

# Lecture 17 - Policy gradient

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning  
Course Page: [\[Click\]](#)

# Policy-based and value-based algorithms

**Value-based algorithms** include Q-learning, temporal-difference learning, and policy and value iteration

- These algorithms learn the values of actions  $V(s)$  or  $Q(s, a)$  and then selected action  $a$  based on the action values  $\pi(s) = \arg \max_{a \in \mathcal{A}} Q(s, a)$ ;
- The policy does not exist without the action value estimates  $Q(s)$ .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Policy-based and value-based algorithms

Concerns about value-based methods.

- The vanilla approaches **can only address discrete action spaces** due to the  $\arg \max_{a \in \mathcal{A}}$  operation. However, in practice, the action space is usually continuous.
- Computing the action value functions  $Q(s, a)$  for **all state-action pair** is **costly** when the action and state spaces are large or continuous.
- The policy of Q-Learning is **deterministic** and  $\epsilon$ -greedy explore can be **inefficient**.
- It **implicitly and indirectly improves the policy** by improving the estimates of the values functions. However, we would think intuitively that improving the policy directly would be more efficient.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Policy-based and value-based algorithms

**Policy gradient** is the canonical approach for policy-based learning.

- Policy-based method directly parameterizes the policy function  $\pi_{\theta}(s)$  without calculating the value functions.
- We use the notation  $\theta \in R^d$  for the policy's parameter vector. We then write  $\pi(a | s, \theta) = \mathbb{P}(a_t = a | s_t = s, \theta)$  as the probability that action  $a$  is taken given that the environment is in state  $s$  with parameter  $\theta$ .
- A value function may still be used to **learn** the policy parameter, but is not required for action selection (will talk about it later in the actor-critic algorithm).



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Policy approximation with parametrization

**Discrete Action Space.** then a natural way to parameterize a policy is to form parameterized **state-action preferences**  $h(s, a, \theta)$  for each  $(s, a)$  pair and use a softmax distribution

$$\pi(a | s, \theta) = \frac{\exp(h(a, s, \theta))}{\sum_{a'} \exp(h(a', s, \theta))}. \quad (\text{softmax in action preferences})$$

- The **state-action preference** measures how the policy  $\pi_\theta$  prefer action  $a$  given state  $s$ . The actions with the highest preferences in each state are given the highest probabilities of being selected.



## Policy approximation with parametrization

**Discrete Action Space.** then a natural way to parameterize a policy is to form parameterized **state-action preferences**  $h(s, a, \theta)$  for each  $(s, a)$  pair and use a softmax distribution

$$\pi(a | s, \theta) = \frac{\exp(h(a, s, \theta))}{\sum_{a'} \exp(h(a', s, \theta))}. \quad (\text{softmax in action preferences})$$

- The **action preferences**  $h(a, s, \theta)$  can be parameterized arbitrarily. For example, it can simply be the linear combinations of features (as for the feature vectors  $x(a, s)$ )

$$h(a, s, \theta) = \theta^T x(a, s).$$



## Policy approximation with parametrization

**Continuous Action Space.** The policy can be defined as the normal probability density over a real-valued scalar action, with mean and standard deviation given by parametric function approximators

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

- We divide the policy's parameter vector into two parts,  $\theta = [\theta_\mu, \theta_\sigma]$ .
- One possible way to parametrize the mean and standard deviation is

$$\mu(s, \theta) = \theta_\mu^T \mathbf{x}_\mu(s), \quad \sigma(s, \theta) = \exp(\theta_\sigma^T \mathbf{x}_\sigma(s)),$$

where  $\mathbf{x}_\sigma(s)$  and  $\mathbf{x}_\mu(s)$  are feature vectors.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Policy approximation with parametrization

## Advantages of using parametrization

- It handles both discrete and continuous action spaces.
- It could be deterministic or stochastic. If the optimal policy is deterministic, then the preference values  $h(a, s, \theta)$  will be driven infinitely higher than all other actions.
- The choice of policy parametrization is sometimes a good way of injecting prior knowledge about the desired form of the policy into the learning.
- Policy gradient has stronger convergence guarantees than value-based method because of the smooth change in the probability.





# Policy Gradient Objective

Recall the gradient descent algorithm,  $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta)}$  where  $J(\theta)$  is our objective function and  $\alpha$  is the learning rate.

**For the episodic case**, where the episode terminates at some terminal state set, we define the objective function  $J(\theta)$  as

$$J(\theta) = V^{\pi_\theta}(s_0) = \sum_{s \in \mathcal{S}} \rho^{\pi_\theta}(s | s_0) r(s),$$

where  $s_0$  is the starting state,  $V^{\pi_\theta}(s_0)$  is the value function for  $\pi_\theta$ , and  $r(s) = \mathbb{E}_{a \sim \pi}[\mathcal{R}(s, a)]$  is the expected reward at  $s$  following  $\pi$ . The occupancy measure  $\rho^{\pi_\theta}(s | s_0) = \frac{1}{T} \sum_{t=0}^T \mathbb{P}(s_t = s | s_0, \pi_\theta)$ , where  $T$  is a random variable denoting the index of the terminal step.



# Policy Gradient Objective

For the continuing case, where the process continues infinitely, we define the objective function  $J(\theta)$  as the averaged reward over the time steps.

$$\begin{aligned} J(\theta) &= \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \mathbb{E}[r_t \mid s_0, \pi_\theta] \\ &= \lim_{t \rightarrow \infty} \mathbb{E}[r_t \mid s_0, \pi_\theta] \\ &= \sum_s \rho^{\pi_\theta}(s \mid s_0) r(s) \\ &= V^{\pi_\theta}(s_0), \end{aligned}$$

where the occupancy measure  $\rho^{\pi_\theta}(s \mid s_0) = \lim_{t \rightarrow \infty} \mathbb{P}(s_t = s \mid s_0, \pi_\theta)$  is the stationary distribution of the Markov chain under policy  $\pi_\theta$ .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Policy Gradient Objective

**For the discounted case** where  $\gamma < 1$ , we define the objective function  $J(\theta)$  as the expected discounted return

$$J(\theta) = V^{\pi_\theta}(s_0) = \sum_{s \in \mathcal{S}} \rho^{\pi_\theta}(s | s_0) r(s),$$

where the occupancy measure  $\rho^{\pi_\theta}(s | s_0) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s | s_0, \pi_\theta)$ .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Policy Gradient Objective

The **policy gradient theorem** states that

$$\begin{aligned}\nabla_{\theta} J(\theta) &\propto \sum_{s \in \mathcal{S}} \rho^{\pi_{\theta}}(s | s_0) \sum_{a \in \mathcal{A}} Q^{\pi_{\theta}}(s, a) \nabla_{\theta} \pi_{\theta}(a | s) \\ &= \mathbb{E}_{\pi} [Q^{\pi_{\theta}}(s, a) \nabla_{\theta} \log \pi_{\theta}(a | s)].\end{aligned}$$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Proof of policy gradient theorem

Please refer to the proof of the episodic case in discrete state-action space in the lecture notes.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# REINFORCE (episodic Monte-Carlo policy-gradient control)

To compute the gradient  $\nabla_{\theta} \mathbf{J}(\theta)$  algorithmically, we can **sample  $N$  trajectories** following the policy  $\pi$  and **use the empirical mean** to estimate the gradient

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

- For  $Q^{\pi}(s, a)$ , we can use return  $G_t = \sum \gamma^t r_t$  to estimate.
- For  $\nabla_{\theta} \log \pi_{\theta}(a | s)$ , it depends on the form of the policy.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# REINFORCE (episodic Monte-Carlo policy-gradient control)

---

**Algorithm 1:** REINFORCE (Monte-Carlo method)

---

Initialize the policy parameter  $\theta$

**for each episode do**

    Sample one trajectory on policy  $\pi_\theta$ :  $s_0, a_0, r_0, s_1, a_1, \dots, s_T$

**for each**  $t = 0, 1, \dots, T$  **do**

$G_t \leftarrow \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$

$\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t | s_t)$

---



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

## REINFORCE with baselines

One problem of policy gradient method is **high variance**. (why? [Click to see a very intuitive explanation.](#)) A natural solution is to subtract a baseline  $b(s)$  from  $Q^\pi$ , i.e.,

$$\nabla_{\theta} J(\theta) \propto \sum_{s \in \mathcal{S}} \rho^{\pi}(s | s_0) \sum_{a \in \mathcal{A}} (Q^{\pi}(s, a) - b(s)) \nabla \pi_{\theta}(a | s).$$

The baseline can be any function, even a random variable, as long as it does not depend on the action  $a$ .

$$\sum_a b(s) \nabla \pi(a | s, \theta) = b(s) \nabla \sum_a \pi(a | s, \theta) = b(s) \nabla 1 = 0.$$

The expectation value does not change. The update rule that we end up with is a new version of REINFORCE that includes a general baseline



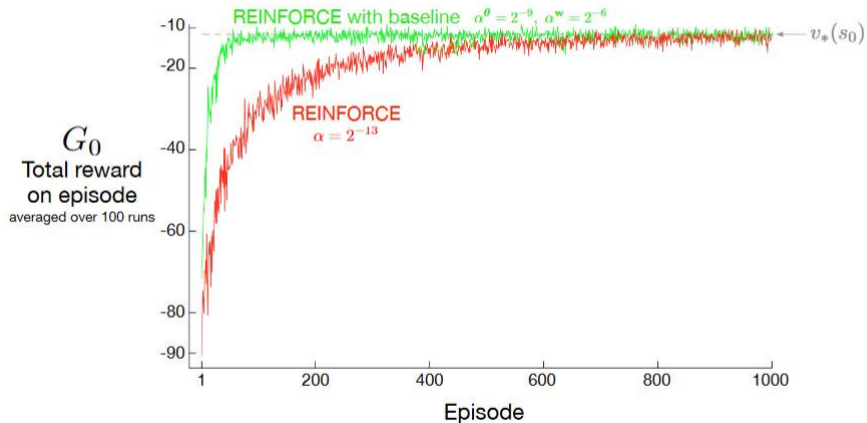
香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

$$\theta \leftarrow \theta + \alpha \gamma^t (G_t - b(s_t)) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t).$$



# REINFORCE with baselines



# REINFORCE with baselines

One natural choice for the baseline is an estimate of the state value  $\hat{V}(s, \mathbf{w})$ , where  $\mathbf{w} \in \mathbb{R}^d$  is a weight vector to be learned. We can use the same method as we adopted in learning  $\theta$  to learn  $\mathbf{w}$ . The complete process is as follows. We have two inputs:

- A differentiable policy parametrization  $\pi_{\theta}(a | s)$ ;
- A differentiable state value function parametrization  $\hat{V}(s, \mathbf{w})$ .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# REINFORCE with baselines

---

## Algorithm 2: REINFORCE with baseline

---

Initialize the policy parameter  $\theta$  and  $w$  at random.

**for each episode do**

    Sample one trajectory under policy  $\pi_\theta$ :  $s_0, a_0, r_0, s_1, a_1, r_1 \dots, s_T$

**for each**  $t = 1, 2, \dots, T$  **do**

$G_t \leftarrow \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$

$\delta \leftarrow G_t - \hat{V}(s_t, w)$

$w \leftarrow w + \alpha_w \delta \nabla_w \hat{V}(s_t, w)$

$\theta \leftarrow \theta + \alpha_\theta \gamma^t \delta \nabla_\theta \log \pi_\theta(a_t | s_t)$

---



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

# Question and Answering (Q&A)



香港中文大學(深圳)  
The Chinese University of Hong Kong, Shenzhen