathcal{N}$
- Gaussian Mean is a linear combination of state features $\boldsymbol{\phi}(s)^T \cdot \boldsymbol{\theta}$
- Variance may be fixed σ^2, or can also be parameterized
- Policy is Gaussian, $a \sim \mathcal{N}(\boldsymbol{\phi}(s)^T \cdot \boldsymbol{\theta}, \sigma^2)$ for all $s \in \mathcal{N}$
- The score function is:

$$\nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) = \frac{(a - \boldsymbol{\phi}(s)^T \cdot \boldsymbol{\theta}) \cdot \boldsymbol{\phi}(s)}{\sigma^2}$$

Proof of Policy Gradient Theorem

Proof of Policy Gradient Theorem

We begin the proof by noting that:

$$J(\boldsymbol{\theta}) = \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot V^{\pi}(S_0) = \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^{\pi}(S_0, A_0)$$

Proof of Policy Gradient Theorem

We begin the proof by noting that:

$$J(\boldsymbol{\theta}) = \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot V^\pi(S_0) = \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0)$$

Calculate $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ by parts $\pi(S_0, A_0; \boldsymbol{\theta})$ and $Q^\pi(S_0, A_0)$

Proof of Policy Gradient Theorem

We begin the proof by noting that:

$$J(\boldsymbol{\theta}) = \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot V^\pi(S_0) = \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0)$$

Calculate $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ by parts $\pi(S_0, A_0; \boldsymbol{\theta})$ and $Q^\pi(S_0, A_0)$

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = & \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) \\
& + \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} Q^\pi(S_0, A_0)
\end{aligned}$$

Proof of Policy Gradient Theorem

Proof of Policy Gradient Theorem

Now expand $Q^\pi(S_0, A_0)$ as:

$$\mathcal{R}_{S_0}^{A_0} + \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0,S_1}^{A_0} \cdot V^\pi(S_1) \text{ (Bellman Policy Equation)}$$

Proof of Policy Gradient Theorem

Now expand $Q^\pi(S_0, A_0)$ as:

$$\mathcal{R}_{S_0}^{A_0} + \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0,S_1}^{A_0} \cdot V^\pi(S_1) \text{ (Bellman Policy Equation)}$$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) +$$

$$\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} (\mathcal{R}_{S_0}^{A_0} + \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0,S_1}^{A_0} \cdot V^\pi(S_1))$$

Proof of Policy Gradient Theorem

Now expand $Q^\pi(S_0, A_0)$ as:

$$\mathcal{R}_{S_0}^{A_0} + \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0,S_1}^{A_0} \cdot V^\pi(S_1) \text{ (Bellman Policy Equation)}$$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) +$$

$$\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} (\mathcal{R}_{S_0}^{A_0} + \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0,S_1}^{A_0} \cdot V^\pi(S_1))$$

Note: $\nabla_\theta \mathcal{R}_{S_0}^{A_0} = 0$, so remove that term

Proof of Policy Gradient Theorem

Now expand $Q^\pi(S_0, A_0)$ as:

$$\mathcal{R}_{S_0}^{A_0} + \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0,S_1}^{A_0} \cdot V^\pi(S_1) \text{ (Bellman Policy Equation)}$$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) +$$

$$\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} (\mathcal{R}_{S_0}^{A_0} + \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0,S_1}^{A_0} \cdot V^\pi(S_1))$$

Note: $\nabla_\theta \mathcal{R}_{S_0}^{A_0} = 0$, so remove that term

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) +$$

$$\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} (\sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0,S_1}^{A_0} \cdot V^\pi(S_1))$$

Proof of Policy Gradient Theorem

Proof of Policy Gradient Theorem

Now bring the $\nabla_{\boldsymbol{\theta}}$ inside the $\sum_{S_1 \in \mathcal{N}}$ to apply only on $V^{\pi}(S_1)$

Proof of Policy Gradient Theorem

Now bring the $\nabla_{\boldsymbol{\theta}}$ inside the $\sum_{S_1 \in \mathcal{N}}$ to apply only on $V^\pi(S_1)$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) +$$
$$\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0, S_1}^{A_0} \cdot \nabla_{\boldsymbol{\theta}} V^\pi(S_1)$$

Proof of Policy Gradient Theorem

Now bring the $\nabla_{\boldsymbol{\theta}}$ inside the $\sum_{S_1 \in \mathcal{N}}$ to apply only on $V^\pi(S_1)$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) +$$
$$\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0, S_1}^{A_0} \cdot \nabla_{\boldsymbol{\theta}} V^\pi(S_1)$$

Now bring $\sum_{S_0 \in \mathcal{N}}$ and $\sum_{A_0 \in \mathcal{A}}$ inside the $\sum_{S_1 \in \mathcal{N}}$

Proof of Policy Gradient Theorem

Now bring the $\nabla_{\boldsymbol{\theta}}$ inside the $\sum_{S_1 \in \mathcal{N}}$ to apply only on $V^{\pi}(S_1)$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^{\pi}(S_0, A_0) +$$

$$\sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0, S_1}^{A_0} \cdot \nabla_{\boldsymbol{\theta}} V^{\pi}(S_1)$$

Now bring $\sum_{S_0 \in \mathcal{N}}$ and $\sum_{A_0 \in \mathcal{A}}$ inside the $\sum_{S_1 \in \mathcal{N}}$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^{\pi}(S_0, A_0) +$$

$$\sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot \left(\sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \mathcal{P}_{S_0, S_1}^{A_0} \right) \cdot \nabla_{\boldsymbol{\theta}} V^{\pi}(S_1)$$

Policy Gradient Theorem

Policy Gradient Theorem

Note that $\sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \mathcal{P}_{S_0,S_1}^{A_0} = p(S_0 \rightarrow S_1, 1, \pi)$

Policy Gradient Theorem

$$\text{Note that} \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \mathcal{P}_{S_0,S_1}^{A_0} = p(S_0 \rightarrow S_1, 1, \pi)$$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^{\pi}(S_0, A_0) +$$

$$\sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot p(S_0 \rightarrow S_1, 1, \pi) \cdot \nabla_{\boldsymbol{\theta}} V^{\pi}(S_1)$$

Policy Gradient Theorem

$$\text{Note that } \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \mathcal{P}_{S_0, S_1}^{A_0} = p(S_0 \rightarrow S_1, 1, \pi)$$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^{\pi}(S_0, A_0) +$$

$$\sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot p(S_0 \rightarrow S_1, 1, \pi) \cdot \nabla_{\boldsymbol{\theta}} V^{\pi}(S_1)$$

$$\text{Now expand } V^{\pi}(S_1) \text{ to } \sum_{A_1 \in \mathcal{A}} \pi(S_1, A_1; \boldsymbol{\theta}) \cdot Q^{\pi}(S_1, A_1)$$

Policy Gradient Theorem

$$\text{Note that } \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot \mathcal{P}_{S_0, S_1}^{A_0} = p(S_0 \rightarrow S_1, 1, \pi)$$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^{\pi}(S_0, A_0) +$$
$$\sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot p(S_0 \rightarrow S_1, 1, \pi) \cdot \nabla_{\boldsymbol{\theta}} V^{\pi}(S_1)$$

$$\text{Now expand } V^{\pi}(S_1) \text{ to } \sum_{A_1 \in \mathcal{A}} \pi(S_1, A_1; \boldsymbol{\theta}) \cdot Q^{\pi}(S_1, A_1)$$

$$= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^{\pi}(S_0, A_0) +$$
$$\sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot p(S_0 \rightarrow S_1, 1, \pi) \cdot \nabla_{\boldsymbol{\theta}} \Big(\sum_{A_1 \in \mathcal{A}} \pi(S_1, A_1; \boldsymbol{\theta}) \cdot Q^{\pi}(S_1, A_1) \Big)$$

Proof of Policy Gradient Theorem

Proof of Policy Gradient Theorem

We are now back to when we started calculating gradient of $\sum_a \pi \cdot Q^\pi$.

Proof of Policy Gradient Theorem

We are now back to when we started calculating gradient of $\sum_a \pi \cdot Q^\pi$. Follow the same process of splitting $\pi \cdot Q^\pi$, then Bellman-expanding Q^π (to calculate its gradient), and iterate.

Proof of Policy Gradient Theorem

We are now back to when we started calculating gradient of $\sum_a \pi \cdot Q^\pi$. Follow the same process of splitting $\pi \cdot Q^\pi$, then Bellman-expanding Q^π (to calculate its gradient), and iterate.

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) +$$

$$\sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot p(S_0 \rightarrow S_1, 1, \pi) \cdot (\sum_{A_1 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_1, A_1; \boldsymbol{\theta}) \cdot Q^\pi(S_1, A_1) + \ldots)$$

Proof of Policy Gradient Theorem

We are now back to when we started calculating gradient of $\sum_a \pi \cdot Q^\pi$.
Follow the same process of splitting $\pi \cdot Q^\pi$, then Bellman-expanding Q^π (to calculate its gradient), and iterate.

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_0, A_0; \boldsymbol{\theta}) \cdot Q^\pi(S_0, A_0) +$$

$$\sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot p(S_0 \rightarrow S_1, 1, \pi) \cdot (\sum_{A_1 \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_1, A_1; \boldsymbol{\theta}) \cdot Q^\pi(S_1, A_1) + \ldots)$$

This iterative process leads us to:

$$= \sum_{t=0}^{\infty} \sum_{S_t \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow S_t, t, \pi) \cdot \sum_{A_t \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_t, A_t; \boldsymbol{\theta}) \cdot Q^\pi(S_t, A_t)$$

Proof of Policy Gradient Theorem

Proof of Policy Gradient Theorem

$$\text{Bring } \sum_{t=0}^{\infty} \text{ inside } \sum_{S_t \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \text{ and note that}$$

$$\sum_{A_t \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_t, A_t; \boldsymbol{\theta}) \cdot Q^{\pi}(S_t, A_t) \text{ is independent of } t$$

Proof of Policy Gradient Theorem

Bring $\sum_{t=0}^{\infty}$ inside $\sum_{S_t \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}}$ and note that

$\sum_{A_t \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_t, A_t; \boldsymbol{\theta}) \cdot Q^{\pi}(S_t, A_t)$ is independent of t

$$= \sum_{s \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

Proof of Policy Gradient Theorem

$$\text{Bring } \sum_{t=0}^{\infty} \text{ inside } \sum_{S_t \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \text{ and note that}$$

$$\sum_{A_t \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_t, A_t; \boldsymbol{\theta}) \cdot Q^{\pi}(S_t, A_t) \text{ is independent of } t$$

$$= \sum_{s \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

$$\text{Reminder that } \sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi) \stackrel{\text{def}}{=} \rho^{\pi}(s). \text{ So,}$$

Proof of Policy Gradient Theorem

Bring $\sum_{t=0}^{\infty}$ inside $\sum_{S_t \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}}$ and note that

$\sum_{A_t \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(S_t, A_t; \boldsymbol{\theta}) \cdot Q^{\pi}(S_t, A_t)$ is independent of t

$$= \sum_{s \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

Reminder that $\sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi) \stackrel{\text{def}}{=} \rho^{\pi}(s)$. So,

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a)$$

$\mathbb{Q}.\mathbb{E}.\mathbb{D}.$

Monte-Carlo Policy Gradient (REINFORCE Algorithm)

Monte-Carlo Policy Gradient (REINFORCE Algorithm)

- Update $\boldsymbol{\theta}$ by stochastic gradient ascent using PGT

Monte-Carlo Policy Gradient (REINFORCE Algorithm)

- Update θ by stochastic gradient ascent using PGT
- Using $G_t = \sum_{k=t}^{T} \gamma^{k-t} \cdot R_{k+1}$ as an unbiased sample of $Q^\pi(S_t, A_t)$

$$\Delta\theta = \alpha \cdot \gamma^t \cdot \nabla_\theta \log \pi(S_t, A_t; \theta) \cdot G_t$$

Monte-Carlo Policy Gradient (REINFORCE Algorithm)

- Update $\boldsymbol{\theta}$ by stochastic gradient ascent using PGT
- Using $G_t = \sum_{k=t}^{T} \gamma^{k-t} \cdot R_{k+1}$ as an unbiased sample of $Q^\pi(S_t, A_t)$

$$\Delta\boldsymbol{\theta} = \alpha \cdot \gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot G_t$$

Monte-Carlo Policy Gradient (REINFORCE Algorithm)

- Update $\boldsymbol{\theta}$ by stochastic gradient ascent using PGT
- Using $G_t = \sum_{k=t}^{T} \gamma^{k-t} \cdot R_{k+1}$ as an unbiased sample of $Q^\pi(S_t, A_t)$

$$\Delta\boldsymbol{\theta} = \alpha \cdot \gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot G_t$$

Algorithm 4.5: $\mathrm{REINFORCE}(\cdot)$

Initialize $\boldsymbol{\theta}$ arbitrarily
for each episode $\{S_0, A_0, R_1, S_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T\} \sim \pi(\cdot, \cdot; \boldsymbol{\theta})$
do $\begin{cases} \textbf{for } t \leftarrow 0 \textbf{ to } T \\ \textbf{do } \begin{cases} G \leftarrow \sum_{k=t}^{T} \gamma^{k-t} \cdot R_{k+1} \\ \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot G \end{cases} \end{cases}$

Reducing Variance using a Critic

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^\pi(s, a)$

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^\pi(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^{\pi}(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
  - Critic updates parameters $\boldsymbol{w}$ to approximate Q-function for policy π

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^\pi(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
  - Critic updates parameters $\boldsymbol{w}$ to approximate Q-function for policy π
  - Critic could use any of the algorithms we learnt earlier:

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^{\pi}(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
  - Critic updates parameters $\boldsymbol{w}$ to approximate Q-function for policy π
  - Critic could use any of the algorithms we learnt earlier:
    - Monte Carlo policy evaluation

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^\pi(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
    - Critic updates parameters $\boldsymbol{w}$ to approximate Q-function for policy π
    - Critic could use any of the algorithms we learnt earlier:
        - Monte Carlo policy evaluation
        - Temporal-Difference Learning

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^{\pi}(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
  - Critic updates parameters $\boldsymbol{w}$ to approximate Q-function for policy π
  - Critic could use any of the algorithms we learnt earlier:
    - Monte Carlo policy evaluation
    - Temporal-Difference Learning
    - $TD(\lambda)$ based on Eligibility Traces

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^\pi(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
    - Critic updates parameters $\boldsymbol{w}$ to approximate Q-function for policy π
    - Critic could use any of the algorithms we learnt earlier:
        - Monte Carlo policy evaluation
        - Temporal-Difference Learning
        - $TD(\lambda)$ based on Eligibility Traces
        - Could even use LSTD (if critic function approximation is linear)

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^\pi(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
    - Critic updates parameters $\boldsymbol{w}$ to approximate Q-function for policy π
    - Critic could use any of the algorithms we learnt earlier:
        - Monte Carlo policy evaluation
        - Temporal-Difference Learning
        - $TD(\lambda)$ based on Eligibility Traces
        - Could even use LSTD (if critic function approximation is linear)
    - Actor updates policy parameters $\boldsymbol{\theta}$ in direction suggested by Critic

Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic $Q(s, a; \boldsymbol{w})$ to estimate $Q^{\pi}(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
  - Critic updates parameters $\boldsymbol{w}$ to approximate Q-function for policy π
  - Critic could use any of the algorithms we learnt earlier:
    - Monte Carlo policy evaluation
    - Temporal-Difference Learning
    - $TD(\lambda)$ based on Eligibility Traces
    - Could even use LSTD (if critic function approximation is linear)
  - Actor updates policy parameters $\boldsymbol{\theta}$ in direction suggested by Critic
  - This is Approximate Policy Gradient due to *Bias* of Critic

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q(s, a; \boldsymbol{w})$$

So what does the algorithm look like?

So what does the algorithm look like?

- Generate a sufficient set of sampling traces $S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$

So what does the algorithm look like?

- Generate a sufficient set of sampling traces $S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$
- S_0 is sampled from the distribution $p_0(\cdot)$

So what does the algorithm look like?

- Generate a sufficient set of sampling traces $S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$
- S_0 is sampled from the distribution $p_0(\cdot)$
- A_t is sampled from $\pi(S_t, \cdot; \boldsymbol{\theta})$

So what does the algorithm look like?

- Generate a sufficient set of sampling traces $S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$
- S_0 is sampled from the distribution $p_0(\cdot)$
- A_t is sampled from $\pi(S_t, \cdot; \boldsymbol{\theta})$
- Receive atomic experience (R_{t+1}, S_{t+1}) from the environment

So what does the algorithm look like?

- Generate a sufficient set of sampling traces $S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$
- S_0 is sampled from the distribution $p_0(\cdot)$
- A_t is sampled from $\pi(S_t, \cdot; \boldsymbol{\theta})$
- Receive atomic experience (R_{t+1}, S_{t+1}) from the environment
- At each time step t, update $\boldsymbol{w}$ proportional to gradient of appropriate (MC or TD-based) loss function of $Q(s, a; \boldsymbol{w})$

So what does the algorithm look like?

- Generate a sufficient set of sampling traces $S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$
- S_0 is sampled from the distribution $p_0(\cdot)$
- A_t is sampled from $\pi(S_t, \cdot; \boldsymbol{\theta})$
- Receive atomic experience (R_{t+1}, S_{t+1}) from the environment
- At each time step t, update $\boldsymbol{w}$ proportional to gradient of appropriate (MC or TD-based) loss function of $Q(s, a; \boldsymbol{w})$
- Sum $\gamma^t \cdot (\nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta})) \cdot Q(S_t, A_t; \boldsymbol{w})$ over t and over paths

So what does the algorithm look like?

- Generate a sufficient set of sampling traces $S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$
- S_0 is sampled from the distribution $p_0(\cdot)$
- A_t is sampled from $\pi(S_t, \cdot; \boldsymbol{\theta})$
- Receive atomic experience (R_{t+1}, S_{t+1}) from the environment
- At each time step t, update $\boldsymbol{w}$ proportional to gradient of appropriate (MC or TD-based) loss function of $Q(s, a; \boldsymbol{w})$
- Sum $\gamma^t \cdot (\nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta})) \cdot Q(S_t, A_t; \boldsymbol{w})$ over t and over paths
- Update $\boldsymbol{\theta}$ using this (biased) estimate of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

So what does the algorithm look like?

- Generate a sufficient set of sampling traces $S_0, A_0, R_1, S_1, A_1, R_2, S_2 \ldots$
- S_0 is sampled from the distribution $p_0(\cdot)$
- A_t is sampled from $\pi(S_t, \cdot; \boldsymbol{\theta})$
- Receive atomic experience (R_{t+1}, S_{t+1}) from the environment
- At each time step t, update $\boldsymbol{w}$ proportional to gradient of appropriate (MC or TD-based) loss function of $Q(s, a; \boldsymbol{w})$
- Sum $\gamma^t \cdot (\nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta})) \cdot Q(S_t, A_t; \boldsymbol{w})$ over t and over paths
- Update $\boldsymbol{\theta}$ using this (biased) estimate of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- Iterate with a new set of sampling traces ...

Reducing Variance with a Baseline

Reducing Variance with a Baseline

- We can reduce variance by subtracting a baseline function $B(s)$ from $Q(s, a; \boldsymbol{w})$ in the Policy Gradient estimate

Reducing Variance with a Baseline

- We can reduce variance by subtracting a baseline function $B(s)$ from $Q(s, a; \boldsymbol{w})$ in the Policy Gradient estimate
- This means at each time step, we replace
  $\gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot Q(S_t, A_t; \boldsymbol{w})$ with
  $\gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot (Q(S_t, A_t; \boldsymbol{w}) - B(S_t))$

Reducing Variance with a Baseline

- We can reduce variance by subtracting a baseline function $B(s)$ from $Q(s, a; \boldsymbol{w})$ in the Policy Gradient estimate
- This means at each time step, we replace $\gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot Q(S_t, A_t; \boldsymbol{w})$ with $\gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot (Q(S_t, A_t; \boldsymbol{w}) - B(S_t))$
- Note that Baseline function $B(s)$ is only a function of s (and not a)

Reducing Variance with a Baseline

- We can reduce variance by subtracting a baseline function $B(s)$ from $Q(s, a; \boldsymbol{w})$ in the Policy Gradient estimate
- This means at each time step, we replace $\gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot Q(S_t, A_t; \boldsymbol{w})$ with $\gamma^t \cdot \nabla_{\boldsymbol{\theta}} \log \pi(S_t, A_t; \boldsymbol{\theta}) \cdot (Q(S_t, A_t; \boldsymbol{w}) - B(S_t))$
- Note that Baseline function $B(s)$ is only a function of s (and not a)
- This ensures that subtracting Baseline $B(s)$ does not add bias

$$
\begin{aligned}
& \sum_{s \in \mathcal{N}} \rho^\pi(s) \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot B(s) \\
= & \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot B(s) \cdot \nabla_{\boldsymbol{\theta}} (\sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta})) \\
= & \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot B(s) \cdot \nabla_{\boldsymbol{\theta}} 1 \\
= & 0
\end{aligned}
$$

Using State Value function as Baseline

Using State Value function as Baseline

- A good baseline $B(s)$ is state value function $V(s; \boldsymbol{v})$

Using State Value function as Baseline

- A good baseline $B(s)$ is state value function $V(s; \boldsymbol{v})$
- Rewrite Policy Gradient algorithm using advantage function estimate

$$A(s, a; \boldsymbol{w}, \boldsymbol{v}) = Q(s, a; \boldsymbol{w}) - V(s; \boldsymbol{v})$$

Using State Value function as Baseline

- A good baseline $B(s)$ is state value function $V(s;\boldsymbol{v})$
- Rewrite Policy Gradient algorithm using advantage function estimate

$$A(s,a;\boldsymbol{w},\boldsymbol{v}) = Q(s,a;\boldsymbol{w}) - V(s;\boldsymbol{v})$$

- Now the estimate of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is given by:

$$\sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s,a;\boldsymbol{\theta}) \cdot A(s,a;\boldsymbol{w},\boldsymbol{v})$$

Using State Value function as Baseline

- A good baseline $B(s)$ is state value function $V(s; \boldsymbol{v})$
- Rewrite Policy Gradient algorithm using advantage function estimate

$$A(s, a; \boldsymbol{w}, \boldsymbol{v}) = Q(s, a; \boldsymbol{w}) - V(s; \boldsymbol{v})$$

- Now the estimate of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is given by:

$$\sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot A(s, a; \boldsymbol{w}, \boldsymbol{v})$$

- At each time step, we update both sets of parameters $\boldsymbol{w}$ and $\boldsymbol{v}$

TD Error as estimate of Advantage Function

TD Error as estimate of Advantage Function

- Consider TD error δ^π for the *true* Value Function $V^\pi(s)$

$$\delta^\pi = r + \gamma \cdot V^\pi(s') - V^\pi(s)$$

TD Error as estimate of Advantage Function

- Consider TD error δ^π for the *true* Value Function $V^\pi(s)$

$$\delta^\pi = r + \gamma \cdot V^\pi(s') - V^\pi(s)$$

- δ^π is an unbiased estimate of Advantage function $A^\pi(s,a)$

$$\mathbb{E}_\pi[\delta^\pi|s,a] = \mathbb{E}_\pi[r+\gamma\cdot V^\pi(s')|s,a] - V^\pi(s) = Q^\pi(s,a) - V^\pi(s) = A^\pi(s,a)$$

TD Error as estimate of Advantage Function

- Consider TD error δ^π for the *true* Value Function $V^\pi(s)$

$$\delta^\pi = r + \gamma \cdot V^\pi(s') - V^\pi(s)$$

- δ^π is an unbiased estimate of Advantage function $A^\pi(s,a)$

$$\mathbb{E}_\pi[\delta^\pi|s,a] = \mathbb{E}_\pi[r + \gamma \cdot V^\pi(s')|s,a] - V^\pi(s) = Q^\pi(s,a) - V^\pi(s) = A^\pi(s,a)$$

- So we can write Policy Gradient in terms of $\mathbb{E}_\pi[\delta^\pi|s,a]$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathbb{E}_\pi[\delta^\pi|s,a]$$

TD Error as estimate of Advantage Function

- Consider TD error δ^π for the *true* Value Function $V^\pi(s)$

$$\delta^\pi = r + \gamma \cdot V^\pi(s') - V^\pi(s)$$

- δ^π is an unbiased estimate of Advantage function $A^\pi(s,a)$

$$\mathbb{E}_\pi[\delta^\pi|s,a] = \mathbb{E}_\pi[r + \gamma \cdot V^\pi(s')|s,a] - V^\pi(s) = Q^\pi(s,a) - V^\pi(s) = A^\pi(s,a)$$

- So we can write Policy Gradient in terms of $\mathbb{E}_\pi[\delta^\pi|s,a]$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s,a;\boldsymbol{\theta}) \cdot \mathbb{E}_\pi[\delta^\pi|s,a]$$

- In practice, we can use func approx for TD error (and sample):

$$\delta(s,r,s';\boldsymbol{v}) = r + \gamma \cdot V(s';\boldsymbol{v}) - V(s;\boldsymbol{v})$$

TD Error as estimate of Advantage Function

- Consider TD error δ^π for the *true* Value Function $V^\pi(s)$

$$\delta^\pi = r + \gamma \cdot V^\pi(s') - V^\pi(s)$$

- δ^π is an unbiased estimate of Advantage function $A^\pi(s, a)$

$$\mathbb{E}_\pi[\delta^\pi | s, a] = \mathbb{E}_\pi[r + \gamma \cdot V^\pi(s') | s, a] - V^\pi(s) = Q^\pi(s, a) - V^\pi(s) = A^\pi(s, a)$$

- So we can write Policy Gradient in terms of $\mathbb{E}_\pi[\delta^\pi | s, a]$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot \mathbb{E}_\pi[\delta^\pi | s, a]$$

- In practice, we can use func approx for TD error (and sample):

$$\delta(s, r, s'; \boldsymbol{v}) = r + \gamma \cdot V(s'; \boldsymbol{v}) - V(s; \boldsymbol{v})$$

- This approach requires only one set of critic parameters $\boldsymbol{v}$

TD Error can be used by both Actor and Critic

TD Error can be used by both Actor and Critic

Algorithm 4.7: ACTOR-CRITIC-TD-ERROR($\cdot$)

Initialize Policy params $\boldsymbol{\theta}$ and State VF params $\boldsymbol{v}$ arbitrarily

for each episode

do $\begin{cases} \text{Initialize } s \text{ (first state of episode)} \\ P \leftarrow 1 \\ \textbf{while } s \text{ is not terminal} \\ \quad \textbf{do} \begin{cases} a \sim \pi(s, \cdot; \boldsymbol{\theta}) \\ \text{Take action } a, \text{ receive } r, s' \text{ from the environment} \\ \delta \leftarrow r + \gamma \cdot V(s'; \boldsymbol{v}) - V(s; \boldsymbol{v}) \\ \boldsymbol{v} \leftarrow \boldsymbol{v} + \alpha_{\boldsymbol{v}} \cdot \delta \cdot \nabla_{\boldsymbol{v}} V(s; \boldsymbol{v}) \\ \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \cdot P \cdot \delta \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) \\ P \leftarrow \gamma \cdot P \\ s \leftarrow s' \end{cases} \end{cases}$

Using Eligibility Traces for both Actor and Critic

Using Eligibility Traces for both Actor and Critic

Algorithm 4.9: ACTOR-CRITIC-ELIGIBILITY-TRACES($\cdot$)

Initialize Policy params $\boldsymbol{\theta}$ and State VF params $\boldsymbol{v}$ arbitrarily

for each episode

do
- Initialize s (first state of episode)
- $\boldsymbol{z_\theta}, \boldsymbol{z_v} \leftarrow 0$ (eligibility traces for $\boldsymbol{\theta}$ and $\boldsymbol{v}$)
- $P \leftarrow 1$
- **while** s is not terminal
- **do**
  - $a \sim \pi(s, \cdot; \theta)$
  - Take action a, observe r, s'
  - $\delta \leftarrow r + \gamma \cdot V(s'; \boldsymbol{v}) - V(s; \boldsymbol{v})$
  - $\boldsymbol{z_v} \leftarrow \gamma \cdot \lambda_{\boldsymbol{v}} \cdot \boldsymbol{z_v} + \nabla_{\boldsymbol{v}} V(s; \boldsymbol{v})$
  - $\boldsymbol{z_\theta} \leftarrow \gamma \cdot \lambda_{\boldsymbol{\theta}} \cdot \boldsymbol{z_\theta} + P \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$
  - $\boldsymbol{v} \leftarrow \boldsymbol{v} + \alpha_{\boldsymbol{v}} \cdot \delta \cdot \boldsymbol{z_v}$
  - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \cdot \delta \cdot \boldsymbol{z_\theta}$
  - $P \leftarrow \gamma \cdot P, s \leftarrow s'$

Overcoming Bias

Overcoming Bias

- We've learnt a few ways of how to reduce variance

Overcoming Bias

- We've learnt a few ways of how to reduce variance
- But we haven't discussed how to overcome bias

Overcoming Bias

- We've learnt a few ways of how to reduce variance
- But we haven't discussed how to overcome bias
- All of the following substitutes for $Q^\pi(s, a)$ in PG have bias:

Overcoming Bias

- We've learnt a few ways of how to reduce variance
- But we haven't discussed how to overcome bias
- All of the following substitutes for $Q^\pi(s,a)$ in PG have bias:
    - $Q(s,a;\boldsymbol{w})$

Overcoming Bias

- We've learnt a few ways of how to reduce variance
- But we haven't discussed how to overcome bias
- All of the following substitutes for $Q^\pi(s, a)$ in PG have bias:
  - $Q(s, a; \boldsymbol{w})$
  - $A(s, a; \boldsymbol{w}, \boldsymbol{v})$

Overcoming Bias

- We've learnt a few ways of how to reduce variance
- But we haven't discussed how to overcome bias
- All of the following substitutes for $Q^\pi(s, a)$ in PG have bias:
    - $Q(s, a; \boldsymbol{w})$
    - $A(s, a; \boldsymbol{w}, \boldsymbol{v})$
    - $\delta(s, s', r; \boldsymbol{v})$

Overcoming Bias

- We've learnt a few ways of how to reduce variance
- But we haven't discussed how to overcome bias
- All of the following substitutes for $Q^\pi(s, a)$ in PG have bias:
    - $Q(s, a; \boldsymbol{w})$
    - $A(s, a; \boldsymbol{w}, \boldsymbol{v})$
    - $\delta(s, s', r; \boldsymbol{v})$
- Turns out there is indeed a way to overcome bias

Overcoming Bias

- We've learnt a few ways of how to reduce variance
- But we haven't discussed how to overcome bias
- All of the following substitutes for $Q^\pi(s, a)$ in PG have bias:
  - $Q(s, a; \boldsymbol{w})$
  - $A(s, a; \boldsymbol{w}, \boldsymbol{v})$
  - $\delta(s, s', r; \boldsymbol{v})$
- Turns out there is indeed a way to overcome bias
- It is called the *Compatible Function Approximation Theorem*

Compatible Function Approximation Theorem

Compatible Function Approximation Theorem

Theorem

If the following two conditions are satisfied:

Compatible Function Approximation Theorem

Theorem

If the following two conditions are satisfied:

1. *Critic gradient is* compatible *with the Actor score function*

$$\nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$$

Compatible Function Approximation Theorem

Theorem

If the following two conditions are satisfied:

1. *Critic gradient is* compatible *with the Actor score function*

$$\nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$$

2. *Critic parameters* $\boldsymbol{w}$ *minimize the following mean-squared error:*

$$\epsilon = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^{\pi}(s, a) - Q(s, a; \boldsymbol{w}))^2$$

Compatible Function Approximation Theorem

Theorem

If the following two conditions are satisfied:

1. *Critic gradient is* compatible *with the Actor score function*

$$\nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$$

2. *Critic parameters* $\boldsymbol{w}$ *minimize the following mean-squared error:*

$$\epsilon = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^{\pi}(s, a) - Q(s, a; \boldsymbol{w}))^2$$

Then the Policy Gradient using critic $Q(s, a; \boldsymbol{w})$ *is exact:*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q(s, a; \boldsymbol{w})$$

Proof of Compatible Function Approximation Theorem

Proof of Compatible Function Approximation Theorem

For $\boldsymbol{w}$ that minimizes

$$\epsilon = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w}))^2,$$

Proof of Compatible Function Approximation Theorem

For $\boldsymbol{w}$ that minimizes

$$\epsilon = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w}))^2,$$

$$\sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = 0$$

Proof of Compatible Function Approximation Theorem

For $\boldsymbol{w}$ that minimizes

$$\epsilon = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w}))^2,$$

$$\sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = 0$$

But since $\nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$, we have:

Proof of Compatible Function Approximation Theorem

For $\boldsymbol{w}$ that minimizes

$$\epsilon = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w}))^2,$$

$$\sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = 0$$

But since $\nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$, we have:

$$\sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) = 0$$

Proof of Compatible Function Approximation Theorem

For $\boldsymbol{w}$ that minimizes

$$\epsilon = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w}))^2,$$

$$\sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = 0$$

But since $\nabla_{\boldsymbol{w}} Q(s, a; \boldsymbol{w}) = \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$, we have:

$$\sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (Q^\pi(s, a) - Q(s, a; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) = 0$$

$$\begin{aligned} \text{Therefore, } & \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^\pi(s, a) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) \\ &= \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot Q(s, a; \boldsymbol{w}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) \end{aligned}$$

Proof of Compatible Function Approximation Theorem

Proof of Compatible Function Approximation Theorem

$$\text{But } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$$

Proof of Compatible Function Approximation Theorem

$$\text{But } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^{\pi}(s, a) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$$

$$\begin{aligned}\text{So, } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot Q(s, a; \boldsymbol{w}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) \\ &= \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q(s, a; \boldsymbol{w})\end{aligned}$$

$\mathbb{Q}.\mathbb{E}.\mathbb{D}.$

Proof of Compatible Function Approximation Theorem

$$\text{But } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot Q^\pi(s, a) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta})$$

$$\begin{aligned}\text{So, } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot Q(s, a; \boldsymbol{w}) \cdot \nabla_{\boldsymbol{\theta}} \log \pi(s, a; \boldsymbol{\theta}) \\ &= \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(s, a; \boldsymbol{\theta}) \cdot Q(s, a; \boldsymbol{w})\end{aligned}$$

$\mathbb{Q}.\mathbb{E}.\mathbb{D}.$

This means with conditions (1) and (2) of Compatible Function Approximation Theorem, we can use the critic func approx $Q(s, a; \boldsymbol{w})$ and still have the exact Policy Gradient.

How to enable Compatible Function Approximation

How to enable Compatible Function Approximation

A simple way to enable Compatible Function Approximation

How to enable Compatible Function Approximation

A simple way to enable Compatible Function Approximation $\frac{\partial Q(s,a;\boldsymbol{w})}{\partial w_i} = \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}, \forall i$ is to set $Q(s,a;\boldsymbol{w})$ to be linear in its features.

How to enable Compatible Function Approximation

A simple way to enable Compatible Function Approximation $\frac{\partial Q(s,a;\boldsymbol{w})}{\partial w_i} = \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}, \forall i$ is to set $Q(s,a;\boldsymbol{w})$ to be linear in its features.

$$Q(s,a;\boldsymbol{w}) = \sum_{i=1}^{m} \phi_i(s,a) \cdot w_i = \sum_{i=1}^{m} \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i$$

How to enable Compatible Function Approximation

A simple way to enable Compatible Function Approximation $\frac{\partial Q(s,a;\boldsymbol{w})}{\partial w_i} = \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}, \forall i$ is to set $Q(s,a;\boldsymbol{w})$ to be linear in its features.

$$Q(s,a;\boldsymbol{w}) = \sum_{i=1}^{m} \phi_i(s,a) \cdot w_i = \sum_{i=1}^{m} \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i$$

We note below that a compatible $Q(s,a;\boldsymbol{w})$ serves as an approximation of the advantage function.

How to enable Compatible Function Approximation

A simple way to enable Compatible Function Approximation $\frac{\partial Q(s,a;\boldsymbol{w})}{\partial w_i} = \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}, \forall i$ is to set $Q(s,a;\boldsymbol{w})$ to be linear in its features.

$$Q(s,a;\boldsymbol{w}) = \sum_{i=1}^{m} \phi_i(s,a) \cdot w_i = \sum_{i=1}^{m} \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i$$

We note below that a compatible $Q(s,a;\boldsymbol{w})$ serves as an approximation of the advantage function.

$$\sum_{a \in \mathcal{A}} \pi(s,a;\boldsymbol{\theta}) \cdot Q(s,a;\boldsymbol{w}) = \sum_{a \in \mathcal{A}} \pi(s,a;\boldsymbol{\theta}) \cdot (\sum_{i=1}^{m} \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i)$$

How to enable Compatible Function Approximation

A simple way to enable Compatible Function Approximation $\frac{\partial Q(s,a;\boldsymbol{w})}{\partial w_i} = \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}, \forall i$ is to set $Q(s,a;\boldsymbol{w})$ to be linear in its features.

$$Q(s,a;\boldsymbol{w}) = \sum_{i=1}^{m} \phi_i(s,a) \cdot w_i = \sum_{i=1}^{m} \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i$$

We note below that a compatible $Q(s,a;\boldsymbol{w})$ serves as an approximation of the advantage function.

$$\sum_{a \in \mathcal{A}} \pi(s,a;\boldsymbol{\theta}) \cdot Q(s,a;\boldsymbol{w}) = \sum_{a \in \mathcal{A}} \pi(s,a;\boldsymbol{\theta}) \cdot (\sum_{i=1}^{m} \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i)$$

$$= \sum_{a \in \mathcal{A}} (\sum_{i=1}^{m} \frac{\partial \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i) = \sum_{i=1}^{m} (\sum_{a \in \mathcal{A}} \frac{\partial \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}) \cdot w_i$$

How to enable Compatible Function Approximation

A simple way to enable Compatible Function Approximation $\frac{\partial Q(s,a;\boldsymbol{w})}{\partial w_i} = \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}, \forall i$ is to set $Q(s,a;\boldsymbol{w})$ to be linear in its features.

$$Q(s,a;\boldsymbol{w}) = \sum_{i=1}^{m} \phi_i(s,a) \cdot w_i = \sum_{i=1}^{m} \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i$$

We note below that a compatible $Q(s,a;\boldsymbol{w})$ serves as an approximation of the advantage function.

$$\sum_{a \in \mathcal{A}} \pi(s,a;\boldsymbol{\theta}) \cdot Q(s,a;\boldsymbol{w}) = \sum_{a \in \mathcal{A}} \pi(s,a;\boldsymbol{\theta}) \cdot (\sum_{i=1}^{m} \frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i)$$

$$= \sum_{a \in \mathcal{A}} (\sum_{i=1}^{m} \frac{\partial \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i} \cdot w_i) = \sum_{i=1}^{m} (\sum_{a \in \mathcal{A}} \frac{\partial \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}) \cdot w_i$$

$$= \sum_{i=1}^{m} \frac{\partial}{\partial \theta_i} (\sum_{a \in \mathcal{A}} \pi(s,a;\boldsymbol{\theta})) \cdot w_i = \sum_{i=1}^{m} \frac{\partial 1}{\partial \theta_i} \cdot w_i = 0$$

Fisher Information Matrix

Fisher Information Matrix

Denoting $[\frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}], i = 1, \ldots, m$ as the score column vector $\boldsymbol{SC}(s, a; \boldsymbol{\theta})$ and assuming compatible linear-approximation critic:

Fisher Information Matrix

Denoting $[\frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}], i = 1, \ldots, m$ as the score column vector $\boldsymbol{SC}(s,a;\boldsymbol{\theta})$ and assuming compatible linear-approximation critic:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{s \in \mathcal{N}} \rho^\pi(s) \cdot \sum_{a \in \mathcal{A}} \pi(s,a;\boldsymbol{\theta}) \cdot (\boldsymbol{SC}(s,a;\boldsymbol{\theta}) \cdot \boldsymbol{SC}(s,a;\boldsymbol{\theta})^T \cdot \boldsymbol{w}) \\ &= \mathbb{E}_{s \sim \rho^\pi, a \sim \pi}[\boldsymbol{SC}(s,a;\boldsymbol{\theta}) \cdot \boldsymbol{SC}(s,a;\boldsymbol{\theta})^T] \cdot \boldsymbol{w} \\ &= FIM_{\rho^\pi,\pi}(\boldsymbol{\theta}) \cdot \boldsymbol{w}\end{aligned}$$

Fisher Information Matrix

Denoting $[\frac{\partial \log \pi(s,a;\boldsymbol{\theta})}{\partial \theta_i}], i = 1, \ldots, m$ as the score column vector $\boldsymbol{SC}(s, a; \boldsymbol{\theta})$ and assuming compatible linear-approximation critic:

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{s \in \mathcal{N}} \rho^{\pi}(s) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \boldsymbol{\theta}) \cdot (\boldsymbol{SC}(s, a; \boldsymbol{\theta}) \cdot \boldsymbol{SC}(s, a; \boldsymbol{\theta})^T \cdot \boldsymbol{w}) \\
&= \mathbb{E}_{s \sim \rho^{\pi}, a \sim \pi}[\boldsymbol{SC}(s, a; \boldsymbol{\theta}) \cdot \boldsymbol{SC}(s, a; \boldsymbol{\theta})^T] \cdot \boldsymbol{w} \\
&= FIM_{\rho^{\pi}, \pi}(\boldsymbol{\theta}) \cdot \boldsymbol{w}
\end{aligned}
$$

where $FIM_{\rho_{\pi}, \pi}(\boldsymbol{\theta})$ is the Fisher Information Matrix w.r.t. $s \sim \rho^{\pi}, a \sim \pi$.

Natural Policy Gradient

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)
- Natural gradient yields better convergence (we won't cover proof)

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)
- Natural gradient yields better convergence (we won't cover proof)

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)
- Natural gradient yields better convergence (we won't cover proof)

Formally defined as: $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = FIM_{\rho_\pi,\pi}(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)
- Natural gradient yields better convergence (we won't cover proof)

$$\text{Formally defined as: } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = FIM_{\rho_\pi,\pi}(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$$

$$\text{Therefore, } \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta}) = \boldsymbol{w}$$

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)
- Natural gradient yields better convergence (we won't cover proof)

$$\text{Formally defined as: } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = FIM_{\rho_\pi,\pi}(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$$

$$\text{Therefore, } \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta}) = \boldsymbol{w}$$

This compact result is great for our algorithm:

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)
- Natural gradient yields better convergence (we won't cover proof)

$$\text{Formally defined as: } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = FIM_{\rho_\pi,\pi}(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$$

$$\text{Therefore, } \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta}) = \boldsymbol{w}$$

This compact result is great for our algorithm:

- Update Critic params $\boldsymbol{w}$ with the critic loss gradient (at step t) as:

$$\gamma^t \cdot (R_{t+1} + \gamma \cdot \boldsymbol{SC}(S_{t+1}, A_{t+1}; \boldsymbol{\theta})^T \cdot \boldsymbol{w} - \boldsymbol{SC}(S_t, A_t; \boldsymbol{\theta})^T \cdot \boldsymbol{w}) \cdot \boldsymbol{SC}(S_t, A_t; \boldsymbol{\theta})$$

Natural Policy Gradient

- Recall the idea of Natural Gradient from Numerical Optimization
- Natural gradient $\nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$ is the direction of optimal $\boldsymbol{\theta}$ movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)
- Natural gradient yields better convergence (we won't cover proof)

$$\text{Formally defined as: } \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = FIM_{\rho_\pi,\pi}(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta})$$

$$\text{Therefore, } \nabla_{\boldsymbol{\theta}}^{nat} J(\boldsymbol{\theta}) = \boldsymbol{w}$$

This compact result is great for our algorithm:

- Update Critic params $\boldsymbol{w}$ with the critic loss gradient (at step t) as:

$$\gamma^t \cdot (R_{t+1} + \gamma \cdot \boldsymbol{SC}(S_{t+1}, A_{t+1}; \boldsymbol{\theta})^T \cdot \boldsymbol{w} - \boldsymbol{SC}(S_t, A_t; \boldsymbol{\theta})^T \cdot \boldsymbol{w}) \cdot \boldsymbol{SC}(S_t, A_t; \boldsymbol{\theta})$$

- Update Actor params $\boldsymbol{\theta}$ in the direction equal to value of $\boldsymbol{w}$