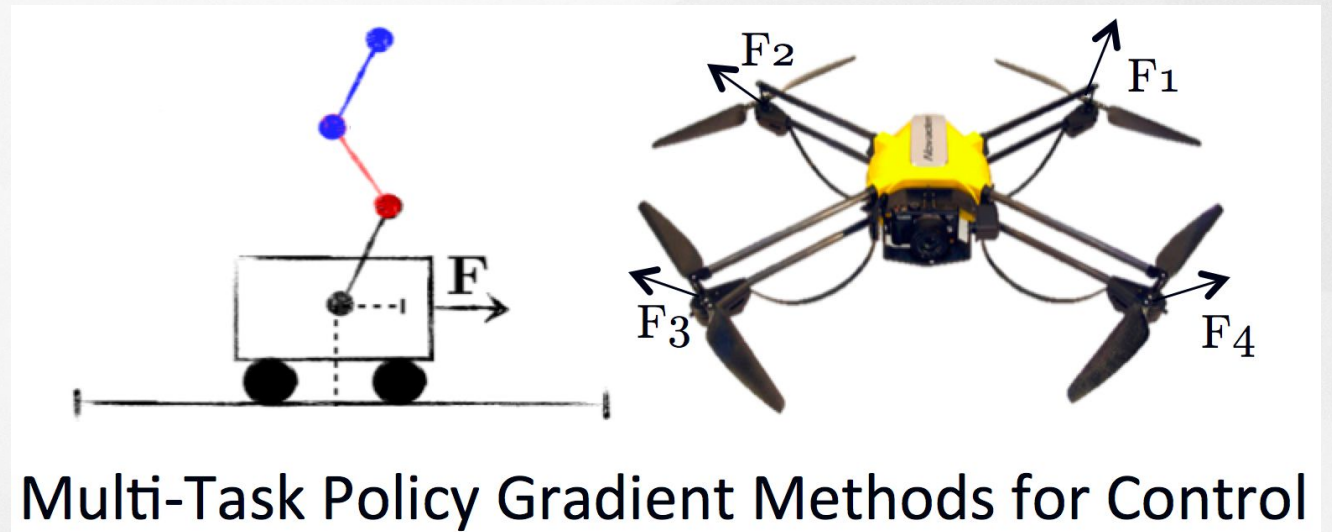


## Chapter 06. 스스로 전략을 짜는 강화학습 ( Reinforcement Learning )

# Policy Gradient



# Value Based RL

[http://www.modulabs.co.kr/RL\\_library/3305](http://www.modulabs.co.kr/RL_library/3305)

\* **Value-based reinforcement learning** vs Policy-based reinforcement learning

- In the last lecture we approximated the value or action-value function using parameters  $\theta$ ,

$$\begin{aligned} V_{\theta}(s) &\approx V^{\pi}(s) \\ Q_{\theta}(s, a) &\approx Q^{\pi}(s, a) \end{aligned}$$

- A policy was generated directly from the value function
  - e.g. using  $\epsilon$ -greedy

# Policy Based RL

[http://www.modulabs.co.kr/RL\\_library/3305](http://www.modulabs.co.kr/RL_library/3305)

\* Value-based reinforcement learning vs **Policy-based reinforcement learning**

- In this lecture we will directly parametrise the policy

$$\pi_{\theta}(s, a) = \mathbb{P}[a \mid s, \theta]$$

- We will focus again on model-free reinforcement learning

# Policy Gradient

[http://www.modulabs.co.kr/RL\\_library/3305](http://www.modulabs.co.kr/RL_library/3305)

## Advantages:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn **stochastic policies**

## Disadvantages:

- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

# Value Based RL

[http://www.modulabs.co.kr/RL\\_library/3305](http://www.modulabs.co.kr/RL_library/3305)

## Unstable Policy

- Greedy updates

$$\theta_{\pi'} = \arg \max_{\theta} \mathbb{E}_{\pi_{\theta}}[Q^{\pi}(s, a)]$$

- $V^{\pi_0} \xrightarrow{\text{small change}} \pi_1 \xrightarrow{\text{large change}} V^{\pi_1} \xrightarrow{\text{large change}} \pi_2 \xrightarrow{\text{large change}}$
- Potentially **unstable** learning process with **large policy jumps**
- Policy Gradient updates

$$\theta_{\pi'} = \theta_{\pi} + \alpha \left. \frac{dJ(\theta)}{d\theta} \right|_{\theta=\theta^{\pi}}$$

- $V^{\pi_0} \xrightarrow{\text{small change}} \pi_1 \xrightarrow{\text{small change}} V^{\pi_1} \xrightarrow{\text{small change}} \pi_2 \xrightarrow{\text{small change}}$
- **Stable** learning process with **smooth policy improvement**

# Value Based RL

[http://www.modulabs.co.kr/RL\\_library/3305](http://www.modulabs.co.kr/RL_library/3305)

## Stochastic Policy

Example: Rock-Paper-Scissors



Sometimes you need stochastic policy!!!



# Policy Gradient

## Policy Objective Function

- Goal: given policy  $\pi_\theta(s, a)$  with parameters  $\theta$ , find best  $\theta$
- But how do we measure the quality of a policy  $\pi_\theta$ ?
- In episodic environments we can use the start value

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta} [v_1]$$

- In continuing environments we can use the average value

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- Or the average reward per time-step

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

$$d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr \{s_t = \tilde{s} | s_0, \pi\}$$

- where  $d^{\pi_\theta}(s)$  is stationary distribution of Markov chain for  $\pi_\theta$

# Policy Gradient

Objective Function의 Gradient를 구하는 방법

1. **Finite Difference Policy Gradient**
2. **Monte-Carlo Policy Gradient**
3. **Actor-Critic Policy Gradient**



# Policy Gradient

## Finite Difference Policy Gradient

- To evaluate policy gradient of  $\pi_{\theta}(s, a)$
- For each dimension  $k \in [1, n]$ 
  - Estimate  $k$ th partial derivative of objective function w.r.t.  $\theta$
  - By perturbing  $\theta$  by small amount  $\epsilon$  in  $k$ th dimension

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

where  $u_k$  is unit vector with 1 in  $k$ th component, 0 elsewhere

- Uses  $n$  evaluations to compute policy gradient in  $n$  dimensions
- Simple, noisy, inefficient - but sometimes effective
- Works for arbitrary policies, even if policy is not differentiable

# Policy Gradient

## Monte-Carlo Policy Gradient

- We now compute the policy gradient analytically
- Assume policy  $\pi_\theta$  is differentiable whenever it is non-zero
- and we know the gradient  $\nabla_\theta \pi_\theta(s, a)$
- Likelihood ratios exploit the following identity

$$\begin{aligned}\nabla_\theta \pi_\theta(s, a) &= \pi_\theta(s, a) \frac{\nabla_\theta \pi_\theta(s, a)}{\pi_\theta(s, a)} \\ &= \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a)\end{aligned}$$

- The score function is  $\nabla_\theta \log \pi_\theta(s, a)$

# Policy Gradient

## Monte-Carlo Policy Gradient

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\pi_{\theta}} [r] \\ &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(s, a) \mathcal{R}_{s,a} \\ \nabla_{\theta} J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(s, a) \nabla_{\theta} \log \pi_{\theta}(s, a) \mathcal{R}_{s,a} \\ &= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) r] \end{aligned}$$

# Policy Gradient

## Monte-Carlo Policy Gradient

### Policy Gradient Theorem

- The policy gradient theorem generalises the likelihood ratio approach to multi-step MDPs
- Replaces instantaneous reward  $r$  with long-term value  $Q^\pi(s, a)$
- Policy gradient theorem applies to start state objective, average reward and average value objective

### Theorem

For any differentiable policy  $\pi_\theta(s, a)$ ,  
for any of the policy objective functions  $J = J_1, J_{avR}$ , or  $\frac{1}{1-\gamma} J_{avV}$ ,  
the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)]$$

# Policy Gradient

## Monte-Carlo Policy Gradient

**\*How can we express stochastic policy?**

### 1. Softmax Policy

$$\pi(s, a) = \frac{e^{\theta^T \phi_{sa}}}{\sum_b e^{\theta^T \phi_{sb}}}, \quad \forall s \in \mathcal{S}, s \in \mathcal{A},$$

**Stochastic Policy Model**

### 2. Gaussian Policy

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

# Monte-Carlo Policy Gradient (REINFORCE)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return  $v_t$  as an unbiased sample of  $Q^{\pi_\theta}(s_t, a_t)$

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

## function REINFORCE

Initialise  $\theta$  arbitrarily

**for** each episode  $\{s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$  **do**

**for**  $t = 1$  to  $T - 1$  **do**

$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$

**end for**

**end for**

**return**  $\theta$

**end function**



# Policy Gradient

## Actor-Critic Policy Gradient

- Monte-Carlo policy gradient still has high variance
- We use a **critic** to estimate the action-value function,

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

- Actor-critic algorithms maintain two sets of parameters
  - Critic** Updates action-value function parameters **w**
  - Actor** Updates policy parameters  **$\theta$** , in direction suggested by critic
- Actor-critic algorithms follow an *approximate* policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)]$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$$

# Policy Gradient

## Actor-Critic Policy Gradient

### Reducing Variance Using a Baseline

- We subtract a baseline function  $B(s)$  from the policy gradient
- This can reduce variance, without changing expectation

$$\begin{aligned}\mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) B(s)] &= \sum_{s \in \mathcal{S}} d^{\pi_{\theta}}(s) \sum_a \nabla_{\theta} \pi_{\theta}(s, a) B(s) \\ &= \sum_{s \in \mathcal{S}} d^{\pi_{\theta}} B(s) \nabla_{\theta} \sum_{a \in \mathcal{A}} \pi_{\theta}(s, a) \\ &= 0\end{aligned}$$

- A good baseline is the state value function  $B(s) = V^{\pi_{\theta}}(s)$
- So we can rewrite the policy gradient using the **advantage function**  $A^{\pi_{\theta}}(s, a)$

$$A^{\pi_{\theta}}(s, a) = Q^{\pi_{\theta}}(s, a) - V^{\pi_{\theta}}(s)$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) A^{\pi_{\theta}}(s, a)]$$

# Policy Gradient

## Actor-Critic Policy Gradient

### 2.4. Off-Policy Actor-Critic

It is often useful to estimate the policy gradient *off-policy* from trajectories sampled from a distinct behaviour policy  $\beta(a|s) \neq \pi_\theta(a|s)$ . In an off-policy setting, the performance objective is typically modified to be the value function of the target policy, averaged over the state distribution of the behaviour policy (Degrís et al., 2012b),

$$\begin{aligned} J_\beta(\pi_\theta) &= \int_{\mathcal{S}} \rho^\beta(s) V^\pi(s) ds \\ &= \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^\beta(s) \pi_\theta(a|s) Q^\pi(s, a) da ds \end{aligned}$$

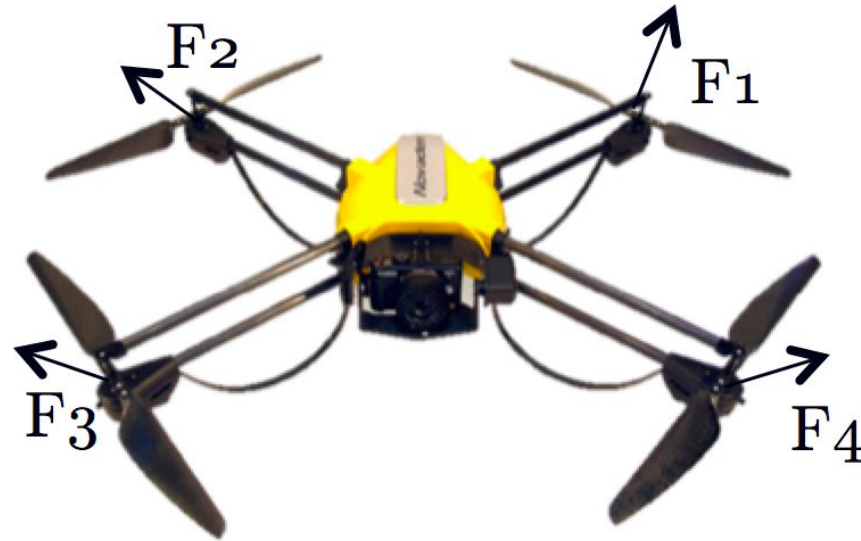
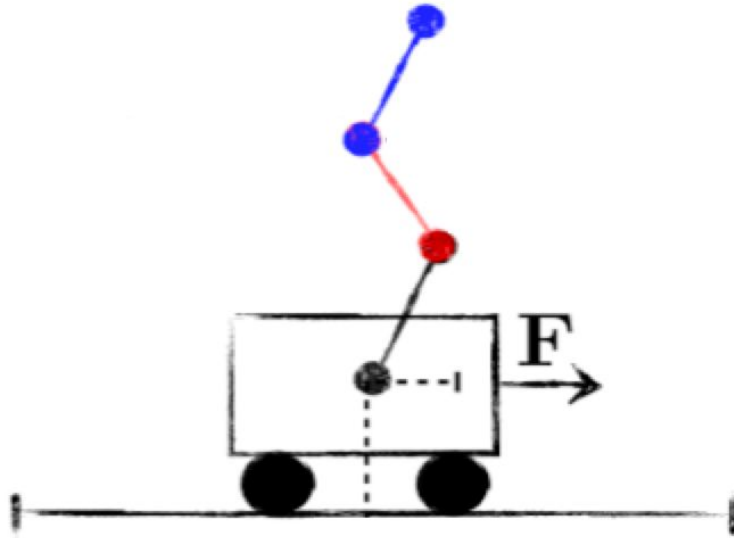
Differentiating the performance objective and applying an approximation gives the *off-policy policy-gradient* (Degrís et al., 2012b)

$$\nabla_\theta J_\beta(\pi_\theta) \approx \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^\beta(s) \nabla_\theta \pi_\theta(a|s) Q^\pi(s, a) da ds \quad (4)$$

$$= \mathbb{E}_{s \sim \rho^\beta, a \sim \beta} \left[ \frac{\pi_\theta(a|s)}{\beta_\theta(a|s)} \nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a) \right] \quad (5)$$

# Policy Gradient

[http://www.modulabs.co.kr/RL\\_library/3305](http://www.modulabs.co.kr/RL_library/3305)



Multi-Task Policy Gradient Methods for Control

<https://talkingaboutme.tistory.com/entry/RL-Policy-Gradient-Algorithms>

• *Thank you*