ST455: Reinforcement Learning

Lecture 8: Policy Gradient Methods

Chengchun Shi

Recap

- We have covered **dynamic programming**, **Monte Carlo** methods (Lecture 3) and **temporal-difference learning** (Lectures 4 – 7). All these methods are **value-based** methods that derive the optimal policy by computing a value function.
- Today's lecture will adopt a difference approach that directly **searches** the optimal policy within a restricted policy class.

Lecture Outline

1. **Policy Gradient Method Introduction**
2. **Policy Gradient Theorem**
3. **REINFORCE and Actor Critic Algorithms**
4. **Advantage Actor-Critic (A2C)**

Lecture Outline

Policy Function Approximation

Value Function Approximation	No	Yes
No	Value-based (tabular)	REINFORCE
Yes	Value-based	Actor-Critic

- **Value-based**
  - Tabular (Lectures 3 & 4)
  - Function approx (Lectures 5 & 7)
- **REINFORCE**
  - No value function
  - Learn policy
- **Actor-critic**
  - Learn value
  - Learn policy
- **Advantage actor-critic**
  - Variance reduction

Lecture Outline

1. **Policy Gradient Method Introduction**
2. Policy Gradient Theorem
3. REINFORCE and Actor Critic Algorithms
4. Advantage Actor-Critic (A2C)

Policy We Studied So Far

- **Greedy policy**:

$$\pi^{\text{opt}}(s) = \arg\max_{a} Q^{\pi^{\text{opt}}}(s, a)$$

- ϵ-**Greedy policy**:

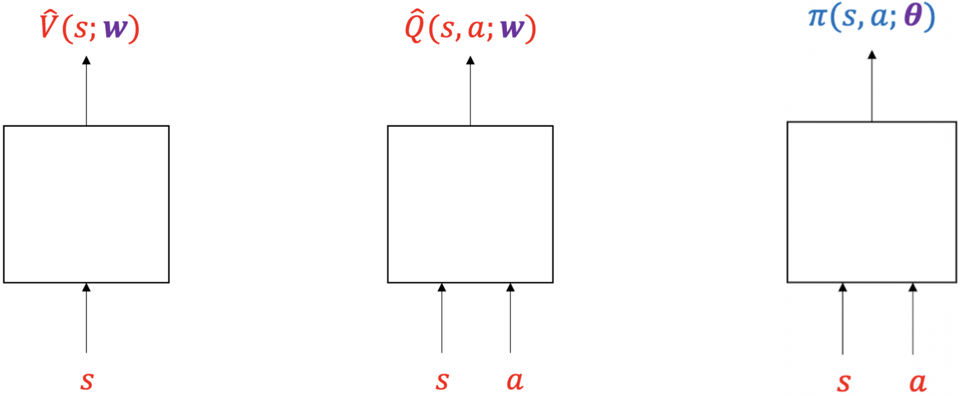
$$\begin{cases} \pi^{\text{opt}}(s), & \text{with probability } 1 - \epsilon \\ \text{random action}, & \text{with probability } \epsilon. \end{cases}$$

- **Value-based methods**: Policy Iteration, Value Iteration, SARSA, Q-Learning, etc.

Value-based v.s. Policy Gradient Methods

- **Value-based methods**: derive π^{opt} by learning an optimal Q-function (with or without function approximation)
- **Policy gradient methods**: search π^{opt} within a restricted function class (e.g., linear, neural networks) that maximizes the value

Value-based v.s. Policy Gradient Methods (Cont'd)



Value-based Methods **Policy Gradient Methods**

Example: Linear Function Approximation

- Linear approximation of features $\phi(s, a)$
- State-action value function approximation

$$Q(s, a; \theta) = \phi^\top(s, a)\theta$$

- Policy function approximation

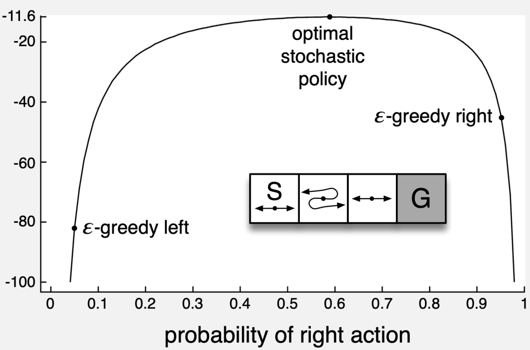
$$\pi(s, a; \theta) = \frac{\exp(\phi^\top(s, a)\theta)}{\sum_{a'} \exp(\phi^\top(s, a')\theta)}$$

$\phi^\top(s, a)\theta$ similar to the preference score in the gradient based methods in HW1

Value-based v.s. Policy Gradient Methods (Cont'd)

- **Pros** of policy gradient methods:
  1. Suitable for learning general **stochastic** policies (value-based methods mainly designed for deterministic policies)
  2. More **robust** to model misspecification
  3. Scalable for **high-dimensional** or **continuous** action spaces (SARSA, Q-learning mainly designed for discrete action space)
- **Cons** of policy gradient methods:
  1. Convergence to local minima
  2. May have large variance

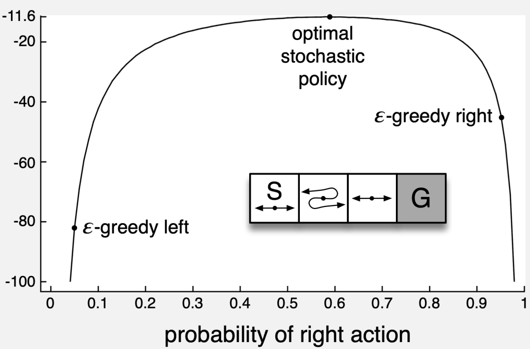
Example I: Advantage of Stochastic Policy



- **Reward** is $-\mathbf{1}$ per step
- **Action**: left or right
- $\boldsymbol{G}$: **terminal** state
- Second state: actions are **reversed**
- Linear combination of features
- $\phi(\boldsymbol{s}, \mathrm{left}) = [1, 0]$ for any $\boldsymbol{s}$
- $\phi(\boldsymbol{s}, \mathrm{right}) = [0, 1]$ for any $\boldsymbol{s}$

- $\pi^{\mathbf{opt}}$: right $\longrightarrow$ left $\longrightarrow$ right
- Under the function approximation, implements the same policy at each state

Example I: Advantage of Stochastic Policy (Cont'd)



- **Reward** is $-\mathbf{1}$ per step
- **Action**: left or right
- $\boldsymbol{G}$: **terminal** state
- Second state: actions are **reversed**
- Linear combination of features
- $\phi(s, \text{left}) = [1, 0]$ for any s
- $\phi(s, \text{right}) = [0, 1]$ for any s

- Value-based method: $\arg\max_a \phi^\top(s, a)\boldsymbol{\theta} = \text{left}\mathbb{I}(\theta_1 > \theta_2) + \text{right}\mathbb{I}(\theta_1 \leq \theta_2)$
  - Deterministic: either always move to the left, or always move to the right
- Policy gradient method: $\pi(\text{right}|s) = \exp(\theta_2)/[\exp(\theta_1) + \exp(\theta_2)]$
  - Stochastic: move to the right at each step with certain probability

Example II: Robustness of Policy Gradient Method

- Q-function is more **difficult** to model compared to the optimal policy
- Example: optimal Q-function: $Q^{\pi^{\text{opt}}}(s, a) = g(\phi^\top(s, a)\theta^*)$ for some monotonically increasing function $g : \mathbb{R} \to \mathbb{R}$
- When g is not **identity** function, value-based method misspecifies Q-function model

$$g(\phi^\top(s, a)\theta^*) \neq \phi^\top(s, a)\theta$$

- However, since g is a monotonically increasing function

$$\pi^{\text{opt}}(s) = \arg\max_a g(\phi^\top(s, a)\theta^*) = \arg\max_a \phi^\top(s, a)\theta^*$$

- Policy gradient methods correctly identifies the optimal policy

$$\frac{\exp(\phi^\top(s, a)\theta)}{\sum_{a'} \exp(\phi^\top(s, a')\theta)} \to \mathbb{I}(a = \pi^{\text{opt}}(s))$$

when $\theta = k\theta^*$ and $k \to \infty$

Policy Objective Functions

- Average rewards:

$$J(\theta) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}^{\pi(\bullet;\theta)}\left[\sum_{t=0}^{T-1} R_t\right] = \sum_{s,a} \nu^{\pi(\bullet;\theta)}(s)\pi(s,a;\theta)\mathcal{R}_s^a$$

  where $\mathcal{R}_s^a = \mathbb{E}(R_t | A_t = a, S_t = s)$
- For each π, the states $\{S_t\}_t$ forms a time-homogeneous Markov chain
- $\nu^{\pi(\bullet;\theta)}$ the stationary distribution of $\{S_t\}_t$ under $\pi(\bullet;\theta)$

Policy Objective Functions (Cont'd)

- Discounted rewards: given a discounted factor $\gamma \in [0, 1]$ and initial state distribution ν, maximize the expected discounted rewards:

$$J(\theta) = \mathbb{E}^{\pi(\bullet;\theta)}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right],$$

  or equivalently,

$$J(\theta) = \sum_{s} \nu(s) V^{\pi(\bullet;\theta)}(s)$$

- If $\gamma = 1$, the task is assumed to be episodic

Lecture Outline

1. Policy Gradient Method Introduction
2. **Policy Gradient Theorem**
3. REINFORCE and Actor Critic Algorithms
4. Advantage Actor-Critic (A2C)

Policy Gradient

- **Objective**: identify the maximizer of $\boldsymbol{J}(\boldsymbol{\theta})$
- **Method**: apply (stochastic) gradient ascent algorithm to update $\boldsymbol{\theta}$ (gradient descent to minimize $-\boldsymbol{J}(\boldsymbol{\theta})$)

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \boldsymbol{\alpha_t} \nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta_t})$$

Need to calculate the gradient $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})$!

Policy Gradient Theorem

Theorem

For any differentiable policy $\pi(s, a; \theta)$ with respect to parameter θ, the policy gradient for average reward and discounted expected rewards objective is

$$\nabla_\theta J(\theta) = \sum_{s,a} \mu^{\pi(\bullet;\theta)}(s, a) \nabla_\theta \log(\pi(s, a; \theta)) Q^{\pi(\bullet;\theta)}(s, a)$$

- For average reward objective:
  $\mu^{\pi(\bullet;\theta)}$ is the stationary distribution of $\{(S_t, A_t)\}_t$ under $\pi(\bullet; \theta)$
- For discounted expected rewards objective:

$$\mu^{\pi(\bullet;\theta)}(s, a) = \sum_{t \geq 0} \gamma^t \mathrm{Pr}^{\pi(\bullet;\theta)}(S_t = s, A_t = a)$$

  Discounted state-action visitation probability

Policy Gradient Theorem (Cont'd)

Theorem

For any differentiable policy $\pi(s, a; \theta)$ with respect to parameter θ, the policy gradient for average reward and discounted expected rewards objective is

$$\nabla_\theta J(\theta) = \sum_{s,a} \mu^{\pi(\bullet;\theta)}(s, a) \nabla_\theta \log(\pi(s, a; \theta)) Q^{\pi(\bullet;\theta)}(s, a)$$

- For average reward objective:

$$Q^\pi(s, a) = \mathbb{E}^\pi\left[\sum_{t \geq 0}(R_t - J(\theta)) | S_0 = s, A_0 = a\right]$$

- For discounted expected rewards objective: Q-function defined as usual.
- Proof given in the appendix

Policy Score

- For any state-action pair (s, a), the term

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

  is referred as the **policy score**

Example 1: Softmax Policy Gradient

- State-action pairs weighted by linear combination of features

$$\pi(s, a; \theta) = \frac{\exp(\phi^\top(s, a)\theta)}{\sum_{a'} \exp(\phi^\top(s, a')\theta)}$$

- The score function

$$\nabla_\theta \log \pi(s, a; \theta) = \phi(s, a) - \frac{\sum_{a'} \phi(s, a) \exp(\phi^\top(s, a)\theta)}{\sum_{a'} \exp(\phi^\top(s, a')\theta)}$$

or equivalently,

$$\nabla_\theta \log \pi(s, a; \theta) = \phi(s, a) - \mathbb{E}_{a' \sim \pi(s, \bullet; \theta)} \phi(s, a')$$

Example 2: Continuous Action Space

- Action space: set of real numbers $\mathcal{A} = \mathbb{R}$
- Policy approximator:

$$\pi(s, a, \theta) = \frac{1}{\sqrt{2\pi}\sigma(s;\theta)} \exp\left(-\frac{(a - \mu(s;\theta))^2}{2\sigma^2(s;\theta)}\right),$$

  where μ and σ are mean and deviation function approximators
- Linear function approximator with feature vectors $\phi_\mu(s)$ and $\phi_\sigma(s)$
  - $\mu(s;\theta) = \phi_\mu^\top(s)\theta_\mu$ and $\sigma(s;\theta) = \phi_\sigma^\top(s)\theta_\sigma$
  - $\nabla_{\theta_\mu} \log \pi(s, a, \theta) = \frac{a-\mu(s;\theta)}{\sigma^2(s;\theta)}\phi_\mu(s)$
  - $\nabla_{\theta_\sigma} \log \pi(s, a, \theta) = \frac{(a-\mu(s;\theta))^2-\sigma^2(s;\theta)}{\sigma^2(s;\theta)}\phi_\sigma(s)$

Example 3: Bernoulli, Logistic Example

- Actions space: binary, $\{\mathbf{0}, \mathbf{1}\}$
- Policy approximator:

$$\boldsymbol{\pi}(\mathbf{1}, \boldsymbol{s}; \boldsymbol{\theta}) = \mathbf{1} - \boldsymbol{\pi}(\mathbf{0}, \boldsymbol{s}; \boldsymbol{\theta}) = \boldsymbol{p}(\boldsymbol{s}; \boldsymbol{\theta})$$

  where $\boldsymbol{p}(\boldsymbol{s}; \boldsymbol{\theta})$ is a function approximator
- Linear function approximator with feature vectors $\phi(\boldsymbol{s})$
  - Logistic function $\boldsymbol{\sigma}(\boldsymbol{x}) = [\mathbf{1} + \exp(-\boldsymbol{x})]^{-\mathbf{1}}$
  - For exponential soft-max policy $\boldsymbol{p}(\boldsymbol{s}; \boldsymbol{\theta}) = \boldsymbol{\sigma}(\phi^\top(\boldsymbol{s})\boldsymbol{\theta})$
  - $\nabla_{\boldsymbol{\theta}} \log(\boldsymbol{\pi}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta})) = (\boldsymbol{a} - \boldsymbol{\sigma}(\phi^\top(\boldsymbol{s})\boldsymbol{\theta}))\phi(\boldsymbol{s})$

Lecture Outline

1. Policy Gradient Method Introduction
2. Policy Gradient Theorem
3. **REINFORCE and Actor Critic Algorithms**
4. Advantage Actor-Critic (A2C)

REINFORCE: MC Policy Gradient Algorithm

- To maximize $\boldsymbol{J}(\boldsymbol{\theta})$, we apply (stochastic) gradient ascent algorithm

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\alpha}_t \nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta}_t)$$

- According to the policy gradient theorem,

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta}) = \sum_{s,a} \boldsymbol{\mu}^{\pi(\bullet;\boldsymbol{\theta})}(s, a) \nabla_{\boldsymbol{\theta}} \log(\boldsymbol{\pi}(s, a; \boldsymbol{\theta})) \boldsymbol{Q}^{\pi(\bullet;\boldsymbol{\theta})}(s, a)$$

- Focus on the average reward setting
- $\boldsymbol{\mu}^{\pi}$ (stationary state-action distribution) is unknown: use empirical state-action distribution $\{(S_t, A_t)\}_t$ as an approx
- $\boldsymbol{Q}^{\pi}$ is unknown: use empirical return $G_t = \sum_{j=t}^{T} R_j$ as an approx

REINFORCE: Pseudocode

- **Initialization**: θ arbitrary
- **For each** episode $(S_0, A_0, R_0, \cdots, S_T, A_T, R_T)$ generated using policy $\pi(\bullet;\theta)$

  For $t = 0, 1, 2, \cdots, T$ **do**:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log(\pi(S_t, A_t; \theta)) G_t$$

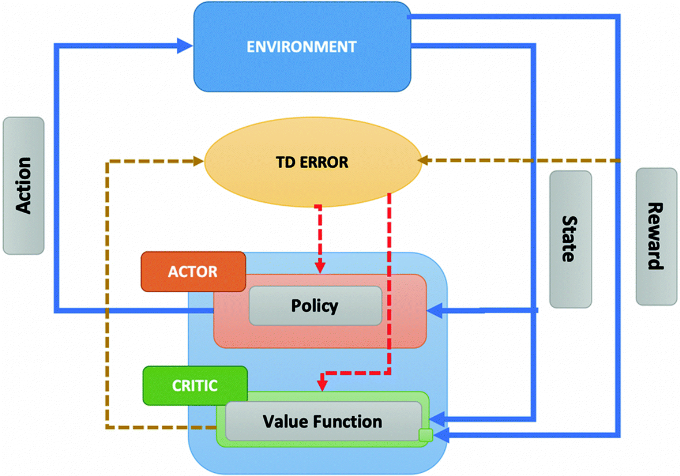
  end for

return θ

Actor-Critic Algorithm

- MC policy gradient algorithm may have a large **variance**
  - Return involves many state transitions, many actions and many rewards
- Solution sought by using **actor-critic algorithms**
- Actor-critic algorithms combine **policy gradient** with **value function estimation**

Actor-Critic Algorithm (Cont'd)



- **Critic** uses function approximator to learn value function
- **Actor** uses policy approximator to learn optimal policy

From `https://link.springer.com/chapter/10.1007/978-981-13-8285-7_11`

Actor-Critic Control

- **Critic**: estimates $Q^{\pi(\bullet;\theta)}(s, a)$ by a function approximator $\widehat{Q}(s, a; \omega)$
  - The critic performs **policy evaluation**
  - Standard methods can be applied: MC, TD($\mathbf{0}$), TD($\boldsymbol{\lambda}$), gradient-based methods

- **Actor**: updates policy parameter θ
  - The actor performs control using approximate policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{(s,a)\sim\mu} \nabla_\theta \log(\pi(s, a; \theta)) \widehat{Q}(s, a; \omega)$$

  - Parameter update
    - Average reward setting

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log(\pi(S_t, A_t; \theta)) \widehat{Q}(S_t, A_t; \omega)$$

    - Discounted reward setting

$$\theta \leftarrow \theta + \alpha \gamma^t \nabla_\theta \log(\pi(S_t, A_t; \theta)) \widehat{Q}(S_t, A_t; \omega)$$

Example: Actor-Critic with Linear Value Function

- Linear value function approximator

$$\widehat{\boldsymbol{Q}}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\omega}) = \boldsymbol{\phi}^\top(\boldsymbol{s}, \boldsymbol{a})\boldsymbol{\omega}$$

- Focus on the discounted reward setting
- **Critic**: updates $\boldsymbol{\omega}$ by linear TD($\mathbf{0}$)

$$\boldsymbol{\omega_{t+1}} = \boldsymbol{\omega_t} + \boldsymbol{\eta\phi}(\boldsymbol{S_t}, \boldsymbol{A_t})(\boldsymbol{R_t} + \gamma\boldsymbol{\phi}^\top(\boldsymbol{S_{t+1}}, \boldsymbol{A_{t+1}})\boldsymbol{\omega_t} - \boldsymbol{\phi}^\top(\boldsymbol{S_t}, \boldsymbol{A_t})\boldsymbol{\omega_t})$$

Pseudocode

- **Initialization**: s, θ, ω
- **For each** episode:

  Initialize $t = 0$

  Sample action a from $\pi(\bullet, s; \theta)$

  Repeat until s is terminal

  Receive reward r and next state s'

  Sample action a' from $\pi(\bullet, s; \theta)$

  $\theta \leftarrow \theta + \alpha\gamma^t \log(\pi(s, a; \theta))\phi^\top(s, a)\omega$

  $\omega \leftarrow \omega + \eta\phi(s, a)[r + \gamma\phi^\top(s', a')\omega - \phi^\top(s, a)\omega]$

  $a \leftarrow a'$ and $s \leftarrow s'$

  $t \leftarrow t + 1$

Bias-Variance Tradeoff

- **REINFORCE** uses Return G_t, an unbiased estimate of $Q^{\pi(\bullet;\theta)}(s, a)$
- **Actor-critic** uses $\widehat{Q}(s, a; \omega)$, a biased estimate of $Q^{\pi(\bullet;\theta)}(s, a)$
- REINFORCE gradient has **high variance** and **zero bias**
- Actor-critic gradient has **low variance** and **some bias**
- Similar to Pros & Cons of MC vs TD (Lecture 4, p13)
- Perhaps surprisingly, actor-critic gradient can be **unbiased** under certain conditions (see next slide)

Compatible Function Approximation Theorem

Theorem

Assume the following two conditions:

(C1) *Compatibility of value function approximator and the policy*

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

(C2) *Value function approximator minimizes the mean-squared error:*

$$\mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\boldsymbol{\mu}^{\pi(\bullet;\theta)}} \left[\boldsymbol{Q}^{\pi(\bullet;\omega)}(\boldsymbol{s}, \boldsymbol{a}) - \widehat{\boldsymbol{Q}}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\omega})\right]^2$$

Then the gradient is unbiased

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\boldsymbol{\mu}^{\pi(\bullet;\theta)}} \nabla_{\boldsymbol{\theta}}(\log \boldsymbol{\pi}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}))\widehat{\boldsymbol{Q}}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\omega})$$

Compatible Linear Function Approximation

- Consider the soft-max policy, for a given state-action feature vector $\phi(s, a)$:

$$\pi(s, a; \theta) = \frac{\exp(\phi^\top(s, a)\theta)}{\sum_{a'} \exp(\phi^\top(s, a')\theta)}$$

- Compatibility condition requires that

$$\nabla_\omega \widehat{Q}(s, a'; \omega) = \nabla_\theta \log(\pi(s, a; \theta)) = \underbrace{\phi(s, a) - \sum_{a'} \phi(s, a')\pi(s, a'; \theta)}_{\text{centered state-action features vectors}}$$

which leads to a linear approximation for the value function

$$\widehat{Q}(s, a'; \omega) = \left[\phi(s, a) - \sum_{a'} \phi(s, a')\pi(s, a'; \theta)\right]^\top \omega$$

Convergence Theorem

Theorem

Assume

- $\pi(\bullet;\theta)$ *and* $\widehat{Q}(\bullet;\omega)$ *are differentiable functions*
- *Compatibility assumption holds*
- *The Hessian matrix* $\nabla^2_\theta \pi(s,a;\theta)$ *are uniformly bounded away from infinity*
- *Step sizes are such that* $\sum_t \alpha_t = \infty$ *and* $\sum_t \alpha_t^2 < \infty$
- *At each step,* ω_t *is chosen to be the solution of*

$$\mathbb{E}_{(s,a)\sim\mu}\pi(s,a;\theta_t)[Q^{\pi(\bullet;\theta_t)}(s,a) - \widehat{Q}(s,a;\omega)]\nabla_\omega \widehat{Q}(s,a;\omega) = 0$$

Then $\{\theta_t\}_t$ *are convergent in the sense that* $\lim_{t\to\infty} \|\nabla_\theta J(\theta_t)\| \to 0$.

Separation of Timescales

- The last condition defines ω_t as a solution of a fixed-point equation which has the policy's parameter vector θ_t as a parameter
- In practice, we update ω_t using stochastic gradient descent algorithm. SGD would update ω_t in a similar manner with a larger step size than α_t. It ensures ω converges faster than θ, thus closer to the solution of the fixed-point equation at each time
- This can be seen as a **separation of timescales**:
  - Critic updates the value function approximator at a **faster** timescale trying to evaluate the current policy chosen by the actor
  - Actor varies the policy's parameter more **slowly** to allow the critic to evaluate the current policy
- Similar assumptions are imposed in gradient Q-learning algorithms [Maei et al., 2010]

Lecture Outline

1. Policy Gradient Method Introduction
2. Policy Gradient Theorem
3. REINFORCE and Actor Critic Algorithms
4. **Advantage Actor-Critic (A2C)**

Variance Reduction Using a Baseline

- Recall that policy parameter update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\alpha}\gamma^{\boldsymbol{t}} \nabla_{\boldsymbol{\theta}} \log(\boldsymbol{\pi}(\boldsymbol{S_t}, \boldsymbol{A_t}; \boldsymbol{\theta}))\widehat{\boldsymbol{Q}}(\boldsymbol{S_t}, \boldsymbol{A_t}; \boldsymbol{\omega})$$

- For any $\boldsymbol{\theta}$, when $\boldsymbol{A_t} \sim \boldsymbol{\pi}(\boldsymbol{S_t}, \bullet, \boldsymbol{\theta})$

$$\mathbb{E}[\nabla_{\boldsymbol{\theta}} \log(\boldsymbol{\pi}(\boldsymbol{S_t}, \boldsymbol{A_t}, \boldsymbol{\theta}))|\boldsymbol{S_t}] = \mathbf{0}$$

- For any baseline function $\boldsymbol{B}(\boldsymbol{s})$, consider the update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\alpha}\gamma^{\boldsymbol{t}} \nabla_{\boldsymbol{\theta}} \log(\boldsymbol{\pi}(\boldsymbol{S_t}, \boldsymbol{A_t}; \boldsymbol{\theta}))[\widehat{\boldsymbol{Q}}(\boldsymbol{S_t}, \boldsymbol{A_t}; \boldsymbol{\omega}) - \boldsymbol{B}(\boldsymbol{S_t})]$$

- The **mean** of gradient is the same without baseline
- However, the **variance** of the gradient would be smaller with a properly chosen $\boldsymbol{B}$

Variance Reduction Using a Baseline (Cont'd)

- Consider the baseline that minimizes the variance of the gradient
- For any random variable $\boldsymbol{Z}$, the mean $\mathbb{E}\boldsymbol{Z}$ minimizes $\arg\min_{\boldsymbol{z}} \mathbb{E}(\boldsymbol{Z}-\boldsymbol{z})^2$
- To minimize variance of the gradient $\nabla_{\boldsymbol{\theta}} \log(\pi(\boldsymbol{S_t}, \boldsymbol{A_t}; \boldsymbol{\theta}))[\widehat{\boldsymbol{Q}}(\boldsymbol{S_t}, \boldsymbol{A_t}; \boldsymbol{\omega}) - \boldsymbol{B}(\boldsymbol{S_t})]$, the baseline is set to the conditional mean of Q-function given the state
- i.e., $\boldsymbol{B}(\boldsymbol{s}) = \sum_{\boldsymbol{a}} \pi(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta})\widehat{\boldsymbol{Q}}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\omega})$, e.g., the estimated state-value
- Similar ideas have been employed in gradient-based algorithms in HW1

Policy Gradient Using Advantage Function

- Advantage function: $\boldsymbol{A}^{\pi(\bullet;\theta)}(s, a) = \boldsymbol{Q}^{\pi(\bullet;\theta)}(s, a) - \boldsymbol{V}^{\pi(\bullet;\theta)}(s)$
- Policy gradient based on advantage function

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta}) = \mathbb{E}_{(s,a)\sim\mu^{\pi(\bullet;\theta)}} \nabla_{\boldsymbol{\theta}} \log(\pi(s, a; \boldsymbol{\theta}))\boldsymbol{A}^{\pi(\bullet;\theta)}(s, a)$$

- The advantage function reduces the variance of policy gradient

An Approach for Estimating Advantage Function

- The critic may compute estimators of both value functions

$$\widehat{\boldsymbol{Q}}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\omega}) \text{ for } \boldsymbol{Q}^{\boldsymbol{\pi}(\bullet;\boldsymbol{\theta})}(\boldsymbol{s}, \boldsymbol{a})$$

  and

$$\widehat{\boldsymbol{V}}(\boldsymbol{s}; \boldsymbol{\omega}) \text{ for } \boldsymbol{V}^{\boldsymbol{\pi}(\bullet;\boldsymbol{\theta})}(\boldsymbol{s})$$

  which can be done by standard methods such as TD learning

- The estimator of the advantage function

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

Another Approach

- $r + \gamma V^{\pi(\bullet;\theta)}(s') - V^{\pi(\bullet;\theta)}(s)$ is **unbiased** to $A^{\pi(\bullet;\theta)}(s, a)$

$$
\begin{aligned}
& \mathbb{E}[r + \gamma V^{\pi(\bullet;\theta)}(s') - V^{\pi(\bullet;\theta)}(s) | a, s] \\
= \; & \mathbb{E}[r + \gamma V^{\pi(\bullet;\theta)}(s') - Q^{\pi(\bullet;\theta)}(s, a) + Q^{\pi(\bullet;\theta)}(s, a) - V^{\pi(\bullet;\theta)}(s) | a, s] \\
= \; & Q^{\pi(\bullet;\theta)}(s, a) - V^{\pi(\bullet;\theta)}(s) = A^{\pi(\bullet;\theta)}(s, a)
\end{aligned}
$$

- As such,

$$
\nabla_\theta J(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi(\bullet;\theta)}} \nabla_\theta \log(\pi(s, a; \theta))[r + \gamma V^{\pi(\bullet;\theta)}(s') - V^{\pi(\bullet;\theta)}(s)]
$$

- No need to estimate the advantage. It suffices to estimate the state-value and use the estimator to compute the policy gradient

Critic Policy Evaluation Methods

- When specialized to linear methods $\widehat{V}(s;\omega) = \phi^\top(s)\omega$, the critic can use different targets to evaluate

$$\omega_{t+1} \leftarrow \omega_t + \eta_t[v_t - \phi^\top(S_t)\omega_t]\phi(S_t)$$

- The target is defined differently for different methods
  - MC: $v_t = G_t$
  - TD: $v_t = R_t + \gamma\widehat{V}(S_{t+1})$
  - TD(λ): $v_t = {G_t}^{\lambda}$

Actor Policy Gradient Methods

- The policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{(s,a)\sim\mu^{\pi(\bullet;\theta)}} \nabla_\theta \log(\pi(s, a; \theta)) A^{\pi(\bullet;\theta)}(s, a)$$

- Gradient-based method

$$\theta \leftarrow \theta + \alpha\gamma^t \nabla_\theta \log(\pi(S_t, A_t; \theta)) \widehat{A}(S_t, A_t; \omega)$$

- Examples:
  - MC: $\widehat{A}(S_t, A_t; \omega) = G_t - \widehat{V}(S_t; \omega)$
  - TD: $\widehat{A}(S_t, A_t; \omega) = R_t + \gamma \widehat{V}(S_{t+1}; \omega) - \widehat{V}(S_t; \omega)$

Summary

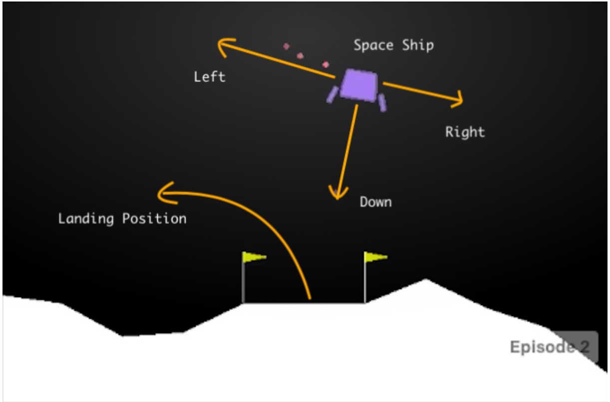
Policy Function Approximation

Value Function Approximation	No	Yes
No	Value-based (tabular)	REINFORCE
Yes	Value-based	Actor-Critic

- **Value-based**
  - Tabular (Lectures 3 & 4)
  - Function approx (Lectures 5 & 7)
- **REINFORCE**
  - No value function
  - Learn policy
- **Actor-critic**
  - Learn value
  - Learn policy
- **Advantage actor-critic**
  - Variance reduction

Seminar Exercise

- Solution to HW7 (Deadline: Wed 12pm)
- Implementation of DQN on LunarLander



Taken from https://shiva-verma.medium.com/solving-lunar-lander-openaigym-reinforcement-learning-785675066197

References I

Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *ICML*, 2010.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Questions

Appendix: Proof of Policy Gradient Theorem

- We focus on the discounted reward setting. Proofs in the average reward setting can be found in Sutton et al. [1999]
- Basic identities

$$(A) \qquad V^{\pi}(s) = \sum_{a} \pi(a|s) Q^{\pi}(s, a)$$

$$(B) \qquad Q^{\pi}(s, a) = R_{s}^{a} + \gamma \sum_{s'} P_{s,s'}^{a} V^{\pi}(s')$$

$$(C) \qquad \nabla_{\theta} V^{\pi}(s) = \sum_{a} [\nabla_{\theta} \pi(a|s)] Q^{\pi}(s, a) + \sum_{a} \pi(a|s) [\nabla_{\theta} Q^{\pi}(s, a)]$$

$$(D) \qquad \nabla_{\theta} Q^{\pi}(s, a) = \gamma \sum_{s'} P_{s,s'}^{a} \nabla_{\theta} V^{\pi}(s')$$

Appendix: Proof (Cont'd)

$$\nabla_\theta V^\pi(s) \overset{(C)}{=} \sum_a [\nabla_\theta \pi(a|s)] Q^\pi(s,a) + \sum_a \pi(a|s)[\nabla_\theta Q^\pi(s,a)]$$

$$\overset{(D)}{=} \sum_a \pi(a|s)[\nabla_\theta \log(\pi(a|s))] Q^\pi(s,a) + \gamma \underbrace{\sum_{a,s'} \pi(a|s) P^a_{s,s'} \nabla_\theta V^\pi(s')}_{I}$$

Now, consider I. Similarly, we have

$$I = \sum_{a,s',a'} \pi(a|bs) P^a_{s,s'} \pi(a'|s')[\nabla_\theta \log(\pi(a'|s'))] Q^\pi(s',a')$$

$$+\gamma \sum_{a,s',a',s''} \pi(a|s) P^a_{s,s'} \pi(a'|s') P^{a'}_{s',s''} \nabla_\theta V^\pi(s'')$$

Appendix: Proof (Cont'd)

Recursively applying the first identity, we obtain

$$\nabla_{\theta} \boldsymbol{V}^{\pi}(\boldsymbol{s}) = \boldsymbol{\mu}^{\pi(\bullet;\theta)}(\boldsymbol{s'}, \boldsymbol{a'}; \boldsymbol{s}) \nabla_{\theta} \log(\pi(\boldsymbol{s'}, \boldsymbol{a'})) \boldsymbol{Q}^{\pi}(\boldsymbol{s'}, \boldsymbol{a'})$$

where

$$\boldsymbol{\mu}^{\pi(\bullet;\theta)}(\boldsymbol{s'}, \boldsymbol{a'}; \boldsymbol{s}) = \sum_{t \geq 0} \gamma^{t} \pi(\boldsymbol{s'}, \boldsymbol{a'}) \Pr^{\pi(\bullet;\theta)}(\boldsymbol{S_t} = \boldsymbol{s'} | \boldsymbol{S_0} = \boldsymbol{s})$$