Reinforcement Learning: Policy Gradient

AI/ML Teaching

Goals & Keyword

- Policy-Based RL vs Value-Based RL

- Monte-Carlo Policy Gradient

- Actor-Critic Policy Gradient

Policy-Based RL vs Value-Based RL

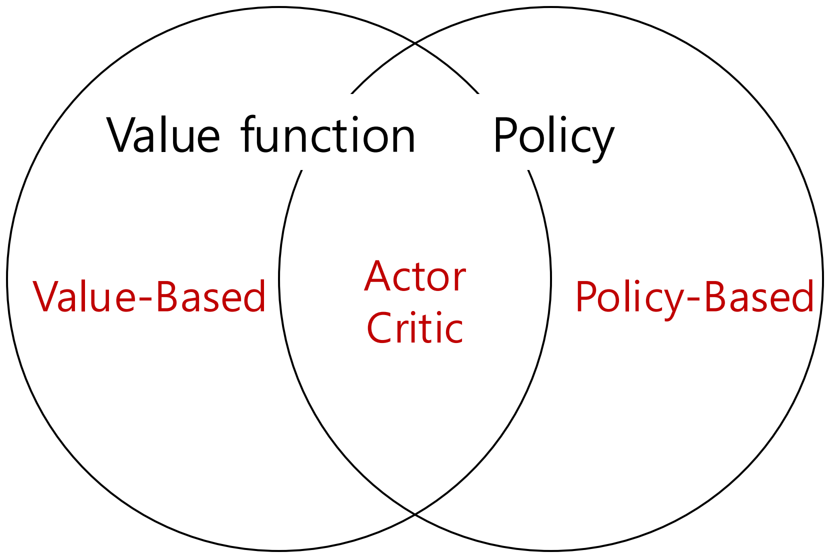
- Value-based RL: A policy is generated directly from the value function, $V^{\pi}(s)$ or $Q^{\pi}(s,a)$
  - e.g. ϵ-greedy

- Policy-based RL: Directly parameterize the policy

$$\pi(s,a) = \mathbb{P}[a|s]$$

Policy-Based RL vs Value-Based RL

- Value Based
  - Learnt value function
  - Implicit policy
- Policy Based
  - No value function
  - Learnt policy
- Actor-Critic
  - Learnt value function
  - Learnt policy



Policy-Based RL vs Value-Based RL

- Policy-Based RL

  (+) Better convergence properties

  (+) Effective in high-dimensional or **continuous** action spaces

  (+) Can learn **stochastic** policies

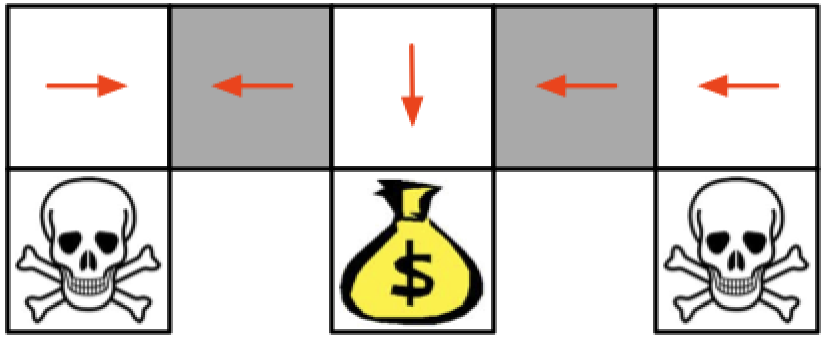
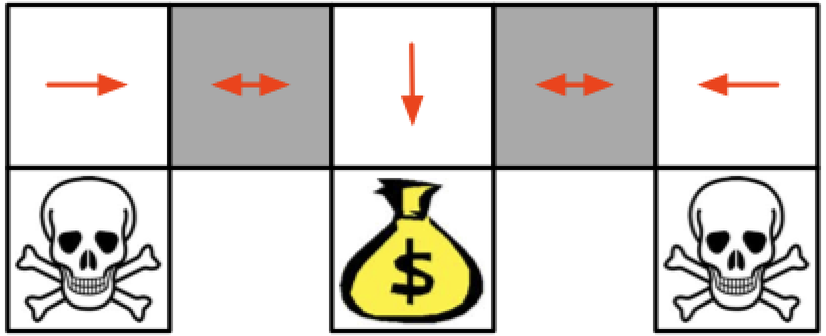
  (–) typically converge to a local rather than global optimum*

  (–) Evaluating a policy is typically inefficient and high variance

*Value-based RL could also converge to local optimum (function approx., bootstrapping, off-policy)

Stochastic policy

- Stochastic environment
- Non-fully observable state

Deterministic policy	Stochastic policy
*Agent cannot differentiate the grey states 	

Policy Search

Policy Objective Functions

- In episodic environments we can use the <u>start value</u>

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

- In continuing environments, we can use the <u>average value</u>

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

- Or the <u>average reward per time-step</u>

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

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

*Note: $J_{avV} \neq J_{avR}$

Policy gradient

- Find θ maximizing $J(\theta)$: policy ascent

- Numerical policy gradient $\dfrac{\partial J(\theta)}{\partial \theta_k} \approx \dfrac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$

- Compute Policy gradient analytically: Policy gradient theorem
  - We can compute $\nabla_\theta \pi_\theta(s,a)$

$$\nabla_\theta \pi_\theta(s,a) = \pi_\theta(s,a) \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} = \pi_\theta(s,a) \boxed{\nabla_\theta \log \pi_\theta(s,a)}$$

(def) Score function

Policy gradient theorem

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

(shortened derivation)

$\nabla_\theta J(\pi_\theta) = \nabla_\theta \sum_{s\in S} d(s) V^\pi(s)$

$= \sum_{s\in \mathcal{S}} d(s) \nabla_\theta V^\pi(s)$

$= \sum_{s\in S} d(s) \nabla_\theta (\sum_{a\in\mathcal{A}} Q^\pi(s,a) \pi_\theta(s,a))$

$= \sum_{s\in S} d(s) \sum_{a\in\mathcal{A}} Q^\pi(s,a) \nabla_\theta \pi_\theta(s,a)$

$= \sum_{s\in S} d(s) \sum_{a\in\mathcal{A}} Q^\pi(s,a) \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a)$

$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_\pi(s,a)]$

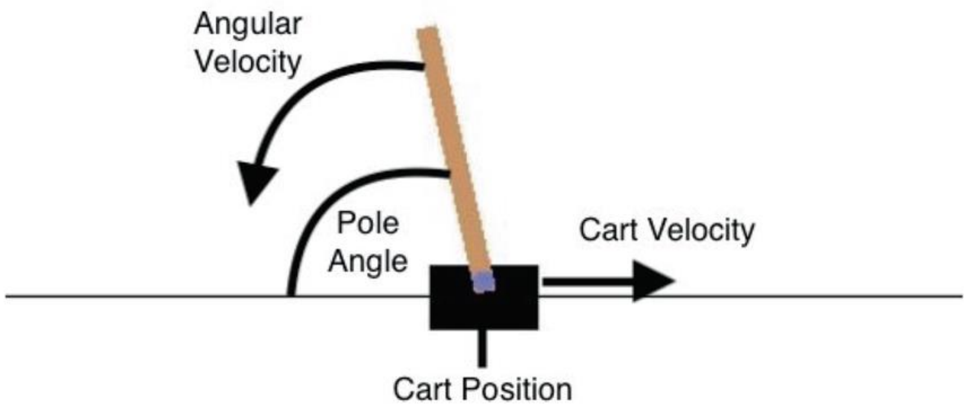
*Also refer to: https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html

(Example) Softmax policy

- Cartpole
  - Action: left/right
  - Softmax policy

Left: $\pi_\theta(a_0|s) = \frac{\exp\left(h_\theta(a_0|s)\right)}{\sum_{a\in\mathcal{A}}\exp\left(h_\theta(a|s)\right)} = p$

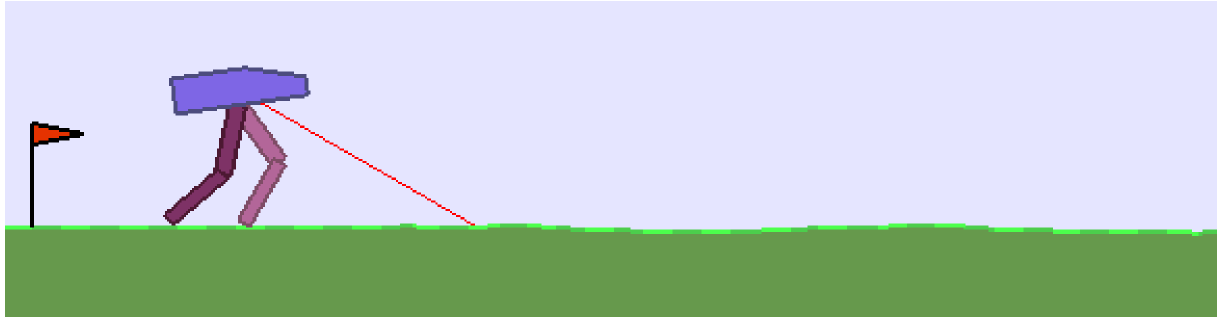
Right: $\pi_\theta(a_1|s) = \frac{\exp\left(h_\theta(a_1|s)\right)}{\sum_{a\in\mathcal{A}}\exp\left(h_\theta(a|s)\right)} = 1 - p$

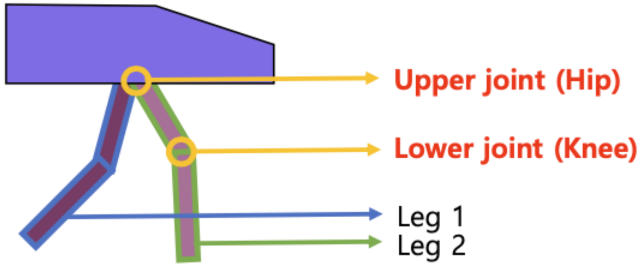


Action: sampling $(p/1-p)$ → left or right

(Example) Gaussian policy

- Bipedal walker
  - Action: motor speed values for each of the 4 joints at both hips and knees





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

Joint1: μ_{θ_1}, σ_{θ_1} from h_{θ_1}
Joint2: μ_{θ_2}, σ_{θ_2} from h_{θ_2}
Joint3: μ_{θ_3}, σ_{θ_3} from h_{θ_3}
Joint4: μ_{θ_4}, σ_{θ_4} from h_{θ_4}

Joint1: sample from $\mathcal{N}(\mu_{\theta_1}, \sigma_{\theta_1})$
Joint2: sample from $\mathcal{N}(\mu_{\theta_2}, \sigma_{\theta_2})$
Joint3: sample from $\mathcal{N}(\mu_{\theta_3}, \sigma_{\theta_3})$
Joint4: sample from $\mathcal{N}(\mu_{\theta_4}, \sigma_{\theta_4})$

Monte-Carlo Policy Gradient (REINFORCE)

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

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

function REINFORCE
  Initialize θ
  for each episode $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ 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 θ

Actor-Critic

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

- Actor-critic algorithms maintain two sets of parameters
  - Critic: updates action-value function parameters w
  - Actor: updates policy parameters θ

- Actor-critic algorithms follow an approximate policy gradient

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

Actor-Critic

function QAC
    Initialize s, θ
    Sample $a \sim \pi_\theta$
    for each step **do**
        Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_s^a$
        Sample action $a' \sim \pi_\theta(s', a')$
        $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$ // Critic
        $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a)\, Q_w(s, a)$ // Actor
        $w \leftarrow w + \beta \delta \nabla_w Q_w(s, a)$
        $a \leftarrow a', s \leftarrow s'$
    end for

Reference

- David Silver, COMPM050/COMPGI13 Lecture Notes
- Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction,” 2nd Ed.
- 김재훈, “Introduction to Policy Gradient,” DMQA Seminar