

# A Guided Tour of Chapter 12: Policy Gradient Algorithms

Ashwin Rao

ICME, Stanford University

# 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  $\operatorname{argmax}_a Q(s, a)$
- How do we do  $\operatorname{argmax}$  when action space is large or continuous?
- Idea: Do Policy Improvement step with a Gradient Ascent instead

# “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:  $\pi(s, a; \theta)$
- In addition to the usual func approx for Action Value Func:  $Q(s, a; \mathbf{w})$
- $\pi(s, a; \theta)$  called *Actor* and  $Q(s, a; \mathbf{w})$  called *Critic*
- Critic parameters  $\mathbf{w}$  are optimized w.r.t  $Q(s, a; \mathbf{w})$  loss function min
- Actor parameters  $\theta$  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
  - Learn Value Function (with a function approximation)
  - Policy is implicit - readily derived from Value Function (eg:  $\epsilon$ -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:

- 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  $\theta \Rightarrow$  small changes in  $\pi$ , 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

- Assume episodic with  $0 \leq \gamma \leq 1$  or non-episodic with  $0 \leq \gamma < 1$
- Assume discrete-time, countable-spaces, time-homogeneous 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  $\pi(s, a; \theta) = \mathbb{P}[A_t = a | S_t = s; \theta]$

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

# “Expected Returns” Objective

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

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

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

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

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

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

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

# Discounted-Aggregate State-Visitation Measure

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1} \right] = \sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{E}_{\pi} [R_{t+1}] \\ &= \sum_{t=0}^{\infty} \gamma^t \cdot \sum_{s \in \mathcal{N}} \left( \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi) \right) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \theta) \cdot \mathcal{R}_s^a \\ &= \sum_{s \in \mathcal{N}} \left( \sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi) \right) \cdot \sum_{a \in \mathcal{A}} \pi(s, a; \theta) \cdot \mathcal{R}_s^a \end{aligned}$$

## Definition

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

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)

## Theorem

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

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

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

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

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

- The score function is:

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

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

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

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

# Proof of Policy Gradient Theorem

We begin the proof by noting that:

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

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

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

# Proof of Policy Gradient Theorem

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

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

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

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

$$\begin{aligned} &= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\theta} \pi(S_0, A_0; \theta) \cdot Q^\pi(S_0, A_0) + \\ &\quad \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \theta) \cdot \nabla_{\theta} \left( \sum_{S_1 \in \mathcal{N}} \gamma \cdot \mathcal{P}_{S_0, S_1}^{A_0} \cdot V^\pi(S_1) \right) \end{aligned}$$

# Proof of Policy Gradient Theorem

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

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

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

$$\begin{aligned} &= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\theta} \pi(S_0, A_0; \theta) \cdot Q^{\pi}(S_0, A_0) + \\ &\quad \sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot \left( \sum_{A_0 \in \mathcal{A}} \pi(S_0, A_0; \theta) \cdot \mathcal{P}_{S_0, S_1}^{A_0} \right) \cdot \nabla_{\theta} V^{\pi}(S_1) \end{aligned}$$

# Policy Gradient Theorem

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

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

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

$$\begin{aligned} &= \sum_{S_0 \in \mathcal{N}} p_0(S_0) \cdot \sum_{A_0 \in \mathcal{A}} \nabla_{\theta} \pi(S_0, A_0; \theta) \cdot Q^{\pi}(S_0, A_0) + \\ &\quad \sum_{S_1 \in \mathcal{N}} \sum_{S_0 \in \mathcal{N}} \gamma \cdot p_0(S_0) \cdot p(S_0 \rightarrow S_1, 1, \pi) \cdot \nabla_{\theta} \left( \sum_{A_1 \in \mathcal{A}} \pi(S_1, A_1; \theta) \cdot Q^{\pi}(S_1, A_1) \right) \end{aligned}$$

# 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  $Q^\pi$  (to calculate its gradient), and iterate.

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

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

This iterative process leads us to:

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



# 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

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

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

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,

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

Q.E.D.

# Monte-Carlo Policy Gradient (REINFORCE Algorithm)

- Update  $\theta$  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  $Q^\pi(S_t, A_t)$

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

## Algorithm 0.1: REINFORCE( $\cdot$ )

Initialize  $\theta$  arbitrarily

**for** each episode  $\{S_0, A_0, R_1, S_1, \dots, S_{T-1}, A_{T-1}, R_T, S_T\} \sim \pi(\cdot, \cdot; \theta)$

**do**  $\left\{ \begin{array}{l} \text{for } t \leftarrow 0 \text{ to } T \\ \text{do } \left\{ \begin{array}{l} G \leftarrow \sum_{k=t}^T \gamma^{k-t} \cdot R_{k+1} \\ \theta \leftarrow \theta + \alpha \cdot \gamma^t \cdot \nabla_{\theta} \log \pi(S_t, A_t; \theta) \cdot G \end{array} \right. \end{array} \right.$

# Reducing Variance using a Critic

- Monte Carlo Policy Gradient has high variance
- We use a Critic  $Q(s, a; \mathbf{w})$  to estimate  $Q^\pi(s, a)$
- Actor-Critic algorithms maintain two sets of parameters:
  - Critic updates parameters  $\mathbf{w}$  to approximate  $Q$ -function for policy  $\pi$
  - 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  $\theta$  in direction suggested by Critic
  - This is Approximate Policy Gradient due to *Bias* of Critic

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

# So what does the algorithm look like?

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

# Reducing Variance with a Baseline

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

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

# Using State Value function as Baseline

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

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

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

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

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

# TD Error as estimate of Advantage Function

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

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

- $\delta^\pi$  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  $\mathbb{E}_\pi[\delta^\pi | s, a]$

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

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

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

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

# TD Error can be used by both Actor and Critic

## Algorithm 0.2: ACTOR-CRITIC-TD-ERROR( $\cdot$ )

Initialize Policy params  $\theta$  and State VF params  $\mathbf{v}$  arbitrarily  
**for** each episode

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



# Using Eligibility Traces for both Actor and Critic

## Algorithm 0.3: ACTOR-CRITIC-ELIGIBILITY-TRACES( $\cdot$ )

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

**for** each episode

**do** {

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

# 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:
  - $Q(s, a; \mathbf{w})$
  - $A(s, a; \mathbf{w}, \mathbf{v})$
  - $\delta(s, s', r; \mathbf{v})$
- Turns out there is indeed a way to overcome bias
- It is called the *Compatible Function Approximation Theorem*

# Compatible Function Approximation Theorem

## Theorem

Let  $\mathbf{w}_\theta^*$  denote the Critic parameters  $\mathbf{w}$  that minimize the following mean-squared-error for given policy parameters  $\theta$ :

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

Assume that the data type of  $\theta$  is the same as the data type of  $\mathbf{w}$  and furthermore, assume that for any policy parameters  $\theta$ , the Critic gradient at  $\mathbf{w}_\theta^*$  is compatible with the Actor score function, i.e.,

$$\nabla_{\mathbf{w}} Q(s, a; \mathbf{w}_\theta^*) = \nabla_{\theta} \log \pi(s, a; \theta) \text{ for all } s \in \mathcal{N}, \text{ for all } a \in \mathcal{A}$$

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

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

# Proof of Compatible Function Approximation Theorem

For a given  $\theta$ , since  $\mathbf{w}_\theta^*$  minimizes the mean-squared-error as defined above, we have:

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

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

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

Therefore,

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

# Proof of Compatible Function Approximation Theorem

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

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

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

# How to enable Compatible Function Approximation

A simple way to enable Compatible Function Approximation

$\frac{\partial Q(s, a; \mathbf{w}_{\theta}^*)}{\partial w_i} = \frac{\partial \log \pi(s, a; \theta)}{\partial \theta_i}$ ,  $\forall i$  is to set  $Q(s, a; \mathbf{w})$  to be linear in its features.

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

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

$$\begin{aligned} \sum_{a \in \mathcal{A}} \pi(s, a; \theta) \cdot Q(s, a; \mathbf{w}) &= \sum_{a \in \mathcal{A}} \pi(s, a; \theta) \cdot \left( \sum_{i=1}^m \frac{\partial \log \pi(s, a; \theta)}{\partial \theta_i} \cdot w_i \right) \\ &= \sum_{a \in \mathcal{A}} \left( \sum_{i=1}^m \frac{\partial \pi(s, a; \theta)}{\partial \theta_i} \cdot w_i \right) = \sum_{i=1}^m \left( \sum_{a \in \mathcal{A}} \frac{\partial \pi(s, a; \theta)}{\partial \theta_i} \right) \cdot w_i \\ &= \sum_{i=1}^m \frac{\partial}{\partial \theta_i} \left( \sum_{a \in \mathcal{A}} \pi(s, a; \theta) \right) \cdot w_i = \sum_{i=1}^m \frac{\partial 1}{\partial \theta_i} \cdot w_i = 0 \end{aligned}$$

# Fisher Information Matrix

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

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

where  $FIM_{\rho^{\pi}, \pi}(\theta)$  is the Fisher Information Matrix w.r.t.  $s \sim \rho^{\pi}, a \sim \pi$ . Hence, updates after each atomic experience are as follows:

$$\Delta \theta = \alpha_{\theta} \cdot \gamma^t \cdot \mathbf{SC}(S_t, A_t; \mathbf{w}) \cdot \mathbf{SC}(S_t, A_t; \mathbf{w})^T \cdot \mathbf{w}$$

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

# Natural Policy Gradient (NPG)

- Natural gradient  $\nabla_{\theta}^{nat} J(\theta)$  is the direction of optimal  $\theta$  movement
- In terms of the KL-divergence metric (versus plain Euclidean norm)
- Formally defined as:

$$\nabla_{\theta} J(\theta) = FIM_{\rho_{\pi}, \pi}(\theta) \cdot \nabla_{\theta}^{nat} J(\theta)$$

- Enabling Compatible Function Approximation implies:

$$\nabla_{\theta}^{nat} J(\theta) = \mathbf{w}_{\theta}^*$$

- **This compact result is great for our algorithm:**
  - Update Critic params  $\mathbf{w}$  with the critic loss gradient (at step  $t$ ) as:
$$(R_{t+1} + \gamma \cdot \mathbf{SC}(S_{t+1}, A_{t+1}; \theta))^T \cdot \mathbf{w} - \mathbf{SC}(S_t, A_t; \theta)^T \cdot \mathbf{w} \cdot \mathbf{SC}(S_t, A_t; \theta)$$
  - Update Actor params  $\theta$  in the direction of  $\mathbf{w}$



# Deterministic Policy Gradient (DPG)

- Function approximation for deterministic policy for continuous actions
- DPG expressed as Expected Gradient of Q-Value Function
- Integrates only over state space, so efficient for high-dim action spaces
- Usual machinery of PG is applicable to DPG
- Is Off-Policy: Target policy  $\pi_D$  learnt, behavior policy  $\mu$  is exploratory
- Core Idea: Instead of greedy policy improvement for continuous action spaces, move policy in the direction of gradient of Q-Value Function
- Policy parameters  $\theta$  are updated in proportion to  $\nabla_{\theta} Q(s, \pi_D(s; \theta))$
- Average direction of policy improvements is given by:

$$\mathbb{E}_{s \sim \rho^{\pi_D}} [\nabla_{\theta} Q(s, \pi_D(s; \theta))] = \mathbb{E}_{s \sim \rho^{\pi_D}} [\nabla_{\theta} \pi_D(s; \theta) \cdot \nabla_a Q^{\pi_D}(s, a) \Big|_{a=\pi_D(s; \theta)}]$$

$$\rho^{\pi_D}(s) = \sum_{S_0 \in \mathcal{N}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(S_0) \cdot p(S_0 \rightarrow s, t, \pi_D)$$

- For multi-dimensional  $a$ ,  $\nabla_{\theta} \pi(s; \theta)$  is a Jacobian matrix

$$J(\theta) = \mathbb{E}_{\pi_D} \left[ \sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1} \right] = \sum_{s \in \mathcal{N}} \rho^{\pi_D}(s) \cdot \mathcal{R}_s^{\pi_D(s; \theta)} = \mathbb{E}_{s \sim \rho^{\pi_D}} [\mathcal{R}_s^{\pi_D(s; \theta)}]$$

## Theorem

For action space  $\mathbb{R}^k$ ,

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \sum_{s \in \mathcal{N}} \rho^{\pi_D}(s) \cdot \nabla_{\theta} \pi_D(s; \theta) \cdot \nabla_a Q^{\pi_D}(s, a) \Big|_{a=\pi_D(s; \theta)} \\ &= \mathbb{E}_{s \sim \rho^{\pi_D}} [\nabla_{\theta} \pi_D(s; \theta) \cdot \nabla_a Q^{\pi_D}(s, a) \Big|_{a=\pi_D(s; \theta)}] \end{aligned}$$

In practice, we use critic  $Q(s, a; \mathbf{w})$  and keep updating both  $\mathbf{w}$  and  $\theta$ :

$$\Delta \mathbf{w} = \alpha_{\mathbf{w}} \cdot (R_{t+1} + \gamma \cdot Q(S_{t+1}, \pi_D(S_{t+1}; \theta); \mathbf{w}) - Q(S_t, A_t; \mathbf{w})) \cdot \nabla_{\mathbf{w}} Q(S_t, A_t; \mathbf{w})$$

$$\Delta \theta = \alpha_{\theta} \cdot \gamma^t \cdot \nabla_{\theta} \pi_D(S_t; \theta) \cdot \nabla_a Q(S_t, A_t; \mathbf{w}) \Big|_{a=\pi_D(S_t; \theta)}$$

# NES applied to solving Markov Decision Processes (MDPs)

- We set  $F(\cdot)$  to be the (stochastic) *Return* of a MDP
- $\psi$  refers to the parameters of a policy  $\pi_\psi : \mathcal{S} \rightarrow \mathcal{A}$
- $\psi$  will be drawn from an isotropic multivariate Gaussian distribution
- Gaussian with mean vector  $\theta$  and fixed diagonal covariance matrix  $\sigma^2 I$
- The average objective (*Expected Return*) can then be written as:

$$\mathbb{E}_{\psi \sim p_\theta}[F(\psi)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)}[F(\theta + \sigma \cdot \epsilon)]$$

- The gradient ( $\nabla_\theta$ ) of *Expected Return* can be written as:

$$\begin{aligned} & \mathbb{E}_{\psi \sim p_\theta}[\nabla_\theta(\log p_\theta(\psi)) \cdot F(\psi)] \\ &= \mathbb{E}_{\psi \sim \mathcal{N}(\theta, \sigma^2 I)}[\nabla_\theta\left(\frac{-(\psi - \theta)^T \cdot (\psi - \theta)}{2\sigma^2}\right) \cdot F(\psi)] \\ &= \frac{1}{\sigma} \cdot \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)}[\epsilon \cdot F(\theta + \sigma \cdot \epsilon)] \end{aligned}$$

# A sampling-based algorithm to solve the MDP

- The above formula helps estimate gradient of *Expected Return*
- By sampling several  $\epsilon$  (each  $\epsilon$  represents a *Policy*  $\pi_{\theta+\sigma \cdot \epsilon}$ )
- And averaging  $\epsilon \cdot F(\theta + \sigma \cdot \epsilon)$  across a large set ( $n$ ) of  $\epsilon$  samples
- Note  $F(\theta + \sigma \cdot \epsilon)$  involves playing an episode for a given sampled  $\epsilon$ , and obtaining that episode's *Return*  $F(\theta + \sigma \cdot \epsilon)$
- Hence,  $n$  values of  $\epsilon$ ,  $n$  *Policies*  $\pi_{\theta+\sigma \cdot \epsilon}$ , and  $n$  *Returns*  $F(\theta + \sigma \cdot \epsilon)$
- Given gradient estimate, we update  $\theta$  in this gradient direction
- Which in turn leads to new samples of  $\epsilon$  (new set of *Policies*  $\pi_{\theta+\sigma \cdot \epsilon}$ )
- And the process repeats until  $\mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)}[F(\theta + \sigma \cdot \epsilon)]$  is maximized
- The key inputs to the algorithm will be:
  - Learning rate (SGD Step Size)  $\alpha$
  - Standard Deviation  $\sigma$
  - Initial value of parameter vector  $\theta_0$

# The Algorithm

**Algorithm 0.4:** NATURAL EVOLUTION STRATEGIES( $\alpha, \sigma, \theta_0$ )

```
for  $t \leftarrow 0, 1, 2, \dots$ 
do  $\begin{cases} \text{Sample } \epsilon_1, \epsilon_2, \dots, \epsilon_n \sim \mathcal{N}(0, I) \\ \text{Compute Returns } F_i \leftarrow F(\theta_t + \sigma \cdot \epsilon_i) \text{ for } i = 1, 2, \dots, n \\ \theta_{t+1} \leftarrow \theta_t + \frac{\alpha}{n\sigma} \sum_{i=1}^n \epsilon_i \cdot F_i \end{cases}$ 
```

# Resemblance to Policy Gradient?

- On the surface, this NES algorithm looks like [Policy Gradient](#) (PG)
- Because it's not Value Function-based (it's Policy-based, like PG)
- Also, similar to PG, it uses a gradient to move towards optimality
- But, ES does not interact with the environment (like PG/RL does)
- ES operates at a high-level, ignoring (state,action,reward) interplay
- Specifically, does not aim to assign credit to actions in specific states
- Hence, ES doesn't have the core essence of RL: *Estimating the Q-Value Function of a Policy and using it to Improve the Policy*
- Therefore, we don't classify ES as Reinforcement Learning
- We consider ES to be an alternative approach to RL Algorithms

- Traditional view has been that ES won't work on high-dim problems
- Specifically, ES has been shown to be data-inefficient relative to RL
- Because ES resembles simple hill-climbing based only on finite differences along a few random directions at each step
- However, ES is very simple to implement (no Value Function approx. or back-propagation needed), and is highly parallelizable
- ES has the benefits of being indifferent to distribution of rewards and to action frequency, and is tolerant of long horizons
- [This paper from OpenAI Researchers](#) shows techniques to make NES more robust and more data-efficient, and they demonstrate that NES has more exploratory behavior than advanced PG algorithms
- I'd always recommend trying NES before attempting to solve with RL

# Key Takeaways from this Chapter

- PG Algorithms are based on GPI with Policy Improvement as a Stochastic Gradient Ascent for "Expected Returns" Objective  $J(\theta)$  where  $\theta$  are parameters of the function approximation for the Policy
- Policy Gradient Theorem gives us a simple formula for  $\nabla_{\theta} J(\theta)$  in terms of the score of the policy function approximation
- We can reduce variance in PG algorithms by using a critic and by using an estimate of the advantage function for the Q-Value Function
- Compatible Function Approximation Theorem enables us to overcome bias in PG Algorithms
- Natural Policy Gradient and Deterministic Policy Gradient are specialized PG algorithms that have worked well in practice
- Evolutionary Strategies are technically not RL, but they resemble PG Algorithms and can sometimes be quite effective for MDP Control