Reinforcement Learning for Economists

Lecture 4

Policy Gradient Methods

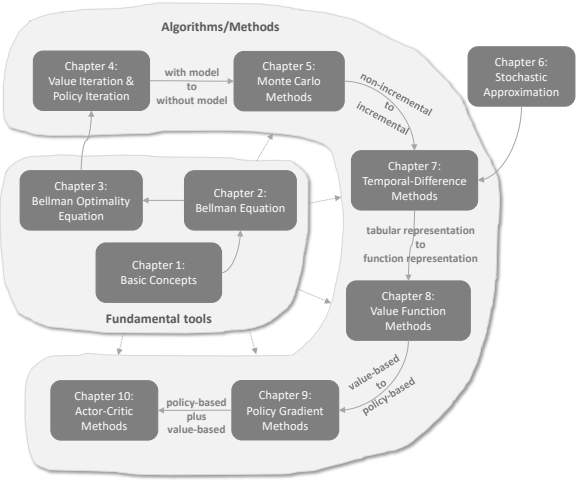
Benjamin Moll

London School of Economics

Plan for the four RL lectures

1. The basic idea of RL
2. Dynamic programming revisited
   - Value function iteration
   - Howard policy iteration
3. Monte Carlo methods
4. Stochastic approximation
5. Temporal difference learning, Sarsa, Q-learning
6. Policy gradient and actor-critic methods

Outline of Shiyu Zhao's book



Note: my slides are mostly a shortened/adapted version of Zhao's great slides

https://github.com/MathFoundationRL/Book-Mathematical-Foundation-of-Reinforcement-Learning

Policy gradient methods most popular in practice

For example, used for LLM post-training by large AI labs

- Reinforcement Learning from Human Feedback (RLHF)
- Reinforcement Learning with Verifiable Rewards (RLVR)
- Reasoning models
- See e.g. Andrej Karpathy's "Deep Dive into LLMs"
  https://www.youtube.com/watch?v=7xTGNNLPyMI

Example of standard algorithm = Proximal Policy Optimization (PPO)

- https://arxiv.org/abs/1707.06347
- https://openai.com/index/openai-baselines-ppo/

Policy gradient methods: basic idea

Recall: value of policy π

$$v_\pi(s) = \mathbb{E}_0\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s\right], \quad s_{t+1} \sim p(\cdot|s_t, a_t), \quad a_t = \pi(s_t)$$

Note: typically allow for stochastic policy $\pi(a|s)$ $\Rightarrow$ replace last part by

$$a_t \sim \pi(\cdot|s_t)$$

Idea of policy gradient methods: brute force maximize v_π with respect to π

Do this by some gradient ascent method ("ascent" not "descent" because max)

- specifically, stochastic approx. counterpart = stochastic gradient ascent
- roughly: incrementally increase likelihood of taking high-payoff actions
- this is about step 2 = policy improvement of Howard policy iteration
- step 1 = policy evaluation still done by estimating action-value fn $q(s, a)$

Good resources on policy gradient methods

Shiyu Zhao's book not quite as brilliant as preceding chapters

- reflected in my notes

Other recommended resources (already mentioned last lecture)

- Murphy (2025) "Reinforcement Learning: An Overview", less accessible
- Lilian Weng (2018) on policy gradient methods

Basic idea of policy gradient

Previously, policies have been represented by tables:

- The action probabilities of all states are stored in a table $\pi(a|s)$. Each entry of the table is indexed by a state and an action.

	a_1	a_2	a_3	a_4	a_5
s_1	$\pi(a_1\|s_1)$	$\pi(a_2\|s_1)$	$\pi(a_3\|s_1)$	$\pi(a_4\|s_1)$	$\pi(a_5\|s_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_9	$\pi(a_1\|s_9)$	$\pi(a_2\|s_9)$	$\pi(a_3\|s_9)$	$\pi(a_4\|s_9)$	$\pi(a_5\|s_9)$

Basic idea of policy gradient

Now, policies can be represented by parameterized functions:

$$\pi(a|s,\theta)$$

where $\theta \in \mathbb{R}^m$ is a parameter vector.

- The function can be, for example, a neural network, whose input is s, output is the probability to take each action, and parameter is θ.
- **Advantage:** when the state space is large, the tabular representation will be of low efficiency in terms of storage and generalization.
- The function representation is also sometimes written as $\pi(a, s, \theta)$, $\pi(a|s)$, or $\pi(a, s)$.

Policy gradient methods: basic idea – most important slide

Goal: parameterize policy $a = \pi(s, \theta)$ (deterministic) or $\pi(a|s, \theta)$ (stochastic) with $\theta \in \mathbb{R}^m$ and choose θ to maximize

$$v_\pi(s) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s \right], \quad s_{t+1} \sim p(\cdot|s_t, a_t), \; a_t = \pi(s_t, \theta)$$

Basic idea is simple:

1. Define some objective function $J(\theta)$ to maximize
2. Gradient ascent to search for optimal policies

$$\theta_{j+1} = \theta_j + \alpha \nabla_\theta J(\theta_j)$$

More precisely, implement stochastic counterpart (SGA rather than GA):

$$\nabla_\theta J(\theta) = \mathbb{E}[\text{stuff}(\theta)] \quad \text{motivates} \quad \theta_{j+1} = \theta_j + \alpha \times \text{stuff}(\theta_j)$$

Conversion to scalar objective $J(\theta)$

Goal: parameterize policy $a = \pi(s, \theta)$ (deterministic) or $\pi(a|s, \theta)$ (stochastic) with $\theta \in \mathbb{R}^m$ and choose θ to maximize

$$v_\pi(s) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s \right], \quad s_{t+1} \sim p(\cdot|s_t, a_t), \ a_t = \pi(s_t, \theta)$$

Problem 1: $v_\pi(s)$ is defined for all $s \in \mathcal{S}$, i.e. v_π = vector / function

- gradient-ascent-type algorithms cannot maximize vector / function

Solution: convert to scalar

$$J(\theta) = \sum_{s \in \mathcal{S}} v_\pi(s) \mu(s), \quad \mu(s) = \text{some distribution, e.g. uniform}$$

The “Policy Gradient Theorem”

Goal: parameterize policy $a = \pi(s, \theta)$ (deterministic) or $\pi(a|s, \theta)$ (stochastic) with $\theta \in \mathbb{R}^m$ and choose θ to maximize

$$v_\pi(s) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s \right], \quad s_{t+1} \sim p(\cdot|s_t, a_t), \; a_t = \pi(s_t, \theta)$$

Problem 2: written as is, differentiating w.r.t. θ (chain rule)

$$\nabla_\theta v_\pi(s)$$

requires knowledge of “model”, specifically of gradients

$$\nabla_a r(s, a) \quad \text{and} \quad \nabla_a p(s'|s, a)$$

but both transition probabilities $p(\cdot|s, a)$ and return fn $r(s, a)$ are unknown

What to do? How compute gradient $\nabla_\theta v_\pi(s)$ in “model-free” fashion?

Answer: Policy Gradient Theorem of Sutton et al. (2000) “Policy gradient methods for reinforcement learning with function approximation”

The “Policy Gradient Theorem”

Here: show two versions of policy gradient theorem

1. deterministic policy gradient theorem

2. stochastic policy gradient theorem

1. More pedagogical, 2. version normally implemented in practice (easier), normally just called “policy gradient theorem”

Deterministic policy gradient theorem (Silver et al., 2014)

Theorem: The gradient of the objective

$$J(\theta) = \sum_{s \in \mathcal{S}} v_\pi(s)\mu(s) \quad \text{with} \quad v_\pi(s) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s \right]$$

can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t \nabla_a q_\pi(s_t, a_t) \nabla_\theta \pi(s_t, \theta) \right], \; s_{t+1} \sim p(\cdot|s_t, a_t), \; a_t = \pi(s_t, \theta), \; s_0 \sim \mu(\cdot)$$

Proof: next slides. How is theorem useful? How implement in practice?

Recall: $\nabla_\theta J(\theta) = \mathbb{E}[\text{stuff}(\theta)]$ motivates $\theta_{j+1} = \theta_j + \alpha \times \text{stuff}(\theta_j)$

$$\Rightarrow \quad \theta_{j+1} = \theta_j + \alpha \sum_{t=0}^{T} \gamma^t \nabla_a q_\pi(s_t, a_t) \nabla_\theta \pi(s_t, \theta_j)$$

Need estimate of Q function $q_\pi(s, a)$ and its derivative $\nabla_a q_\pi(s, a)$

- will return to this point after covering stochastic version

Deterministic policy gradient theorem: proof

Goal: parameterize policy $\pi(s, \theta)$ with $\theta \in \mathbb{R}^m$ and choose θ to maximize

$$v_\pi(s) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s \right], \quad s_{t+1} \sim p(\cdot|s_t, a_t), \ a_t = \pi(s_t, \theta)$$

Work with Bellman equation for v_π

$$v_\pi(s) = r(s, \pi(s, \theta)) + \gamma \sum_{s' \in \mathcal{S}} v_\pi(s') p(s'|s, \pi(s, \theta))$$

Differentiate with respect to θ:

$$\nabla_\theta v_\pi(s) = \left(\nabla_a r(s, \pi) + \gamma \sum_{s' \in \mathcal{S}} v_\pi(s') \nabla_a p(s'|s, \pi) \right) \nabla_\theta \pi + \gamma \sum_{s' \in \mathcal{S}} \left(\nabla_\theta v_\pi(s') \right) p(s'|s, \pi)$$

Problem: presence of $\nabla_a r$ and $\nabla_a p$, both of which are unknown

What to do? Same as earlier: Q function to the rescue!

Deterministic policy gradient theorem: proof

Recall

$$\nabla_\theta v_\pi(s) = \left(\nabla_a r(s,\pi) + \gamma \sum_{s' \in \mathcal{S}} v_\pi(s') \nabla_a p(s'|s,\pi) \right) \nabla_\theta \pi + \gamma \sum_{s' \in \mathcal{S}} \left(\nabla_\theta v_\pi(s') \right) p(s'|s,\pi) \quad (*)$$

As before, can write action-value or Q function

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} v_\pi(s') p(s'|s,a)$$

with derivative

$$\nabla_a q_\pi(s,a) = \nabla_a r(s,a) + \gamma \sum_{s' \in \mathcal{S}} v_\pi(s') \nabla_a p(s'|s,a)$$

Substitute into $(*)$

$$\nabla_\theta v_\pi(s) = \nabla_a q_\pi(s, \pi(s,\theta)) \nabla_\theta \pi + \gamma \sum_{s' \in \mathcal{S}} \left(\nabla_\theta v_\pi(s') \right) p(s'|s, \pi(s,\theta))$$

This is a Bellman equation for $\nabla_\theta v_\pi(s) \Rightarrow$ can write it as PDV!

Deterministic policy gradient theorem: proof

Recall Bellman equation for $\nabla_\theta v_\pi(s)$

$$\nabla_\theta v_\pi(s) = \nabla_a q_\pi(s, \pi(s,\theta))\nabla_\theta \pi + \gamma \sum_{s' \in \mathcal{S}} \left(\nabla_\theta v_\pi(s')\right) p(s'|s, \pi(s,\theta))$$

Use this Bellman equation to write $\nabla_\theta v_\pi(s)$ in PDV form:

$$\nabla_\theta v_\pi(s_0) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t \nabla_a q_\pi(s_t, a_t) \nabla_\theta \pi(s_t, \theta) \right], \; s_{t+1} \sim p(\cdot|s_t, a_t), \; a_t = \pi(s_t, \theta)$$

and hence recalling $J(\theta) = \sum_{s \in \mathcal{S}} v_\pi(s)\mu(s)$ also

$$\nabla_\theta J(\theta) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t \nabla_a q_\pi(s_t, a_t) \nabla_\theta \pi(s_t, \theta) \right], \; s_{t+1} \sim p(\cdot|s_t, a_t), \; a_t = \pi(s_t, \theta), \; s_0 \sim \mu(\cdot)$$

Policy gradient theorem with stochastic $a_t \sim \pi(\cdot|s_t, \theta)$

Theorem: The gradient of the objective

$$J(\theta) = \sum_{s \in \mathcal{S}} v_\pi(s)\mu(s) \quad \text{with} \quad v_\pi(s) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s \right]$$

can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t q_\pi(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t, \theta) \right], \; s_{t+1} \sim p(\cdot|s_t, a_t), \; a_t \sim \pi(\cdot|s_t, \theta), \; s_0 \sim \mu(\cdot)$$

Rel to det. version: $\nabla_\theta \ln \pi(a_t|s_t, \theta)$, no gradient $\nabla_a q_\pi(s, a)$. **Proof:** in 2 slides.

How implement? $\nabla_\theta J(\theta) = \mathbb{E}[\text{stuff}(\theta)]$ motivates $\theta_{j+1} = \theta_j + \alpha \times \text{stuff}(\theta_j)$

$$\Rightarrow \quad \theta_{j+1} = \theta_j + \alpha \sum_{t=0}^{T} \gamma^t q_\pi(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t, \theta_j)$$

Note: need estimate of Q function $q_\pi(s, a)$ (but not derivative $\nabla_a q_\pi(s, a)$)

Policy gradient theorem: proof

Goal: parameterize policy $\pi(a|s,\theta)$ with $\theta \in \mathbb{R}^m$ and choose θ to maximize

$$v_\pi(s) = \mathbb{E}_0\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s\right], \quad s_{t+1} \sim p(\cdot|s_t, a_t), \quad a_t \sim \pi(\cdot|s_t, \theta)$$

Work with Bellman equation for v_π:

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s,\theta)\left(r(s,a) + \gamma \sum_{s'\in\mathcal{S}} v_\pi(s') p(s'|s,a)\right)$$

Differentiate with respect to θ:

$$\nabla_\theta v_\pi(s) = \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s,\theta)\left(r(s,a) + \gamma \sum_{s'\in\mathcal{S}} v_\pi(s') p(s'|s,a)\right) + \sum_{a\in\mathcal{A}} \pi(a|s,\theta)\gamma \sum_{s'\in\mathcal{S}} \left(\nabla_\theta v_\pi(s')\right) p(s'|s,a)$$

Use definition of Q function

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s'\in\mathcal{S}} v_\pi(s') p(s'|s,a)$$

Policy gradient theorem: proof

$$\Rightarrow \quad \nabla_\theta v_\pi(s) = \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s,\theta) q_\pi(s,a) + \sum_{a\in\mathcal{A}} \pi(a|s,\theta)\gamma \sum_{s'\in\mathcal{S}} \left(\nabla_\theta v_\pi(s')\right) p(s'|s,a)$$

Not quite Bellman equation for $\nabla_\theta v_\pi(s)$ because 1st term features $\nabla_\theta \pi(a|s,\theta)$ rather than $\pi(a|s,\theta)$. But can convert into one by massaging first term:

$$\nabla_\theta \pi(a|s,\theta) = \pi(a|s,\theta)\frac{\nabla_\theta \pi(a|s,\theta)}{\pi(a|s,\theta)} = \pi(a|s,\theta)\nabla_\theta \ln \pi(a|s,\theta)$$

$$\Rightarrow \quad \nabla_\theta v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s,\theta) \left(\nabla_\theta \ln \pi(a|s,\theta) q_\pi(s,a) + \gamma \sum_{s'\in\mathcal{S}} \left(\nabla_\theta v_\pi(s')\right) p(s'|s,a) \right)$$

Now it's a Bellman equation for $\nabla_\theta v_\pi(s) \Rightarrow$ can write it as PDV:

$$\nabla_\theta v_\pi(s_0) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t q_\pi(s_t,a_t) \nabla_\theta \ln \pi(a_t|s_t,\theta) \right], \; s_{t+1} \sim p(\cdot|s_t,a_t), \; a_t \sim \pi(\cdot|s_t,\theta)$$

Expression for $J(\theta) = \sum_{s\in\mathcal{S}} v_\pi(s)\mu(s)$ follows immediately.

REINFORCE algorithm

Recall policy gradient theorem: $\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t q_\pi(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t, \theta)\right]$

$$\Rightarrow \quad \theta_{j+1} = \theta_j + \alpha \sum_{t=0}^{T} \gamma^t q_\pi(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t, \theta_j)$$

Need estimate of Q function $q_\pi(s, a)$. One option: Monte Carlo

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \middle| s_t = s, a_t = a \right], \quad G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$$

so that $\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t G_t \nabla_\theta \ln \pi(a_t|s_t, \theta)\right]$

$$\Rightarrow \quad \theta_{j+1} = \theta_j + \alpha \sum_{t=0}^{T} \gamma^t G_t \nabla_\theta \ln \pi(a_t|s_t, \theta_j), \quad G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$$

If q_π is estimated by Monte Carlo, algorithm has specific name: REINFORCE

Note that REINFORCE is "episodic": need to wait until T to update θ

REINFORCE pseudocode and a clarification

REINFORCE updates parameter vector θ in "episodic" fashion according to

$$\theta_{j+1} = \theta_j + \alpha \sum_{t=0}^{T} \gamma^t G_t \nabla_\theta \ln \pi(a_t|s_t, \theta_j), \quad G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$$

All of Zhao, Sutton-Barto & Murphy write this as (here from Sutton-Barto)

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\quad (G_t)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Last 3 lines make it look like policy $\pi(a|s, \theta)$ is updated at each time step, θ_t to θ_{t+1}
But it is not. This is just an efficient way of calculating $\sum_{t=0}^{T}$ between updates j:

$$\theta_j^{t+1} = \theta_j^t + \alpha \gamma^t G_t \nabla_\theta \ln \pi(a_t|s_t, \theta_j) \quad \Rightarrow \quad \theta_j^T = \theta_j^0 + \alpha \sum_{t=0}^{T} \gamma^t G_t \nabla_\theta \ln \pi(a_t|s_t, \theta_j)$$

Policy gradient theorem: further simplification

Can further simplify this: write as simple expectation rather than expected PDV

Roughly $\nabla_\theta J(\theta) \propto \mathbb{E}\left[q(s,a)\nabla_\theta \ln \pi(a|s,\theta)\right]$ for some $\mathbb{E}$

To this end, define "discounted state visitation distribution under policy π":

$$d_{\pi,\mu}(s) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr(s_t = s, s_0 \sim \mu, \pi) \qquad (*)$$

Note: $d_{\pi,\mu}(s)$ is a probability distribution, in particular

$$\sum_{s\in\mathcal{S}} d_{\pi,\mu}(s) = 1$$

Proof: write $(*)$ more precisely using vector notation and transition matrix $\mathbf{P}_\pi$

$$\mathbf{d}_{\pi,\mu} = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t(\mathbf{P}_\pi^\top)^t\boldsymbol{\mu} \quad \Rightarrow \quad \mathbf{d}_{\pi,\mu}^\top\mathbf{1} = 1$$

Policy gradient theorem: summary

Theorem: The gradient of the objective

$$J(\theta) = \sum_{s \in \mathcal{S}} v_\pi(s)\mu(s) \quad \text{with} \quad v_\pi(s) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s \right]$$

can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \gamma^t q_\pi(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t, \theta) \right], \; s_{t+1} \sim p(\cdot|s_t, a_t), \; a_t \sim \pi(\cdot|s_t, \theta), \; s_0 \sim \mu(\cdot)$$

or equivalently

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{\pi,\mu}(\cdot), a \sim \pi(\cdot|s,\theta)} \left[q_\pi(s, a) \nabla_\theta \ln \pi(a|s, \theta) \right] \quad \text{where}$$

$d_{\pi,\mu}(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s, s_0 \sim \mu, \pi) =$ disc. state visitation dist'n.

Momentarily: motivates a simpler, incremental stochastic gradient ascent algorithm

$$\theta_{t+1} = \theta_t + \alpha q_\pi(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t, \theta_t)$$

Actor-Critic Methods

Introduction

Actor-critic methods are still policy gradient methods

- They emphasize the structure that incorporates the policy gradient and value-based methods.

What are “actor” and “critic”?

- Here, “actor” refers to policy update. It is called *actor* because the policies will be applied to take actions.
- Here, “critic” refers to policy evaluation or value estimation. It is called *critic* because it criticizes the policy by evaluating it.

The simplest actor-critic

How to get $q_t(s_t, a_t)$?

So far, we have studied two ways to estimate action values:

- **Monte Carlo learning:** If MC is used, the corresponding algorithm is called REINFORCE or Monte Carlo policy gradient.
  - We introduced earlier in this lecture.
- **Temporal-difference learning:** If TD is used, such kind of algorithms are usually called actor-critic.
  - We will introduce now.

In contrast to REINFORCE, actor-critic is incremental

Recall REINFORCE updates

$$\theta_{j+1} = \theta_j + \alpha \sum_{t=0}^{T} \gamma^t G_t \nabla_\theta \ln \pi(a_t|s_t, \theta_j), \quad G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$$

"Episodic": have to wait until end of episode ($t = 0, \ldots, T$) to update θ

Actor-critic methods instead update in incremental fashion at each t:

$$\theta_{t+1} = \theta_t + \alpha q_\pi(s_t, a_t) \nabla_\theta \ln \pi(a_t|s_t, \theta_t)$$

From Murphy (2025):

3.2 Actor-critic methods

An **actor-critic** method [BSA83] uses the policy gradient method, but where the expected return G_t is estimated using temporal difference learning of a value function instead of MC rollouts. (The term "actor" refers to the policy, and the term "critic" refers to the value function.) The use of bootstrapping in TD updates allows more efficient learning of the value function compared to MC, and further reduces the variance. In addition, it allows us to develop a fully online, incremental algorithm, that does not need to wait until the end of the trajectory before updating the parameters.

The simplest actor-critic

Revisit the idea of policy gradient introduced earlier in this lecture

1. A scalar metric $J(\theta) = \sum_{s \in \mathcal{S}} v_\pi(s)\mu(s)$ as before.
2. The gradient-ascent algorithm maximizing $J(\theta)$ is

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha \nabla_\theta J(\theta_t) \\ &= \theta_t + \alpha \mathbb{E}_{S \sim d_{\pi,\mu}, A \sim \pi} \left[\nabla_\theta \ln \pi(A|S, \theta_t)\, q_\pi(S, A) \right] \end{aligned}$$

3. The stochastic gradient-ascent algorithm is

$$\boxed{\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi(a_t|s_t, \theta_t)\, q_t(s_t, a_t)}$$

This expression is very important! We can directly see "actor" and "critic" from it:

- This expression corresponds to actor!
- The algorithm estimating $q_t(s_t, a_t)$ corresponds to critic!

The simplest actor-critic

The simplest actor-critic algorithm (QAC)

Initialization: A policy function $\pi(a|s, \theta_0)$ where θ_0 is the initial parameter. A value function $q(s, a, w_0)$ where w_0 is the initial parameter. $\alpha_w, \alpha_\theta > 0$.
Goal: Learn an optimal policy to maximize $J(\theta)$.

At time step t in each episode, do

Generate a_t following $\pi(a|s_t, \theta_t)$, observe r_{t+1}, s_{t+1}, and then generate a_{t+1} following $\pi(a|s_{t+1}, \theta_t)$.

Actor (policy update):

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \ln \pi(a_t|s_t, \theta_t)\, q(s_t, a_t, w_t)$$

Critic (value update):

$$w_{t+1} = w_t + \alpha_w \big[r_{t+1} + \gamma\, q(s_{t+1}, a_{t+1}, w_t) - q(s_t, a_t, w_t)\big] \nabla_w q(s_t, a_t, w_t)$$

The simplest actor-critic

Remarks:

- The *critic* corresponds to “Sarsa + value function approximation”
- The *actor* corresponds to the policy update algorithm
- This particular actor-critic algorithm is sometimes referred to as Q Actor-Critic (QAC)
- Though simple, this algorithm reveals the core idea of actor-critic methods
- It can be extended to many other algorithms – see Zhao, Sutton-Barto, ...