# Policy Gradient

Pedram Rabiee

## Contents

# 1 Introduction

The policy gradient methods target at modeling and optimizing the policy directly. The policy is modeled with a parameterized function with respect to $\theta$

$$\pi(a \mid s, \theta) = \pi_\theta(a \mid s) = P\{A_t = a \mid S_t = s, \theta_t = \theta\}.$$

The objective is to maximize some performance measure $J(\theta)$

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, J(\theta)$$
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

To ensure exploration we generally require that the policy never becomes deterministic.

## 1.1 Why Policy Gradient?

**Advantages**

- Better convergence properties.
- Effective in high-dimensional (there is no need to use maximization) or continuous action spaces.
- Can learn stochastic policies. In some problems, the optimal policy is stochastic policy (non-Markovian, aliased state). There is always an optimal deterministic policy for a given MDP, that's for Markov Decision Process where we have a perfect state representation. The moment you have state aliasing (so you are partially observed), you are in POMDP, or your function approximator, or the features that you use limit your view of the world which is equivalent to being in POMDP, then it can be optimal to use a stochastic policy, in which case policy-based method can do better than value based methods.
- For some problems policy is simpler than action-value function. For these problems a policy-based method will typically learn faster and yield a superior asymptotic policy.
- With value methods when you are using the max are extremely aggressive. This max in one step is trying to improve policy in direction that absolutely pushes you all the way to what currently think is the best policy. While policy gradient methods just take a little step in that direction, they smoothly update in that direction which makes them more stable, also sometimes less efficient, sometimes with high variance.

**Disadvantages**

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

# 2 Raw Policy Gradient

## 2.1 Definitions and Notations

| Symbol | Meaning |
|---|---|
| $\rho^\pi(s \to x, k)$ | Probability of transitioning from state $s$ to state $x$ in $k$ steps under policy $\pi$ |
| $d^\pi(s)$ | Stationary probability of state $s$ under policy $\pi$ |
| $p_0$ | Initial state distribution |
| $G_t$ | Reward-to-go, starting from time $t$ |
| $\tau^{(t)}$ | Trajectory of $(s, a, r)$ tuples up to time $t$: the set $\{s_{t'}, a_{t'}, r_{t'}\}_{t'=0}^{t}$ |
| $p_\theta$ | Trajectory distribution parameterized by parameter $\theta$ |
| $w_{t_1 \to t_2}$ | Product of importance sampling ratios from time $t_1$ to $t_2 \geq t_1$ (will be defined in Section 2.6) |

Consider the following definitions

$$\rho^\pi(s \to x, k) \triangleq \sum_{a_0} \pi(a_0|s) \sum_{s_1} P(s_1|s, a_0) \sum_{a_1} \pi(a_1|s_1) \sum_{s_2} P(s_2|s_1, a_1) \cdots \pi(a_{k-1}|s_{k-1}) P(x|s_{k-1}, a_{k-1}),$$

$$\rho^\pi(s \to x, 0) \triangleq \begin{cases} 1, & \text{if } x = s, \\ 0, & \text{else,} \end{cases}$$

$$\eta^\pi(s) \triangleq \sum_{s_0} p_0(s_0) \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k).$$

Then, we have

$$\rho^\pi(s \to x, 1) = \sum_{a_0} \pi(a_0|s) P(x|s, a_0),$$

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

The stationary distribution $d^\pi$ is defined as

$$d^\pi(s) \triangleq \frac{\eta^\pi(s)}{\sum_s \eta^\pi(s)}.$$

Also, consider reward-to-go $G_t$ defined as

$$G_t \triangleq \sum_{t'=t}^{T-1} r(s_{t'}, a_{t'}).$$

For the trajectory $\tau^{(t)} \triangleq (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_t, a_t, r_t)$, the trajectory distribution $p_\theta$ is defined as

$$p_\theta(\tau^{(t)}) \triangleq p_0(s_0) \prod_{t'=0}^{t-1} \pi_\theta(a_{t'}|s_{t'}) P(s_{t'+1}|s_{t'}, a_{t'}) \pi_\theta(a_t|s_t)$$

Note that for the case where $t = T$, where $T$ is the length of an episode, we may use $\tau$ to refer to $\tau^{(T-1)}$.

## 2.2 Policy Gradient Derivation

Starting from different objectives, we obtain different policy gradient expressions.Table 1 lists two objectives and their corresponding policy gradient expressions.

| **Objective:** $J(\theta)$ | **Objective's Gradient:** $\nabla_\theta J(\theta)$ |
|---|---|
| $\mathbb{E}_{s_0 \sim p_0}\left[V^{\pi_\theta}(s_0)\right]$ | $\mathbb{E}_{a \sim \pi_\theta, s \sim d^{\pi_\theta}}\left[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a \mid s)\right]$ |
| $\sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(t)} \sim p_\theta}[r(s_t, a_t)]$ | $\mathbb{E}_{\tau^{(T-1)} \sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t\right]$ |

Table 1: Policy gradient objective and expressions

Now, consider the first objective $J(\theta) = \mathbb{E}_{s_0 \sim p_0}\left[V^{\pi_\theta}(s_0)\right]$. We put forward the proof from [1]. It is to be noted that, the subscript $\theta$ is often times dropped from $\pi$ in the proofs to save space.

**Policy Gradient Derivation for $J(\theta) = \mathbb{E}_{s_0 \sim p_0}\left[V^{\pi_\theta}(s_0)\right]$**

We first seek to extend the expression for $\nabla_\theta V^\pi(s)$, for $s \in \mathcal{S}$.

$$
\begin{aligned}
\nabla_\theta V^\pi(s) &= \nabla_\theta \left( \sum_a \pi_\theta(a|s) Q^\pi(s,a) \right) \\
&= \sum_a \left( \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \nabla_\theta Q^\pi(s,a) \right) \\
&= \sum_a \left( \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \nabla_\theta \sum_{s'} P(s'|s,a)[r(s,a) + V^\pi(s')] \right) \\
&= \sum_a \left( \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) + \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V^\pi(s') \right) \\
&= \underbrace{\sum_a \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a)}_{\triangleq \phi(s)} + \sum_a \pi_\theta(a|s) \sum_{s'} P(s'|s,a) \nabla_\theta V^\pi(s') \\
&= \phi(s) + \sum_{s'} \rho(s \to s', 1) \nabla_\theta V^\pi(s') \\
&= \phi(s) + \sum_{s'} \rho(s \to s', 1) \left[ \phi(s') + \sum_{s''} \rho(s' \to s'', 1) \nabla_\theta V^\pi(s'') \right] \\
&= \phi(s) + \sum_{s'} \rho(s \to s', 1) \phi(s') + \sum_{s'} \rho(s \to s', 1) \sum_{s''} \rho(s' \to s'', 1) \nabla_\theta V^\pi(s'') \\
&= \phi(s) + \sum_{s'} \rho(s \to s', 1) \phi(s') + \sum_{s''} \rho(s \to s'', 2) \nabla_\theta V^\pi(s'') \\
&= \phi(s) + \sum_{s'} \rho(s \to s', 1) \phi(s') + \sum_{s''} \rho(s \to s'', 2) \phi(s'') \\
&\qquad + \sum_{s'''} \rho(s \to s''', 3) \nabla_\theta V^\pi(s''') \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \rho^\pi(s \to x, k) \phi(x).
\end{aligned}
$$

Now, we derive the expression for $\nabla_\theta J(\theta)$.

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{s_0 \sim p_0}\left[V^\pi(s_0)\right] \\
&= \sum_{s_0} p_0(s_0) \sum_{s \in \mathcal{S}} \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k) \phi(x) \\
&= \sum_{s \in \mathcal{S}} \phi(s) \underbrace{\sum_{s_0} p_0(s_0) \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k)}_{= \eta^\pi(s)}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_s \eta^\pi(s)\phi(s) \\
&= \underbrace{\sum_{s'} \eta^\pi(s')}_{=1 \text{ for continuing case}} \sum_s \frac{\eta^\pi(s)}{\sum_{s'} \eta^\pi(s')}\phi(s) \\
&= \sum_s d^\pi(s)\phi(s) \\
&= \sum_s d^\pi(s) \sum_a \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) \\
&= \sum_s d^\pi(s) \sum_a Q^\pi(s,a)\pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \\
&= \mathbb{E}_{a\sim\pi_\theta, s\sim d^{\pi_\theta}} \left[ Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s) \right]
\end{aligned}
$$

What the expression $\nabla J(\theta) = \mathbb{E}_{a\sim\pi_\theta, s\sim d^{\pi_\theta}} \left[ Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s) \right]$ asserts is that to improve the policy $\pi_\theta$, take the average of $Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)$, for all state and action pairs $(s,a)$, where the state is sampled from the stationary distribution $d^{\pi_\theta}$, and the action is sampled from the policy $\pi_\theta$.

Now, let's consider the second objective $J(\theta) = \sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(t)}\sim p_\theta}[r(s_t, a_t)]$.

**Policy Gradient Derivation for $J(\theta) = \sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(t)}\sim p_\theta}[r(s_t, a_t)]$**

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(t)}\sim p_\theta}[r(s_t, a_t)] \\
&= \sum_{t=0}^{T-1} \nabla_\theta \mathbb{E}_{\tau^{(t)}\sim p_\theta}[r(s_t, a_t)] \\
&= \sum_{t=0}^{T-1} \int \nabla_\theta \left( p_\theta(\tau^{(t)}) r(s_t, a_t) \right) \mathrm{d}\tau^{(t)} \\
&= \sum_{t=0}^{T-1} \int p_\theta(\tau^{(t)}) \nabla_\theta \log p_\theta(\tau^{(t)}) r(s_t, a_t) \mathrm{d}\tau^{(t)} \\
&= \sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(t)}\sim p_\theta} \left[ \nabla_\theta \log p_\theta(\tau^{(t)}) r(s_t, a_t) \right].
\end{aligned}
$$

Now, first take a look at $\nabla_\theta \log p_\theta(\tau^{(t)})$

$$
\begin{aligned}
\nabla_\theta \log p_\theta(\tau^{(t)}) &= \nabla_\theta \log \left( p_0(s_0) \prod_{t'=0}^{t-1} \pi_\theta(a_{t'}|s_{t'}) P(s_{t'+1}|s_{t'}, a_{t'}) \pi_\theta(a_t|s_t) \right) \\
&= \nabla_\theta \left( \log p_0(s_0) + \sum_{t'=0}^{t-1} \log \pi_\theta(a_{t'}|s_{t'}) + \log P(s_{t'+1}|s_{t'}, a_{t'}) + \log \pi_\theta(a_t|s_t) \right) \\
&= \sum_{t'=0}^{t} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}).
\end{aligned}
$$

Then, we have

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(t)}\sim p_\theta} \underbrace{\left[ r(s_t, a_t) \sum_{t'=0}^{t} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \right]}_{\triangleq g_t}$$

Now, we can take the summation inside the expectation by writing all expectations under$p_\theta(\tau^{(T-1)})$. In other words, $\sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(t)}\sim p_\theta}[g_t] = \mathbb{E}_{\tau^{(T-1)}\sim p_\theta}\left[\sum_{t=0}^{T-1} g_t\right]$. We will demonstrate why this is correct, later. Thus, we have

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau^{(T-1)}\sim p_\theta}\left[ \sum_{t=0}^{T-1} r(s_t, a_t) \sum_{t'=0}^{t} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'}) \right]$$

Using an algebraic trick, we can rearrange this expression as follows:
Let $f_t \triangleq \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'})$ and $r_t \triangleq r(s_t, a_t)$. We expand the expression inside the expectation in the following form:

$$\begin{aligned}
&r_0 f_0 + \\
&r_1 f_0 + r_1 f_1 + \\
&r_2 f_0 + r_2 f_1 + r_2 f_2 + \\
&\dots \\
&r_{T-1} f_0 + r_{T-1} f_1 + \ldots + r_{T-1} f_{T-1}
\end{aligned}$$

Now, instead of summing row-wise, we sum column-wise:

$$(r_0 + r_1 + \ldots + r_{T-1}) f_0 + (r_1 + \ldots + r_{T-1}) f_1 + (r_2 + \ldots + r_{T-1}) f_2 + \ldots + r_{T-1} f_{T-1}$$

Thus, we have:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau^{(T-1)}\sim p_\theta}\left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T-1} r(s_t, a_t) \right] = \mathbb{E}_{\tau^{(T-1)}\sim p_\theta}\left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \right]$$

This concludes the proof, except for the fact that we need to show that:

$$\sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(t)}\sim p_\theta}[g_t] = \mathbb{E}_{\tau^{(T-1)}\sim p_\theta}\left[ \sum_{t=0}^{T-1} g_t \right], \tag{1}$$

where $g_t \triangleq r(s_t, a_t) \sum_{t'=0}^{t} \nabla_\theta \log \pi_\theta(a_{t'}|s_{t'})$. We have

$$\begin{aligned}
\mathbb{E}_{\tau^{(T-1)}\sim p_\theta}\left[ \sum_{t=0}^{T-1} g_t \right] &= \mathbb{E}_{\tau^{(T-1)}\sim p_\theta}\left[ g_0 + g_1 + \ldots + g_t + \ldots + g_{T-1} \right] \\
&= \sum_{t=0}^{T-1} \mathbb{E}_{\tau^{(T-1)}\sim p_\theta}[g_t]
\end{aligned}$$

Thus, in order to prove (1), it is sufficient to show that for arbitrary $t$, $\mathbb{E}_{\tau^{(t)}\sim p_\theta}[g_t] = \mathbb{E}_{\tau^{(T-1)}\sim p_\theta}[g_t]$.

$$\mathbb{E}_{\tau^{(T-1)}\sim p_\theta}[g_t] = \int_{s_0}\int_{a_0}\dots\int_{s_t}\int_{a_t}\dots\int_{s_T} p_0(s_0)\pi_\theta(a_0|s_0)\dots P(s_t|s_{t-1}, a_{t-1})\pi_\theta(a_t|s_t) g_t \mathrm{d}s_0 \mathrm{d}a_0 \dots \mathrm{d}s_t \mathrm{d}a_t \dots \mathrm{d}s_T$$

$$
\begin{aligned}
&= \int_{s_0}\int_{a_0}\dots\int_{s_t}\int_{a_t} p_0(s_0)\pi_\theta(a_0|s_0)\dots P(s_t|s_{t-1},a_{t-1})\pi_\theta(a_t|s_t)g_t \\
&\qquad \int_{s_{t+1}}\dots\int_{a_{T-1}} \pi_\theta(a_{T-1}|s_{T-1})\,\mathrm{d}a_{T-1}\dots\mathrm{d}a_0\mathrm{d}s_0 \\
&= \int_{s_0}\int_{a_0}\dots\int_{s_t}\int_{a_t} p_0(s_0)\pi_\theta(a_0|s_0)\dots P(s_t|s_{t-1},a_{t-1})\pi_\theta(a_t|s_t)g_t \\
&\qquad \int_{s_{t+1}}\dots\int_{a_{T-1}} \pi_\theta(a_{T-1}|s_{T-1})\,\mathrm{d}a_{T-1}\dots\mathrm{d}a_0\mathrm{d}s_0 \\
&= \int p_\theta(\tau^{(t)})g_t\,\mathrm{d}\tau^{(t)} = \mathbb{E}_{\tau^{(t)}\sim p_\theta}[g_t]
\end{aligned}
$$

This concludes the result.

**Important note:** The version of the policy gradient described in this section is on-policy since the expectations are under current policy. Thus, you can't use samples that come from other policies, and samples collected under the current policy have to be thrown away after each update. This makes this method extremely inefficient.

### 2.3 REINFORCE

**REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for $\pi_*$**

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

Figure 1: REINFORCE Algorithm

### 2.4 State-Based Baseline

In order to reduce variance, we introduce a baseline to our gradient estimate. The addition of a state-based baseline $b(s)$ does not introduce bias into our estimate. That is, the gradient of the policy objective function $J(\theta)$ can be expressed as:

$$
\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)G_t\right] = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))\right]
$$

In other words, $\mathbb{E}_{\tau\sim p_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right] = 0$ for any $t$.
Before proving this we first show that

$$
\int_{a_t} \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\,\mathrm{d}a_t = 0 \tag{2}
$$

$$
\int_{a_t} \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\,\mathrm{d}a_t = \int_{a_t} \pi_\theta(a_t|s_t)\left(\frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}\right)\mathrm{d}a_t
$$

$$= \int_{a_t} \nabla_\theta \pi_\theta(a_t|s_t)\,\mathrm{d}a_t$$
$$= \nabla_\theta \int_{a_t} \pi_\theta(a_t|s_t)\,\mathrm{d}a_t = \nabla_\theta \cdot 1 = 0$$

Next, consider the proof of $\mathbb{E}_{\tau\sim p_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\right] = 0$.

**$\mathbb{E}_{\tau\sim p_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\right] = 0$**

$$\begin{aligned}
&\mathbb{E}_{\tau\sim p_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\right] \\
&= \int p_\theta(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) \mathrm{d}\tau \\
&= \int_{s_0}\int_{a_0}\cdots\int_{s_t}\int_{a_t}\cdots\int_{s_T} p_0(s_0)\pi_\theta(a_0|s_0)\cdots P(s_t|s_{t-1},a_{t-1})\pi_\theta(a_t|s_t)\cdots P(s_T|s_{T-1},a_{T-1}) \\
&\qquad \nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\,\mathrm{d}s_0\,\mathrm{d}a_0\cdots\mathrm{d}s_T \\
&= \int_{s_0}\cdots\int_{s_t}\int_{a_t} p_0(s_0)\cdots P(s_t|s_{t-1},a_{t-1})\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) \\
&\qquad \cdots\int_{a_{T-1}} \pi_\theta(a_{T-1}|s_{T-1}) \underbrace{\int_{s_T} P(s_T|s_{T-1},a_{T-1})\,\mathrm{d}s_T}_{=1}\,\mathrm{d}a_{T-1}\cdots\mathrm{d}a_0\,\mathrm{d}s_0 \\
&= \int_{s_0}\cdots\int_{s_t}\int_{a_t} p_0(s_0)\cdots P(s_t|s_{t-1},a_{t-1})\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) \\
&\qquad \cdots\underbrace{\int_{a_{T-1}} \pi_\theta(a_{T-1}|s_{T-1})\,\mathrm{d}a_{T-1}}_{=1}\cdots\mathrm{d}a_0\,\mathrm{d}s_0 \\
&= \int_{s_0}\int_{a_0}\cdots\int_{s_t}\int_{a_t} p_0(s_0)\pi_\theta(a_0|s_0)\cdots P(s_t|s_{t-1},a_{t-1})\pi_\theta(a_t|s_t) \\
&\qquad \nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\,\mathrm{d}a_t\,\mathrm{d}s_t\cdots\mathrm{d}a_0\,\mathrm{d}s_0 \\
&= \int_{s_0}\int_{a_0}\cdots\int_{s_t} p_0(s_0)\pi_\theta(a_0|s_0)\cdots P(s_t|s_{t-1},a_{t-1}) b(s_t) \\
&\qquad \underbrace{\int_{a_t} \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\,\mathrm{d}a_t}_{=0 \text{ from (2)}}\,\mathrm{d}s_t\cdots\mathrm{d}a_0\,\mathrm{d}s_0 \\
&= 0
\end{aligned}$$

Thus, we have shown that $\mathbb{E}_{\tau\sim p_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)\right] = 0$.

But why does it reduce the variance? Intuitively, by making the target values smaller (by subtracting the baseline), we are reducing the variance. To understand why the introduction of a baseline reduces variance, we consider informal reasoning by finding the baseline that minimizes the variance.

The variance $\mathsf{Var}(X)$ of a random variable $X$ is defined as

$$\mathsf{Var}(X) \triangleq \mathbb{E}[X^2] - \mathbb{E}[X]^2.$$

We want to show that $\mathsf{Var}(X) < \mathsf{Var}(Y)$, where $X = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))$, and $Y = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$.

**Analysis 1. Finding the baseline that minimizes the variance**

Let $X = \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\triangleq \psi_t}(G_t - \underbrace{b(s_t)}_{\triangleq b_t})$

Now, let's consider the variance:

$$\text{Var}(\psi_t(G_t - b_t)) = \mathbb{E}[(\psi_t(G_t - b_t))^2] - \mathbb{E}[\psi_t(G_t - b_t)]^2$$

We showed that $\mathbb{E}[\psi_t(G_t - b_t)] = \mathbb{E}[\psi_t(G_t)]$, thus, the second term, $\mathbb{E}[g_t(G_t - b_t)]^2$, does not depend on the choice of $b_t$. In order to minimize the variance, we need to find the optimal baseline $b(s_t)$ that satisfies the condition $\frac{d\text{Var}(\psi_t(G_t - b_t)}{db_t} = 0$. Differentiating with respect to $b_t$ and setting it to zero, we obtain:

$$\frac{d\text{Var}(\psi_t(G_t - b_t))}{db_t} = \frac{d}{db_t}\mathbb{E}[(\psi_t(G_t - b_t))^2] = \mathbb{E}[-2\psi_t^2(G_t - b_t)] = 0$$

Simplifying the equation, we find that:

$$\mathbb{E}[\psi_t^2 G_t] = \mathbb{E}[\psi_t^2 b_t]$$

This allows us to determine the optimal baseline $b_t$ as:

$$b_t = \frac{\mathbb{E}[(\nabla_\theta \log \pi_\theta(a_t|s_t))^2 G_t]}{\mathbb{E}[(\nabla_\theta \log \pi_\theta(a_t|s_t))^2]}$$

While this expression provides a general formula for the optimal baseline, in practice, a common choice is to use the on-policy value function $V_\pi(s_t)$ as the baseline.

- Another informal reasoning can be found here.
- For a discussion on a more general form of a baseline see section 4.1.

### 2.5 Vanilla Policy Gradient

**Algorithm 1** Vanilla Policy Gradient Algorithm

1: Input: initial policy parameters $\theta_0$, initial value function parameters $\phi_0$
2: **for** $k = 0, 1, 2, ...$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function $V_{\phi_k}$.
6: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

7: Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.
8: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t\right)^2,$$

typically via some gradient descent algorithm.
9: **end for**

Figure 2: Vanilla Policy Gradient Algorithm

### 2.6 Off-Policy Policy Gradient Using Importance Sampling

If we were to collect samples using a policy other than $\pi_\theta$, such as $\pi_\beta$, then the gradient for the policy can be rewritten as the expectation of trajectories under $\pi_\beta$ using importance sampling. Let us first define $w_{t_1 \to t_2}$ for times $t_1$ and $t_2$ where $t_2$ is greater than or equal to $t_1$.

$$w_{t_1 \to t_2} \triangleq \prod_{t=t_1}^{t_2} \frac{\pi_\theta(a_t|s_t)}{\pi_\beta(a_t|s_t)}. \tag{3}$$

Next, let $r(\tau)$ denote the sum of reward for trajectory $\tau$, then

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[r(\tau)] = \mathbb{E}_{\tau \sim p_\beta}\left[\frac{p_\theta(\tau)}{p_\beta(\tau)} r(\tau)\right]$$

where

$$\frac{p_\theta}{p_\beta} = \frac{p_0(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t)P(s_{t+1}|a_t,s_t)}{p_0(s_0)\prod_{t=0}^{T-1}\pi_\beta(a_t|s_t)P(s_{t+1}|a_t,s_t)} = \prod_{t=0}^{T-1}\frac{\pi_\theta(a_t|s_t)}{\pi_\beta(a_t|s_t)} = w_{0\to T-1}.$$

Thus,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\beta}\left[w_{0\to T-1}\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=0}^{T-1} r(s_t,a_t)\right)\right] \tag{4}$$

This version of off-policy policy gradient however has high variance. From here forward, different works, make different estimations of (4) in order to reduce the variance.

Causality trick 1: Current action cannot affect the past rewards. Thus, we change the trajectory reward to reward-to-go.

$$\begin{aligned}
\nabla_\theta J(\theta) &\approx \mathbb{E}_{\tau \sim p_\beta}\left[w_{0\to T-1}\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\sum_{t'=t}^{T} r(s_{t'},a_{t'})\right)\right] \\
&= \mathbb{E}_{\tau \sim p_\beta}\left[\left(\sum_{t=0}^{T-1} w_{0\to T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\sum_{t'=t}^{T-1} r(s_{t'},a_{t'})\right)\right] \\
&= \mathbb{E}_{\tau \sim p_\beta}\left[\left(\sum_{t=0}^{T-1} w_{0\to t}\nabla_\theta \log \pi_\theta(a_t|s_t) w_{t+1\to T-1}\sum_{t'=t}^{T-1} r(s_{t'},a_{t'})\right)\right]
\end{aligned}$$

Causality trick 2: Future actions cannot affect the current reward. Thus, we drop the importance samplings into the future. First note that

$$w_{t+1\to T-1} = \frac{w_{0\to T-1}}{w_{0\to t}}$$

$$\begin{aligned}
\nabla_\theta J(\theta) &\approx \mathbb{E}_{\tau \sim p_\beta}\left[\left(\sum_{t=0}^{T-1} w_{0\to t}\nabla_\theta \log \pi_\theta(a_t|s_t)\sum_{t'=t}^{T-1}\frac{w_{0\to T-1}}{w_{0\to t}} r(s_{t'},a_{t'})\right)\right] \\
&\approx \mathbb{E}_{\tau \sim p_\beta}\left[\left(\sum_{t=0}^{T-1} w_{0\to t}\nabla_\theta \log \pi_\theta(a_t|s_t)\sum_{t'=t}^{T-1}\frac{w_{0\to t'}}{w_{0\to t}} r(s_{t'},a_{t'})\right)\right]
\end{aligned} \tag{5}$$

Introducing state-based baseline and discount factor to (5) yields

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau \sim p_\beta}\left[\left(\sum_{t=0}^{T-1} w_{0\to t}\gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1}\gamma^{t'-t}\frac{w_{0\to t'}}{w_{0\to t}} r(s_{t'},a_{t'}) - b(s_t)\right)\right)\right], \tag{6}$$

which is the formulation in [2].

As before, (6) can have high variance due to having the multiplication of importance sampling fractions. One approach to further reduce the variance is to make sure that the learned policy $\pi_\theta$ does not deviate too far from the behavior policy $\pi_\beta$, thus keeping the variance of the importance weights from becoming too large [3]. One way to achive this to introduce regulizer [4, 3]:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau \sim p_\beta} \left[ \left( w_{0 \to T-1} \sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}(s_t, a_t) \right) + \lambda \log w_{0 \to T-1} \right].$$

This regulizer which is the softmax over the importance weight, adjust the policy $\pi_\theta$ to ensure that at least one sample has a high importance weight.

# 3 Actor-Critic

To reduce variance in policy gradient methods, one approach is to employ a bootstrapping target. By replacing the advantage function used in the Vanilla policy gradient with a TD(0)-like target, we arrive at the actor-critic algorithm. This method mitigates variance but introduces bias. Much like the Vanilla policy gradient, the actor-critic algorithm employs two neural networks: one for value-function approximation and another for policy. However, the key distinction lies in the actor-critic's use of the same network to estimate both the baseline and the return, as opposed to Vanilla PG, where the value network serves only as a baseline. The policy network, often referred to as "the actor," interacts with the environment, while the value network, known as "the critic," maintains the value of the actions taken by the actor.

### 3.1 Actor-Critic Derivation

The policy gradient expression

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right],$$

can be interpreted as a weighted version of the maximum likelihood objective, with the weights being the single-sample reward-to-go. In other words, policy gradient increases or decreases the likelihood of an action depending on the value of its single-sample reward-to-go. Compare this to behavioral cloning, where the likelihood of all actions in a given dataset is increased irrespective of their corresponding reward-to-go:

$$\nabla_\theta J_{\mathrm{BC}}(\theta) = \mathbb{E}_{\tau\sim p_\beta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\right].$$

However, one way to reduce the variance in policy gradient is to replace the single-sample reward-to-go with the expected return given the state and action. That is

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\mathbb{E}_\pi[G_t|s_t,a_t]\right] \\
&= \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) Q^\pi(s_t,a_t)\right] \\
&= \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\left[R(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim P(s_{t+1}|s_t,a_t)}[V^\pi(s_{t+1})]\right]\right].
\end{aligned}$$

Now we make an approximation and we replace $\mathbb{E}_{s_{t+1}\sim P(s_{t+1}|s_t,a_t)}[V^\pi(s_{t+1})]$ by $V^\pi(s_{t+1})$, meaning that we consider a single-sample estimate for $V^\pi(s_{t+1})$. Note, however, that we are making the single-sample estimate only for one timestep. Since, after one timestep, at $s_{t+1}$, since we are using $V^\pi(s_{t+1})$, we are going to be under $\mathbb{E}_\pi$ thereafter. **This will introduce bias at the cost of lower variance**. The modified policy gradient then becomes

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\left[R(s_t,a_t) + V^\pi(s_{t+1})\right]\right].$$

To obtain the actor-critic method, we introduce the state-base baseline and discount factor, $V^\pi(s_t)$ and $\gamma$ and we define the estimate of the advantage at time step $t$, denoted as $\hat{A}_t$, as

$$\hat{A}_t = \underbrace{R(s_t,a_t) + \gamma V^\pi_\phi(s_{t+1})}_{\text{return estimator}} - \underbrace{V^\pi_\phi(s_t)}_{\text{baseline}},$$

and

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\right].$$

This forms the basis of the online actor-critic algorithm.

**One-step Actor–Critic (episodic), for estimating $\pi_\theta \approx \pi_*$**

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize $S$ (first state of episode)
  $I \leftarrow 1$
  Loop while $S$ is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action $A$, observe $S', R$
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if $S'$ is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Figure 3: Online Actor-Critic Algorithm

**Batch actor-critic algorithm**

(1) Sample $\{s_i, a_i, s_i', r_i\}$ under $\pi_\theta(a|s)$ (run the agent on the environment)

(2) Fit $\hat{V}_\phi^\pi(s)$ to sampled reward sum or TD

(3) Evaluate $\hat{A}^\pi(s_i, a_i) = r(s_i, a_i) + \hat{V}_\phi^\pi(s_i') - \hat{V}_\phi^\pi(s_i)$

(4) Update $\theta$ using $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i)$

(5) Go back to step 1

This represents the Batch actor-critic algorithm, where the agent collects a batch of samples from the environment, updates the value function approximation, evaluates the advantage function, and performs a policy update using the policy gradient. The process is repeated iteratively to improve the policy and value function estimates.

### 3.2 Discount Factor

The discount factor introduces changes to the original MDP (with the transition dynamics $p(s'|s,a)$), altering it into another MDP depicted in Figure 4. This modified MDP incorporates a "death" state, denoted as $s_{\mathrm{d}}$, which corresponds to adjusted transition dynamics $\tilde{p}(s'|s,a)$ defined as:

$$\tilde{p}(s'|s,a) = \begin{cases} 1-\gamma, & \text{if } s' = s_{\mathrm{d}} \\ \gamma p(s'|s,a), & \text{else.} \end{cases}$$

Given this adjusted MDP (Figure 4), we express the value function $V_\pi(s)$ as follows:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_{a\sim\pi(a|s)}\left[r(s,a) + \mathbb{E}_{s'\sim\tilde{p}(s'|s,a)}[V_\pi(s')]\right] \\ &= \mathbb{E}_{a\sim\pi(a|s)}\left[r(s,a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)}[V_\pi(s')] + (1-\gamma)(\underbrace{V_\pi(s_{\mathrm{d}})}_{=0})\right] \\ &= \mathbb{E}_{a\sim\pi(a|s)}\left[r(s,a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)}[V_\pi(s')]\right] \end{aligned}$$

The policy gradient for this modified MDP, after applying the causality trick, can be approximated as

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T-1} \gamma^{t'} r(s_t, a_t)\right)\right]$$

$$= \mathbb{E}_{\tau \sim p_\theta} \left[ \sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \left( \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_t, a_t) \right) \right] \tag{7}$$

This is similar to the (6) with $\beta = \theta$. The interpretation of Equation (7) is that the introduction of the death state (due to the discount factor) results in reduced importance assigned to both future rewards and future decisions. In essence, making the correct decision sooner becomes more critical, and the impact of $\nabla_\theta \log \pi_\theta(a_t|s_t)$ diminishes as $t$ increases.

However, in practice, we commonly omit the $\gamma^t$ term, leading to the use of the following policy gradient approximation:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\tau \sim p_\theta} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \left( \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_t, a_t) \right) \right] \tag{8}$$

Based on Equation (8), the sole alteration in the actor-critic formulation lies in the computation of $\hat{A}^\pi(s, a)$, which is expressed as:

$$\hat{A}^\pi(s, a) = r(s, a) + \gamma \hat{V}_\phi^\pi(s') - \hat{V}_\phi^\pi(s_i)$$

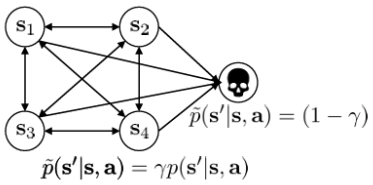


Figure 4: MDP with Discount Factor

### 3.3 Off-Policy Actor-Critic

- Transitions $(s_i, a_i, r_i, s_i')$ under any policy is stored in the replay buffer. Define $a_i^\pi \triangleq \pi_\theta(s_i)$.
- On-policy actor-critic has two limitations for being used as an off-policy algorithm:
  1. In the actor-critic derivation, we replaced $\mathbb{E}_\pi[V^\pi(s')]$ with the single-sample estimate $V^\pi(s')$. This replacement was sort of valid as long as we at least sample under the same policy. Thus, we cannot use the transition $(s, a, s', r)$ from the replay buffer which was collected under another policy to update the value function (the critic update). However, if we instead use $Q$ values. Then, we can use the transition $(s, a, s', r)$ to update the $\hat{Q}_\phi^\pi$ values.
  2. For the same reason, during the actor update, we are interested in updating the chance of selecting an action, based on the value of that action. Thus, we require to update the action that current policy would have chosen at state $s$, which is $\pi_\theta(s)$, and not the action $a$ from the transition.
- To train the actor we completely ignore the transition from the buffer, we only use $s_i$. The reason is that the reward $r_i$ in the buffer is the reward collected under $a_i$. However, the current policy takes action $a_i^\pi$, for which we don't know the reward to be collected. However, we can estimate the reward-to-go using the critic network $\hat{Q}_\phi^\pi$. We want to know how good is the action that we would have taken under current policy if we were at $s_i$.
- Note that we removed the baseline from the Actor Target.

### 3.4 Actor-critic implementation notes

- In theory, we should perform value function regression every time we update our policy to match the new policy's behavior. However, in practice, this can be computationally expensive. Therefore, we may instead take a few gradient steps at each iteration.

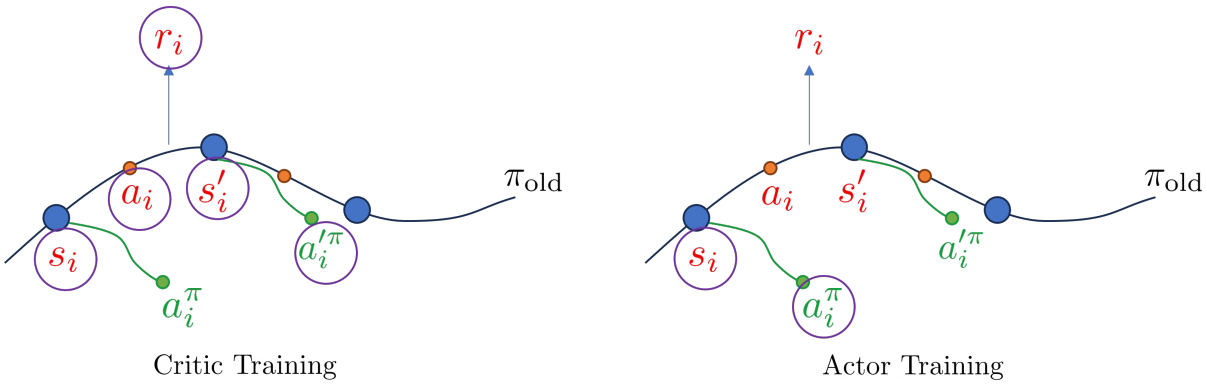


Figure 5: Off-Policy Actor-Critic Training

| | **On-Policy** | **Off-Policy** |
|---|---|---|
| **Networks** | $\hat{V}^{\pi}_{\phi}, \pi_{\theta}$ | $\hat{Q}^{\pi}_{\phi}, \pi_{\theta}$ |
| **Critic Target** | $\hat{V}^{\pi}_{\phi}(s_i) \leftarrow r_i + \gamma \hat{V}^{\pi}_{\phi}(s'_i)$ | $\hat{Q}^{\pi}_{\phi}(s_i, a_i) \leftarrow r_i + \gamma \hat{Q}^{\pi}_{\phi}(s'_i, a'^{\pi}_i)$ |
| **Actor Grad** | $\frac{1}{N} \sum_i \nabla_{\theta} \log \pi_{\theta}(a_i \vert s_i) \hat{A}^{\pi}_{\phi}(s_i, a_i)$ | $\frac{1}{N} \sum_i \nabla_{\theta} \log \pi_{\theta}(a^{\pi}_i \vert s_i) \hat{Q}^{\pi}_{\phi}(s_i, a^{\pi}_i)$ |

Table 2: On-Policy vs. Off-Policy Actor-Critic

- Since our target values are based on the old value function, we need to update the targets (i.e., $r + \gamma V(s')$) with the updated value function. This can be done by following these steps:
  1. Update targets with current value function.
  2. Regress onto targets to update the value function by taking a few gradient steps.
  3. Repeat steps 1 and 2 several times.
- The process of fitting the value function critic is an iterative process where we alternate between computing target values and updating the value function to match the targets. This iterative process is crucial for training the critic network.
- In regular policy gradient methods, it is necessary to visit a state multiple times to have a good estimate of the return. However, in actor-critic methods, even if the exact same state is not visited multiple times, there may be other similar states visited in different trials. As a result, the function approximator can capture some shared information between these states, leading to a better estimate of the expected return compared to a single-sample Monte Carlo update. However, this estimate is not as accurate as repeatedly visiting the exact same state and taking multiple trials from it, but it still provides some benefit.

# 4 On-Policy Algorithms

## 4.1 GAE: Generalized Advantage Estimation

Link to paper | Link to code ADD

### 4.1.1 Preliminaries

Consider the definitions in Figure 6, and the following definition.

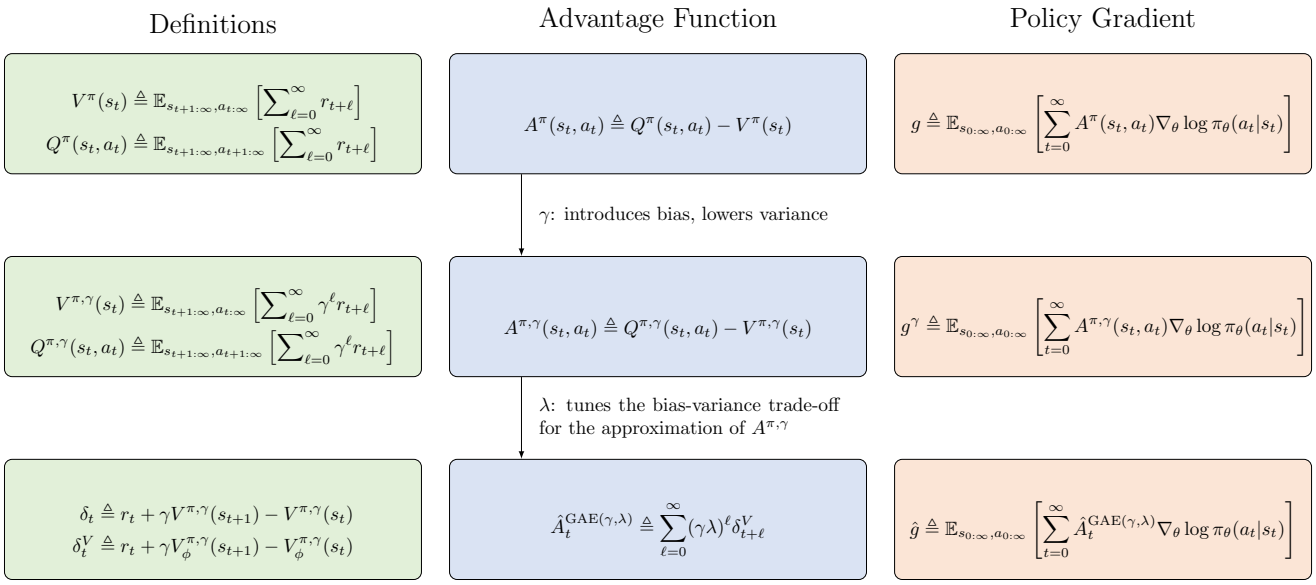


Figure 6: Advantage functions definitions and their associated policy gradients

**Definition 1** *The estimator $\hat{A}_t$ is $\gamma$-just if*

$$\mathbb{E}_{s_{0:\infty}a_{0:\infty}}[\hat{A}_t(s_{0:\infty},a_{0:\infty})\nabla_\theta \log \pi_\theta(a_t|s_t)] = \mathbb{E}_{s_{0:\infty}a_{0:\infty}}[A_{\pi,\gamma}(s_t,a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)].$$

A $\gamma$-just advantage function estimator $\hat{A}$ has the property that when it is substituted in place of $A^{\pi,\gamma}$ in the policy gradient, it does not introduce any additional bias.

With these definitions, we can present Proposition 1, which give a sufficient condition for $\hat{A}$ to be $\gamma$-just.

**Proposition 1** *Suppose that $\hat{A}_t$ can be written in the form $\hat{A}_t(s_{0:\infty},a_{0:\infty}) = Q_t(s_{t:\infty},a_{t:\infty}) - b_t(s_{0:t},a_{0:t-1})$ such that for all $(s_t,a_t)$, $\mathbb{E}_{s_{t+1:\infty},a_{t+1:\infty}|s_t,a_t}[Q_t(s_{t:\infty},a_{t:\infty})] = Q^{\pi,\gamma}(s_t,a_t)$. Then, $\hat{A}$ is $\gamma$-just.*

See [5] for the proof.

Proposition 1 shows that an advantage estimator $\hat{A}_t$ will be $\gamma$-just if it is composed of two terms - a state-dependent baseline $b_t$ and a $\gamma$-discounted Q-function estimator $Q_t$. Critically, $b_t$ can only depend on past and current states and past actions while $Q_t$ can depend on the full trajectory.

This implies we can reduce policy gradient variance by subtracting any function of past and current states and past actions as a baseline, without introducing bias.

### 4.1.2 Objectives and Approach

**Objectives**:

- Provide an intuition for a variance reduction scheme in policy gradients.

- Introduce an advantage function estimator that allows tuning the bias-variance tradeoff using two parameters $\gamma$ and $\lambda$.

**Approach**:

The GAE aims to find a lower variance approximation to the true advantage function $A^\pi$. It does this in two main ways:

- Introducing a discount factor $\gamma$ decreases variance at the cost of some bias. The GAE tries to estimate the $\gamma$-discounted advantage $A^{\pi,\gamma}$.
- The parameter $\lambda$ controls the bias-variance tradeoff in the GAE estimator. Setting $\lambda = 1$ gives an unbiased estimate of $A^{\pi,\gamma}$, while lower $\lambda$ reduces variance further but introduces some bias.

In summary, the GAE provides an advantage function estimator with two knobs - $\gamma$ and $\lambda$ - to tune the bias-variance tradeoff. This enables constructing policy gradient algorithms with reduced variance. In this method,

- GAE uses $V_\phi^{\pi,\gamma}$, the value function estimator, instead of the ground-truth $V^{\pi,\gamma}$
- $\gamma$ controls the bias in $A^{\pi,\gamma}$ with respect to $A^\pi$. $A^{\pi,\gamma}$ is an unbiased estimator of $A^\pi$ for $\gamma = 1$. Even with an accurate value function, setting $\gamma < 1$ introduces bias into the estimate.
- $\lambda$ controls the bias in $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$ with respect to $A^{\pi,\gamma}$. $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$ is an unbiased estimator of $A^{\pi,\gamma}$ for $\lambda = 1$, regardless of the accuracy of the function approximator (will be described). In contrast, $\lambda < 1$ introduces bias only when the value function is inaccurate.
- Empirically, the paper suggests that the optimal value for $\lambda$ is much lower than the optimal value for $\gamma$. This difference is likely because $\lambda$ introduces less bias than $\gamma$, especially when the value function is reasonably accurate.
- Increasing either $\lambda$ or $\gamma$ reduces bias, but they do so toward different targets, and this reduction in bias comes at the cost of increased variance.

### 4.1.3 GAE Advantage Function

GAE advantage function is equivalent to TD($\lambda$), which is the weighted-sum $n$-step TD errors, weighted by, $\lambda^n$, with $V_\phi^{\pi,\gamma}$ being the value function approximator. Consider the following defintions:

$$\begin{aligned}
\delta_t &\triangleq r_t + \gamma V_\phi^{\pi,\gamma}(s_{t+1}) - V_\phi^{\pi,\gamma}(s_t), \\
\hat{A}_t^{(k)} &\triangleq \sum_{\ell=1}^{k-1} \gamma^\ell r_{t+\ell} + \gamma^k V_\phi^{\pi,\gamma}(s_{t+k}) - V_\phi^{\pi,\gamma}(s_t) = \sum_{\ell=0}^{k-1} \gamma^\ell \delta_{t+\ell}.
\end{aligned}$$

**Infinite-horizon Case**

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} \triangleq (1-\lambda) \sum_{\ell=0}^{\infty} \lambda^\ell \hat{A}_t^{(\ell+1)} = \sum_{\ell=0}^{\infty} (\gamma\lambda)^\ell \delta_{t+\ell}$$

It can be shown from the above definition that the following holds:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \frac{1}{1-\lambda}\delta_t + \gamma\lambda \hat{A}_{t+1}^{\mathrm{GAE}(\gamma,\lambda)}$$

Note that

$$\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,1)} &= \sum_{\ell=0}^{\infty} \gamma^\ell \delta_{t+\ell} = \sum_{\ell=0}^{\infty} \gamma^\ell r_{t+\ell} - V_\pi^{\pi,\gamma}(s_t) \\
\hat{A}_t^{\mathrm{GAE}(\gamma,0)} &= \delta_t
\end{aligned}$$

Thus, $\mathrm{GAE}(\gamma, 1)$ is $\gamma$-just regardless of accuracy of $V_\phi^{\pi,\gamma}$.

**Finite-horizon Case**

$$\hat{A}_t^{\mathrm{GAE}} = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \ldots + (\gamma\lambda)^{T-t-1}\delta_{T-1},$$

and

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} \triangleq \frac{1-\lambda}{1-\lambda^T}\sum_{\ell=0}^{T-1}\lambda^\ell \hat{A}_t^{(\ell+1)} = \frac{1}{1-\lambda^T}\sum_{\ell=0}^{T-1}(\gamma\lambda)^\ell\left(1-\lambda^{T-\ell}\right)\delta_{t+\ell}$$

# 5 Off-Policy Algorithms

## 5.1 DDPG: Deep Deterministic Policy Gradient

Link to paper | Link to code ADD

### 5.1.1 Objectives and Approach

**Objective**:

- Develop a reinforcement learning method for continuous control tasks in which an agent interacts with an environment to learn optimal control policies.
- Address the challenge of high-dimensional action spaces and provide a solution for learning effective policies in such environments.

**Approach**:

- Utilize a Deep Deterministic Policy Gradient (DDPG) algorithm that combines deep neural networks for both the actor and critic networks, allowing for continuous action spaces and effective policy optimization.
- Incorporate experience replay and target networks to stabilize and improve the training process, facilitating the efficient learning of complex control policies in continuous action spaces.

### 5.1.2 DDPG Theory

Let's consider the critic $Q^{\mu}_{\theta^Q}(s,a)$, which is parameterized by $\theta^Q$, and the deterministic actor $\mu_{\theta_\mu}$, parameterized by $\theta_\mu$. The training of the $Q^{\mu}_{\theta^Q}$ network follows the DQN paradigm as outlined in [6]. However, note that the DQN training process is off-policy in nature. Therefore, to construct a fully off-policy actor-critic method, we must have an actor that is trained using off-policy trajectories. To achieve this, we must derive the expression for the off-policy deterministic policy gradient (DPG). Before delving into these details, let's introduce the concept of on-policy DPG from [7].

**On-Policy DPG**
The objective is defined as follows:

$$J(\theta^{\mu}) \triangleq \mathbb{E}_{s_0 \sim p_0}\left[V^{\mu}(s_0)\right] \tag{9}$$

This objective is analogous to the one used in stochastic policy gradient methods, with the key distinction that $\mu_{\theta^\mu}(s)$ is a deterministic policy. The on-policy deterministic policy gradient (DPG), as introduced by [7], can be expressed as:

$$\nabla_{\theta^\mu} J(\theta^{\mu}) \triangleq \mathbb{E}_{s\sim d^{\mu}}\left[\nabla_a Q^{\mu}_{\theta^Q}(s,a)|_{a=\mu(s)} \nabla_{\theta^\mu}\mu(s)\right].$$

The proof, which resembles the proof for the stochastic policy gradient but with the replacement of the stochastic policy $\pi(a|s)$ by a deterministic policy $\mu(s)$, can be found in Appendix B of [7].

**Off-Policy DPG**
The objective is defined as follows:

$$J^{\beta}(\theta^{\mu}) = \mathbb{E}_{s\sim d^{\beta}}\left[V^{\mu}(s)\right], \tag{10}$$

and the off-policy DPG is approximated as:

$$\nabla_{\theta^\mu} J^{\beta}(\theta^{\mu}) \approx \mathbb{E}_{s\sim d^{\beta}}\left[\nabla_{\theta^\mu}\mu(s) \nabla_a Q^{\mu}(s,a)|_{a=\mu(s)}\right] \tag{11}$$

Off-Policy Deterministic Policy Gradient (Off-Policy DPG)

First, we directly compute the gradient of eq. (10), yielding:

$$\begin{aligned}
\nabla_{\theta^\mu} J^\beta(\theta^\mu) &= \mathbb{E}_{s\sim d^\beta}\left[\nabla_{\theta^\mu} V^\mu(s)\right] = \mathbb{E}_{s\sim d^\beta}\left[\nabla_{\theta^\mu} Q^\mu(s,\mu(s))\right] \\
&= \mathbb{E}_{s\sim d^\beta}\left[\nabla_{\theta^\mu}\left[r(s,\mu(s)) + \gamma\int_{\mathcal{S}} p(s'|s,\mu(s))V^\mu(s')\mathrm{d}s'\right]\right] \\
&= \mathbb{E}_{s\sim d^\beta}\Bigg[\nabla_{\theta^\mu}\mu(s)\nabla_a r(s,a)|_{a=\mu(s)} \\
&\qquad + \gamma\int_{\mathcal{S}}\Big(p(s'|s,\mu(s))\nabla_{\theta^\mu}V^\mu(s') + \nabla_{\theta^\mu}\mu(s)\nabla_a p(s'|s,\mu(s))|_{a=\mu(s)}V^\mu(s')\Big)\mathrm{d}s'\Bigg] \\
&= \mathbb{E}_{s\sim d^\beta}\left[\nabla_{\theta^\mu}\mu(s)\nabla_a\left(r(s,a) + \gamma\int_{\mathcal{S}} p(s'|s,\mu(s))V^\mu(s')\mathrm{d}s'\right)\Big|_{a=\mu(s)} + \gamma\int_{\mathcal{S}} p(s'|s,\mu(s))\nabla_{\theta^\mu}V^\mu(s')\mathrm{d}s'\right] \\
&= \mathbb{E}_{s\sim d^\beta}\left[\nabla_{\theta^\mu}\mu(s)\nabla_a Q^\mu(s,a)|_{a=\mu(s)} + \gamma\int_{\mathcal{S}} p(s'|s,\mu(s))\nabla_{\theta^\mu}V^\mu(s')\mathrm{d}s'\right]
\end{aligned}$$

In the on-policy case, we maintain the expansion of the $V^\mu(s')$ terms as they occur (see Appendix B in [7]). In contrast, within the off-policy case, we introduce an approximation by considering only the initial term. As a result, the approximated policy gradient is formulated as (11)

### 5.1.3 DDPG Algorithm

**Algorithm 1** DDPG algorithm

Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $\mathcal{R}$
**for** episode $= 1, M$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  **for** $t = 1, T$ **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to current policy and exploration noise
    Execute action $at$ and observe reward $r_t$ and new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{R}$
    Sample random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $\mathcal{R}$
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  **end for**
**end for**

### 5.1.4 Stabilizing Learning

DDPG makes several tweaks to stabilize training with neural networks:

- **Use replay buffer**: Store transitions in a replay buffer. Sample mini-batches for training.
- **Use soft target update**: Use separate slowly-updating target networks $Q'$ and $\mu'$ to calculate the target Q values during Bellman updates. This provides more consistent targets during temporal difference learning.
- **Use batch minimization**: To address varying units and ranges in low-dimensional feature vectors, batch normalization is applied—a deep learning technique normalizing each dimension in mini-batches to maintain consistency across environments and unit scales, reducing manual scaling needs. It's used on the state input and all layers of $\mu$-network and $Q$-network before the action input, facilitating effective learning across diverse tasks with different unit types without manual adjustments. This technique also mitigates covariance shift during training.
- **Use noise for the policy**: Add temporally correlated noise to the actor policy during exploration to facilitate efficient exploration in environments with momentum. For example, use an Ornstein-Uhlenbeck process rather than white noise.

### 5.1.5 Network Specifications

| **Parameter/Specification** | **Value** |
|---|---|
| Optimizer | Adam |
| Learning Rate (Actor) | $10^{-4}$ |
| Learning Rate (Critic) | $10^{-3}$ |
| L2 Weight Decay (Q) | $10^{-2}$ |
| Discount Factor ($\gamma$) | 0.99 |
| Soft Target Update ($\tau$) | 0.001 |
| Activation Function | Rectified Non-linearity |
| Final Output Layer (Actor) | tanh |
| Hidden Layers (Low-dimensional Networks) | 400 units, 300 units |
| Actions Included | 2nd Hidden Layer of Q |
| Minibatch Size (Low-dimensional) | 64 |
| Replay Buffer Size | $10^6$ |

Table 3: Neural Network Training Parameters and Specifications

## 5.2 SAC: Soft Actor-Critic

# 6 Appendix I: Basic Statistics

- **Probability:**
  - $P(X)$ represents the probability of event $X$, ranging from 0 to 1.
  - $X'$ denotes the complement of event $X$, such that $P(X') = 1 - P(X)$.
  - $X \cup Y$ signifies the union of events $X$ and $Y$, with $P(X \cup Y) = P(X) + P(Y) - P(X \cap Y)$.
- **Probability Properties:**
  - **Marginal Probability:**
    - For joint probability distribution $P(X, Y)$, the marginal probability of $X$ is obtained by summing or integrating over all possible values of $Y$: $P(X) = \int P(X, y)dy$.
  - **Conditional Probability:**
    - $P(X|Y) = \frac{P(X \cap Y)}{P(Y)}$ denotes the conditional probability of $X$ given $Y$.
  - **Independence:**
    - Events $X$ and $Y$ are independent if $P(X \cap Y) = P(X) \cdot P(Y)$.
  - **Law of Total Probability:**
    - $P(A) = \sum_i P(A|B_i) \cdot P(B_i)$ for mutually exclusive and exhaustive events $B_1, B_2, \ldots$.
- **Expectation:**
  - $\mathbb{E}(X)$ or $\mu$ represents the expectation of random variable $X$.
    - For discrete $X$: $\mathbb{E}(X) = \sum_x x \cdot P(X = x)$.
    - For continuous $X$ with probability density function $f(x)$: $\mathbb{E}(X) = \int x \cdot f(x)dx$.
  - **Linearity of Expectation:**
    - $\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$ for random variables $X$ and $Y$, and constants $a$ and $b$.
  - **Law of the Unconscious Statistician:**
    - $\mathbb{E}(g(X)) = \sum_x g(x)P(X = x)$, where $P(X = x)$ is the probability mass function of $X$.
  - **Importance Sampling:**
    - $\mathbb{E}_{x\sim p(x)}[f(x)] = \mathbb{E}_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]$.
- **Variance:**
  - The variance of random variable $X$, denoted as $\text{Var}(X)$ or $\sigma^2$, measures the spread of $X$ around its expected value:
$$\text{Var}(X) = E[(X - E(X))^2] = E(X^2) - [E(X)]^2$$
  - **Covariance:**
    - $\text{Cov}(X, Y) = E[(X - E(X))(Y - E(Y))]$ measures the linear relationship between random variables $X$ and $Y$.
  - **Conditional Expectation:**
    - $\mathbb{E}(X|Y = y) = \sum_x x \cdot P(X = x|Y = y)$ for discrete $X$ and $Y$, and $\mathbb{E}(X|Y = y) = \int x \cdot f(x|y)dx$ for continuous $X$ and $Y$ with conditional probability density function $f(x|y)$.
  - **Law of Iterated Expectations:**
    - $\mathbb{E}(E(X|Y)) = E(X)$.

# 7 Appendix II: Maximum Entropy Principle

**Maximum Entropy Principle**

The Maximum Entropy Principle provides a framework for finding probability distributions that are maximally non-committal or, in other words, the least informative given a set of constraints. This principle is particularly useful in situations where we have limited information about a system but wish to make informed probabilistic predictions.

The Maximum entropy principle can be mathematically expressed as follows:

$$p^*(x) = \underset{p}{\operatorname{argmax}} \quad \mathcal{H}_p(x) \tag{12a}$$

$$\text{s.t.} \quad \forall k \in \{1, 2, \ldots, n\}, \quad \mathbb{E}_p[f_k(x)] = \tilde{f}_k \tag{12b}$$

$$\int p(x)\mathrm{d}x = 1 \tag{12c}$$

The solution to (12) is

$$p^*(x) = \frac{\exp\left(\sum_{k=1}^n \lambda_k f_k(x)\right)}{\int \exp\left(\sum_{k=1}^n \lambda_k f_k(x)\right) \mathrm{d}x}, \tag{13}$$

Maximum Entropy Principle Solution

To find the solution to (12), we formulate the Lagrangian associated with (12):

$$\mathcal{L}(x, p(x)) = \int p(x) \ln p(x) \mathrm{d}x - \lambda_0 \left( \int p(x)\mathrm{d}x - 1 \right) - \sum_{k=1}^n \lambda_k \left( \int p(x) f_k(x)\mathrm{d}x - \tilde{f}_k \right) \tag{14}$$

$$= \int \underbrace{p(x) \left( \ln p(x) - \lambda_0 - \sum_{k=1}^n \lambda_k f_k(x) \right) \mathrm{d}x}_{\triangleq F(x, p(x))} + \lambda_0 + \sum_{k=1}^n \lambda_k \tilde{f}_k, \tag{15}$$

where $\lambda_0$ and $\lambda_k$ are

The objective is to find the distribution $p(x)$ that makes (14) stationary. Applying the Euler-Lagrange equation we have

$$\frac{\partial F(x, p(x))}{\partial p(x)} = 0$$

Thus,

$$\ln p(x) - \lambda_0 - \sum_{k=1}^n \lambda_k f_k(x) + 1 = 0$$

$$\ln p(x) = \lambda_0 + \sum_{k=1}^n \lambda_k f_k(x) - 1$$

$$p(x) = \exp\left( \lambda_0 + \sum_{k=1}^n \lambda_k f_k(x) - 1 \right) \tag{16}$$

Next, the Lagrange multipliers $\lambda_0, \lambda_1, \ldots, \lambda_n$ is determined from constraints (12b) and (12c). Starting from (12b), we can derive $\lambda_0$ as follows.

$$\int \exp\left( \lambda_0 + \sum_{k=1}^n \lambda_k f_k(x) - 1 \right) \mathrm{d}x = 1$$

$$e^{\lambda_0 - 1} \int \exp\left( \sum_{k=1}^n \lambda_k f_k(x) - 1 \right) \mathrm{d}x = 1$$

$$\lambda_0 - 1 + \ln \int \exp\left(\sum_{k=1}^{n} \lambda_k f_k(x)\right) \mathrm{d}x = 0$$

$$\lambda_0 = 1 - \ln \int \exp\left(\sum_{k=1}^{n} \lambda_k f_k(x)\right) \mathrm{d}x \tag{17}$$

Substituting the expression for $\lambda_0$ from (17) into (16) yields the final solution for $p(x)$, where the remaining Lagrange multipliers $\lambda_1, \ldots, \lambda_n$ can be determined using (12c).

$$p(x) = \frac{\exp\left(\sum_{k=1}^{n} \lambda_k f_k(x)\right)}{\int \exp\left(\sum_{k=1}^{n} \lambda_k f_k(x)\right) \mathrm{d}x}.$$

# References


[1] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[2] D. Precup, Eligibility traces for off-policy policy evaluation, Computer Science Department Faculty Publication Series (2000) 80.

[3] S. Levine, A. Kumar, G. Tucker, J. Fu, Offline reinforcement learning: Tutorial, review, and perspectives on open problems, arXiv preprint arXiv:2005.01643 (2020).

[4] S. Levine, V. Koltun, Guided policy search, in: International conference on machine learning, PMLR, 2013, pp. 1–9.

[5] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv preprint arXiv:1506.02438 (2015).

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, nature 518 (7540) (2015) 529–533.

[7] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International conference on machine learning, Pmlr, 2014, pp. 387–395.