Basic fundalmentals of reinforcement learning

Nguyen Nhat Minh,
HUST
CMC AI Intership
Data Science and Artificial Intelligence

February 15, 2026

Outline

1. Part I : Introduction
   - Markov Decision
   - Bellman Equation
   - Value-based method
   - Policy Gradient

Part I: Fundalmentals of Reinforcement Learning

Outline

1. Part I : Introduction
   - Markov Decision
   - Bellman Equation
   - Value-based method
   - Policy Gradient

Markov Decision Process

In more formal terms, almost all the RL problems can be framed as Markov Decision Processes (MDPs). All states in MDP has "Markov" property, referring to the fact that the future only depends on the current state, not the history:

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

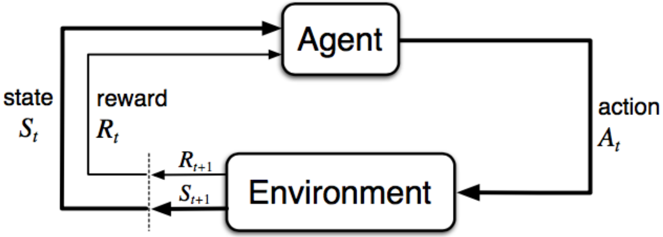


Figure: Markov Decision Process

Problem Formulation

Given Decision Markov Process(MDP) :

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, p_0),$$

- $\mathcal{A}$: action space
- $\mathcal{S}$: state space
- $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathbb{R})$ is the environment's transition function. $P(s', r \mid s, a)$ of transitioning to a new environment state s' and receiving reward $r \in \mathbb{R}$
- $\gamma \in [0, 1]$ is a discount rate
- $p_0 \in \Delta(\mathcal{S})$ is a probability distribution over potential starting states.

We call sequences of states, actions, and rewards as trajectories:

$$(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, \ldots, s_{t+k-1}, a_{t+k-1}, r_{t+k}, s_{t+k})$$

Accordingly, we can also compute the alternative transition probabilities as

$$P(s' \mid s, a) = \int_{r \in \mathbb{R}} P(s', r \mid s, a)\, dr.$$

Problem Formulation

- The main goal is to solve the *control problem* of learning a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ to maximize the expected return:

$$G_t := \sum_{k=0}^{T} \gamma^k r_{t+k+1},$$

  G_t is bounded since rewards are bounded.

- RL algorithms $\mathcal{A} : \pi \to \pi$ for the control problem now iteratively learn policies by interacting with the environment using the current policy to sample transitions, which are then used to update the policy.

Introduction

Policy:

- The agent's behavior function π, tells us which action to take in state s. It is a mapping from state s to action a and can be either deterministic or stochastic:
  - **Deterministic:** $\pi(s) = a$.
  - **Stochastic:** $\pi(a \mid s) = \mathbb{P}_\pi[A = a \mid S = s]$.
- For a policy π, its stationary state distribution d^π determines the probability of being in a specific state $s \in \mathcal{S}$ at any point in time when following π.

Introduction

Value Function:

- Return value G_t starting from the time t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The **state-value** of a state s is the expected return if we are in this state at time t, $S_t = s$:

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

- Similarly, we define the **action-value** ("Q-value"; Q as "Quality") of a state-action pair as:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

- The difference between action-value and state-value is the action **advantage** function ("A-value"):

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$$

Introduction

- **Model-based:** Rely on the model of the environment; either the model is known or the algorithm learns it explicitly.
- **Model-free:** No dependency on the model during learning.
- **On-policy:** Use the deterministic outcomes or samples from the target policy to train the algorithm.
- **Off-policy:** Training on a distribution of transitions or episodes produced by a different behavior policy rather than that produced by the target policy.

Outline

1. Part I : Introduction
   - Markov Decision
   - Bellman Equation
   - Value-based method
   - Policy Gradient

Bellman equation

- Bellman equations refer to a set of equations that decompose the value function into the immediate reward plus the discounted future values.

$$\begin{aligned}
V(s) &= \mathbb{E}[G_t \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s]
\end{aligned}$$

- Similarly for Q-value,

$$\begin{aligned}
Q(s,a) &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a' \sim \pi} Q(S_{t+1}, a') \mid S_t = s, A_t = a]
\end{aligned}$$

Bellman Expectation Equations

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s, a) \tag{1}$$

$$Q_\pi(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V_\pi(s') \tag{2}$$

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V_\pi(s') \right) \tag{3}$$

$$Q_\pi(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) \sum_{a' \in \mathcal{A}} \pi(a'|s') Q_\pi(s', a') \tag{4}$$

- (1): Average over actions
- (2): Equals to rewards at (s,a) + discounted future return.
- (3): (1) + (2)
- (4): (1) + (2)

Bellman Optimalizty Equations

The optimal values V_* and Q_* are the best returns we can obtain, defined as:

$$V_*(s) = \max_{a \in \mathcal{A}} Q_*(s, a)$$

$$Q_*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V_*(s')$$

$$V_*(s) = \max_{a \in \mathcal{A}} \left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V_*(s') \right)$$

$$Q_*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) \max_{a' \in \mathcal{A}} Q_*(s', a')$$

In most scenarios, we do not know $P^a_{ss'}$ or $R(s, a)$, so we cannot solve MDPs by directly applying Bellman equations, but it lays the theoretical foundation for many RL algorithms.

Outline

1. Part I : Introduction
    - Markov Decision
    - Bellman Equation
    - Value-based method
    - Policy Gradient

Common Approaches in Tabular RL

- **Dynamic Programming**: When the model is fully known, following Bellman equations, we can use Dynamic Programming (DP) to iteratively evaluate value functions and improve policy.
- **Monte Carlo Method** : learns from episodes of raw experience without modeling the environmental dynamics and computes the observed mean return as an approximation of the expected return.(low bias, high variance)
- **Temporal Difference** : TD learning can learn from incomplete episodes and hence we don't need to track the episode up to termination.(high bias, low variance)

Dynamic Programming

- **Policy Evaluation** compute the state-value V_π for a given policy π:

$$\begin{aligned} V_{t+1}(s) &= \mathbb{E}_\pi[r + \gamma V_t(s') \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{s',r} P(s', r \mid s, a)\big(r + \gamma V_t(s')\big) \end{aligned}$$

- **Policy Improvement** Based on the value functions, Policy Improvement generates a better policy $\pi' \geq \pi$ by acting greedily: $\pi'(s) = \text{argmax}_{a\in\mathcal{A}} \text{Q}_\pi(s, a)$,where

$$Q_\pi(s, a) = \sum_{s',r} P(s', r \mid s, a)\big(r + \gamma V_\pi(s')\big)$$

- The value of π' is guranteed to improve since:

$$\begin{aligned} Q_\pi(s, \pi'(s)) &= Q_\pi\left(s, \arg\max_{a\in\mathcal{A}} Q_\pi(s, a)\right) \\ &= \max_{a\in\mathcal{A}} Q_\pi(s, a) \geq Q_\pi(s, \pi(s)) = V_\pi(s) \end{aligned}$$

Dynamic Programming

- **Policy Iteration**: The *Generalized Policy Iteration (GPI)* algorithm refers to an iterative procedure to improve the policy when combining policy evaluation and improvement:

$$\pi_0 \xrightarrow{\text{evaluation}} V_{\pi_0} \xrightarrow{\text{improve}} \pi_1 \xrightarrow{\text{evaluation}} V_{\pi_1} \cdots \xrightarrow{\text{improve}} \pi_* \xrightarrow{\text{evaluation}} V_*$$

Theorem (Generalized Policy Iteration)

Let π_{old} be the current policy. Then, GPI updates its policy by

$$\pi_{new} \in \arg\max_{\pi \in \Pi} \mathbb{E}_{A\sim\pi}\left[Q_{\pi_{old}}(s, A)\right]$$

for all $s \in \mathcal{S}$. Let $(\pi_n)_{n=0}^{\infty}$ be a sequence of policies obtained through GPI. Then, this sequence converges to an optimal policy, i.e.

$$\lim_{n\to\infty} \pi_n = \pi^*, \lim_{n\to\infty} Q_{\pi_n} = Q^*.$$

Monte Carlo Methods

- To compute the empirical return G_t, Monte Carlo (MC) methods need to learn from **complete episodes** $S_1, A_1, R_2, \ldots, S_T$ to compute:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

- The empirical mean return for state s is:

$$V(s) = \frac{\sum_{t=1}^{T} \mathbf{1}[S_t = s]\, G_t}{\sum_{t=1}^{T} \mathbf{1}[S_t = s]}$$

- This way of approximation can be easily extended to action-value functions by counting (s, a) pairs:

$$Q(s, a) = \frac{\sum_{t=1}^{T} \mathbf{1}[S_t = s, A_t = a]\, G_t}{\sum_{t=1}^{T} \mathbf{1}[S_t = s, A_t = a]}$$

Temporal Difference

Value Estimation

The key idea in TD learning is to update the value function $V(S_t)$ towards an estimated return $R_{t+1} + \gamma V(S_{t+1})$ (**TD target**). To what extent we want to update the value function is controlled by the learning rate hyperparameter α:

$$\begin{aligned} V(S_t) &\leftarrow (1-\alpha)V(S_t) + \alpha G_t \\ V(S_t) &\leftarrow V(S_t) + \alpha(G_t - V(S_t)) \\ V(S_t) &\leftarrow V(S_t) + \alpha \underbrace{\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)}_{\textbf{TD error}} \end{aligned}$$

Similarly, for action-value estimation:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\big)$$

n-Step TD

In order to control the **bias-variance trade-off**, we would like an estimator for the return with intermediate properties between MC and TD. This is what the **n-step return** offers:

$$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k+1} + \gamma^n V(s_{t+n})$$

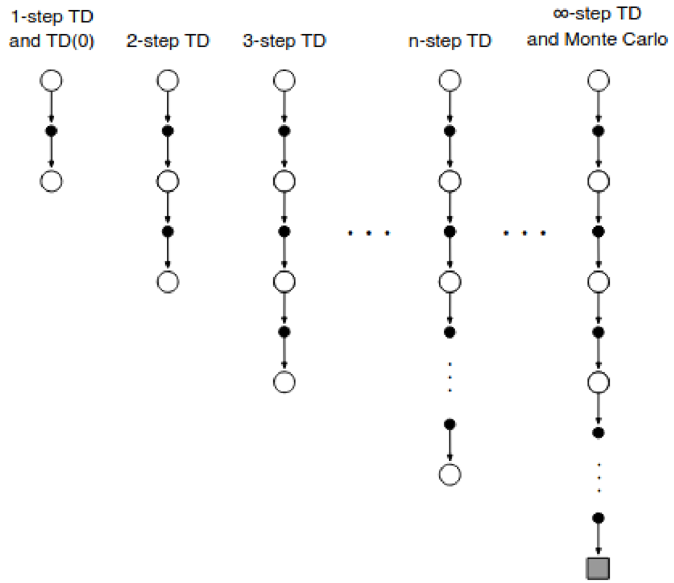
This is the target used by *n*-step TD:

$$V(s_t) \leftarrow V(s_t) + \alpha\big(G_t^{(n)} - V(s_t)\big).$$

Special cases:

- $n = 1$: $G_t^{(1)} = r_{t+1} + \gamma V(s_{t+1}) \rightarrow$ standard 1-step TD.
- $n \rightarrow \infty$ (episodic): $G_t^{(\infty)}$ becomes the Monte Carlo return (no bootstrapping).

n-Step TD


1-step TD
and TD(0)
2-step TD
3-step TD
n-step TD
∞-step TD
and Monte Carlo

Eligibility Traces

Picking n can be tough, and most certainly won't generalize to different environments
$\longrightarrow$ We'll do this by picking all different values of at once. The intuition is to average all of the possible n-step returns into a single return.

Forward view

TD(λ) forward view forms a λ-weighted mixture of all *n*-step returns. The λ-return at time t is

$$G_t^{(\lambda)} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}.$$

This is a convex combination (weights sum to 1 for infinite-horizon episodic case). The TD(λ) forward-view update is:

$$V(s_t) \leftarrow V(s_t) + \alpha\big(G_t^{(\lambda)} - V(s_t)\big).$$

Importantly, can express $G_t^{(\lambda)} - V(s_t)$ as weighted sum of future TD errors:

$$G_t^{(\lambda)} - V(s_t) = \sum_{k=0}^{\infty} (\gamma\lambda)^k \, \delta_{t+k},$$

where $\delta_{t+k} = r_{t+k+1} + \gamma V(s_{t+k+1}) - V(s_{t+k})$.

Proof: expand $G_t^{(n)}$, telescope the V terms:

$$
\begin{aligned}
G_t^{(n)} &= r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}) \\
&= (r_{t+1} + \gamma V(s_{t+1}) - \gamma V(s_{t+1})) + \gamma(r_{t+2} + \gamma V(s_{t+2}) - \gamma V(s_{t+2})) + \cdots \\
&\quad + \gamma^{n-1}(r_{t+n} + \gamma V(s_{t+n}) - \gamma V(s_{t+n})) + \gamma^n V(s_{t+n}) \\
&= V(s_t) + \sum_{k=0}^{n-1} \gamma^k \left(r_{t+k+1} + \gamma V(s_{t+k+1}) - V(s_{t+k})\right) = V(s_t) + \sum_{k=0}^{n-1} \gamma^k \delta_{t+k}.
\end{aligned}
$$

Now take the λ-weighted sum:

$$
\begin{aligned}
G_t^{(\lambda)} - V(s_t) &= (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \left(G_t^{(n)} - V(s_t)\right) \\
&= (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \sum_{k=0}^{n-1} \gamma^k \delta_{t+k}. \\
&= \sum_{k=0}^{\infty} (\gamma\lambda)^k \, \delta_{t+k} \quad (Q.E.D)
\end{aligned}
$$

Backward view

The backward view implements the λ-return *online* without waiting for many future rewards. For each state s we maintain an **eligibility trace** $e_t(s)$. The standard accumulating-trace definition (on stepping from $t-1$ to t):

$$e_t(s) = \gamma\lambda e_{t-1}(s) + \mathbf{1}\{s_t = s\},$$

where **1** is the indicator function (1 if current state equals s, else 0). At each time t, compute the TD error δ_t and update all states according to:

$$V(s) \leftarrow V(s) + \alpha\,\delta_t\,e_t(s) \qquad \text{for all } s.$$

Interpretation: δ_t is used to update not only the present state s_t but all past states weighted by how eligible they are (recent visits get larger eligibility).

Equivalence of Forward and Backward View

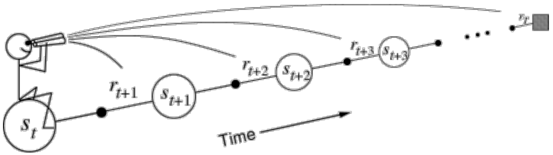


Figure: Forward View of TD(λ)

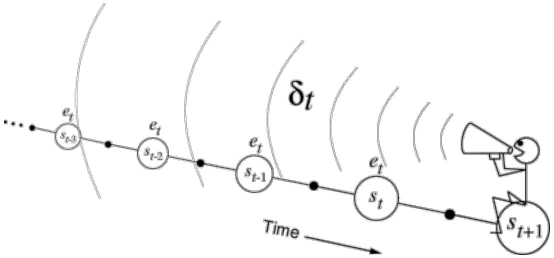


Figure: Backward View of TD(λ)

GAE

Given TD error: $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$. It is easy to show recursively that it depends on the TD error:

$$\begin{aligned}
\hat{A}_t^{(1)} &:= \delta_t^V = -V(s_t) + r_t + \gamma V(s_{t+1}) \\
\hat{A}_t^{(2)} &:= \delta_t^V + \gamma \delta_{t+1}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \\
\hat{A}_t^{(3)} &:= \delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3}) \\
&\cdots \\
\hat{A}_t^{(k)} &:= \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})
\end{aligned}$$

GAE

The generalized advantage estimator GAE(γ, λ) is defined as the exponentially-weighted average of these k-step estimators:

$$
\begin{aligned}
&\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} \\
&:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots\right) \\
&= (1-\lambda)\left(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots\right) \\
&= (1-\lambda)\left(\delta_t^V(1 + \lambda + \lambda^2 + \dots) + \gamma\delta_{t+1}^V(\lambda + \lambda^2 + \lambda^3 + \dots) + \right. \\
&\gamma^2\delta_{t+2}^V(\lambda^2 + \lambda^3 + \lambda^4 + \dots) + \dots \\
&= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + \dots\right) \\
&= \sum_{l=0}^{\infty}(\gamma\lambda)^l \delta_{t+l}^V
\end{aligned}
$$

GAE

There are two notable special cases of this formula, obtained by setting $\lambda = 0$ and $\lambda = 1$.

$$\text{GAE}(\gamma, 0): \quad \hat{A}_t := \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{5}$$

$$\text{GAE}(\gamma, 1): \quad \hat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) \tag{6}$$

$\text{GAE}(\gamma, 1)$ is γ-just regardless of the accuracy of V, but it has high variance due to the sum of terms. $\text{GAE}(\gamma, 0)$ is γ-just for $V = V^{\pi,\gamma}$ and otherwise induces bias, but it typically has much lower variance. The generalized advantage estimator for $0 < \lambda < 1$ makes a compromise between bias and variance, controlled by parameter λ.

Outline

1 Part I : Introduction
- Markov Decision
- Bellman Equation
- Value-based method
- Policy Gradient

Policy Gradient

The policy gradient methods target at modeling and optimizing the policy directly.

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

Then, using gradient ascent, we can update by formula:

$$\theta \leftarrow \theta + \nabla J(\theta)$$

Theorem (Policy Gradient Theorem)

For a given MDP, let π_θ be differentiable w.r.t. θ and $\nabla_\theta \pi_\theta$ be bounded, let Q_{π_θ} be differentiable w.r.t. θ and $\nabla_\theta Q_{\pi_\theta}$ be bounded for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Then, there exists a constant η such that

$$\nabla_\theta J(\theta) = \eta \, \mathbb{E}_{S \sim d^{\pi_\theta}, A \sim \pi_\theta} \left[Q_{\pi_\theta}(S, A) \, \nabla_\theta \ln \pi_\theta(A \mid S) \right].$$

Policy Gradient

Proof

$$\nabla_\theta V^\pi(s) = \nabla_\theta \left(\sum_{a\in\mathcal{A}} \pi_\theta(a|s) Q^\pi(s,a) \right)$$

$$= \sum_{a\in\mathcal{A}} \left(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \nabla_\theta Q^\pi(s,a) \right)$$

$$= \sum_{a\in\mathcal{A}} \left(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \nabla_\theta \sum_{s',r} P(s',r|s,a)(r + V^\pi(s')) \right)$$

$$= \sum_{a\in\mathcal{A}} \left(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \sum_{s',r} P(s',r|s,a) \nabla_\theta V^\pi(s') \right)$$

$$= \sum_{a\in\mathcal{A}} \left(\nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V^\pi(s') \right)$$

Policy Gradient

$$\nabla_\theta V^\pi(s) = \phi(s) + \sum_a \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V^\pi(s')$$

$$= \phi(s) + \sum_{s'} \sum_a \pi_\theta(a|s) P(s'|s,a) \nabla_\theta V^\pi(s')$$

$$= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \nabla_\theta V^\pi(s')$$

$$= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \sum_{a \in \mathcal{A}} \Big(\nabla_\theta \pi_\theta(a|s') Q^\pi(s', a) +$$

$$\pi_\theta(a|s') \sum_{s''} P(s''|s', a) \nabla_\theta V^\pi(s'') \Big)$$

$$= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \Big[\phi(s') + \sum_{s''} \rho^\pi(s' \to s'', 1) \nabla_\theta V^\pi(s'') \Big]$$

Policy Gradient

$$= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1)\phi(s') + \sum_{s'} \rho^\pi(s \to s', 1)$$

$$\sum_{s''} \rho^\pi(s' \to s'', 2)\nabla_\theta V^\pi(s'')$$

$$= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1)\phi(s') + \sum_{s''} \rho^\pi(s \to s'', 2)\phi(s'') +$$

$$\sum_{s'''} \rho^\pi(s \to s''', 3)\nabla_\theta V^\pi(s''')$$

$$= \cdots \quad \text{(Repeatedly unrolling the part of } \nabla_\theta V^\pi(\cdot)\text{)}$$

$$= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \rho^\pi(s \to x, k)\phi(x)$$

$$= \sum_{s} \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k)\phi(s) \quad ; \text{Let } \eta(s) = \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k)$$

Policy Gradient

$$= \sum_s \eta(s)\phi(s) = \left(\sum_s \eta(s)\right) \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \phi(s)$$

$$\propto \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \phi(s), \quad \sum_s \eta(s) \text{ is a constant}$$

$$= \sum_s d^\pi(s) \sum_a \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a), d^\pi(s) \text{is stationary distribution.}$$

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

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

$$= \mathbb{E}_\pi \left[Q^\pi(s,a) \nabla_\theta \ln \pi_\theta(a|s)\right] \quad \text{; Because } (\ln x)' = \frac{1}{x}, \quad (Q.E.D)$$

Value Function Estimation with Baselines

In practice, estimates of the policy gradients can become very noisy $\longrightarrow$ reduce the variance of the gradients while keeping the bias low.

Let $\hat{Q}(s, a)$ be a sampled estimate of $Q_\pi(s, a)$, assuming $\mathbb{E}[\hat{Q}(s, a)] = Q_\pi(s, a)$. Then, we can construct a new estimator $\hat{Q}_b(s, a)$ by subtracting some baseline $b : \mathcal{S} \to \mathbb{R}$, i.e. $\hat{Q}_b(s, a) = \hat{Q}(s, a) - b(s)$. Our sampled estimate of the gradient $\nabla_\theta J(\theta)$ becomes

$$\hat{\nabla}_\theta J(\theta) = \nabla_\theta \ln \pi_\theta(a \mid s) \left(\hat{Q}(s, a) - b(s) \right).$$

In expectation over the policy π, this yields

$$\mathbb{E}_\pi \left[\hat{\nabla}_\theta J(\theta) \right] = \mathbb{E}_\pi \left[\nabla_\theta \ln \pi_\theta(A \mid S) \left(\hat{Q}(S, A) - b(S) \right) \right]$$

$$= \mathbb{E}_\pi \left[\nabla_\theta \ln \pi_\theta(A \mid S) \hat{Q}(S, A) \right] - \mathbb{E}_\pi \left[\nabla_\theta \ln \pi_\theta(A \mid S) b(S) \right].$$

Value Function Estimation with Baselines

To demonstrate the unbiased estimation, we can have:

$$\begin{aligned}
&\mathbb{E}_{S\sim d^{\pi_\theta}, A\sim\pi_\theta}\left[\nabla_\theta \ln \pi_\theta(A|S)\, b(S)\right] \\
&= \sum_{\mathcal{S}} d^\pi(s) \sum_{\mathcal{A}} \pi_\theta(a|s)\, \nabla_\theta \ln \pi_\theta(a|s)\, b(s) \\
&= \sum_{\mathcal{S}} d^\pi(s)\, b(s) \sum_{\mathcal{A}} \pi_\theta(a|s)\, \nabla_\theta \ln \pi_\theta(a|s) \\
&= \sum_{\mathcal{S}} d^\pi(s)\, b(s) \sum_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) \\
&= \sum_{\mathcal{S}} d^\pi(s)\, b(s)\, \nabla_\theta \sum_{\mathcal{A}} \pi_\theta(a|s) \\
&= \sum_{\mathcal{S}} d^\pi(s)\, b(s)\, \nabla_\theta 1 \\
&= 0
\end{aligned}$$

Value Function Estimation with Baselines

Next, we analyze the effect on the variance of the gradient estimates:

$$\arg\min_b \mathrm{Var}_\pi\left[\nabla_\theta \ln \pi_\theta(A|S)\big(\hat{Q}(S,A) - b(S)\big)\right]$$

$$= \arg\min_b \mathbb{E}_\pi\left[\Big(\nabla_\theta \ln \pi_\theta(A|S)\big(\hat{Q}(S,A) - b(S)\big)\Big)^2\right]$$

$$\approx \arg\min_b \Big(\mathbb{E}_\pi\left[\big(\nabla_\theta \ln \pi_\theta(A|S)\big)^2\right] \cdot \mathbb{E}_\pi\left[\big(\hat{Q}(S,A) - b(S)\big)^2\right]\Big).$$

Importance Sampling

Traditionally, this is only needed in off-policy RL, where we sample transitions using a behavior policy β but want to calculate expectations over the target policy π.

Definition (Importance Sampling Ratio) Given a target policy π, a behavior policy β and a trajectory $\tau = (a_t, s_{t+1}, a_{t+1}, \dots, s_T)$ generated by β, the importance sampling ratio is defined as

$$\rho_{t:T-1} := \frac{\prod_{k=t}^{T-1} \pi(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k)}{\prod_{k=t}^{T-1} \beta(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k)} = \frac{\prod_{k=t}^{T-1} \pi(a_k \mid s_k)}{\prod_{k=t}^{T-1} \beta(a_k \mid s_k)}.$$

Importance Sampling

Let $\mathcal{T}$ be the set of possible trajectories. By multiplying returns of trajectories $\tau \in \mathcal{T}$ generated by the behavior policy β with the importance sampling ratio ρ, we get

$$\mathbb{E}_\beta[\rho_{t:T-1} G_t \mid S_t = s] = \mathbb{E}_\beta[\rho_{t:T-1} G(\tau) \mid S_t = s]$$

$$= \sum_{\tau \in \mathcal{T}} \rho_{t:T-1} G(\tau) \prod_{k=t}^{T-1} \beta(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k)$$

$$= \sum_{\tau \in \mathcal{T}} \frac{\prod_{k=t}^{T-1} \pi(a_k \mid s_k)}{\prod_{k=t}^{T-1} \beta(a_k \mid s_k)} G(\tau) \prod_{k=t}^{T-1} \beta(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k)$$

$$= \sum_{\tau \in \mathcal{T}} G(\tau) \prod_{k=t}^{T-1} \frac{\pi(a_k \mid s_k)}{\beta(a_k \mid s_k)} \beta(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k)$$

$$= \sum_{\tau \in \mathcal{T}} G(\tau) \prod_{k=t}^{T-1} \pi(a_k \mid s_k) P(s_{k+1} \mid s_k, a_k) = \mathbb{E}_\pi[G_t \mid S_t = s] = V_\pi(s).$$