Week 7: Policy Gradient

Ardhito Nurhadyansah[1], Mohamad Arvin Fadriansyah[2], and Ian Suryadi Timothy H[3]

[1]2106750206
[2]2006596996
[3]2106750875

1 Generalized Policy Iteration

As the name suggest, Generalized Policy Iteration (GPI) is a generalized version of Policy Iteration. In policy iteration, there are 2 main parts: policy evaluation and policy improvement. In GPI, the generalization refers to those 2 parts, where they're computed in an inexact way. This enables us to advance from Dynamic Programming setting to Reinforcement Learning setting, where we can use function approximation and sampling to improve the agent's policy. In DP setting, the agent needs to know the state transition distribution p and the reward function r, while in RL setting, the agent doesn't need to know them.

1.1 Inexact Policy Evaluation

This part deals with the problem of estimating the value function of a policy π. For this, we can choose the following loss function to minimize:

- e_{MS} - Mean Squared Error, which measures the average squared difference between the estimated value function and the true value function:
  $$e_{MS}(\boldsymbol{w}) = \sum_{s \in \mathcal{S}} p^{\star}(s)\big(v(s) - \hat{v}_(s; \boldsymbol{w})\big)^2 \tag{1}$$
  This also called a semi-gradient method, where it uses an approximation for the true value function using BEE that involves bootstrapping as one of the component to compute the gradient of the error.
- e_{PB} - Projected Bellman Error, which measures the difference between the estimated value function and its Projected Bellman version:
  $$e_{PB}(\boldsymbol{w}) = \|(\hat{v}(\boldsymbol{w}) - \mathbb{P}\mathbb{B}\hat{v}(\boldsymbol{w}))\|^2 \tag{2}$$
  This also called a Least Square Temporal Difference (LSTD) method.

1.2 Inexact Policy Improvement

This part deals with modifying the policy π^k such that the new policy π^{k+1} is better than the previous one, i.e., $v(\pi^{k+1}) > v(\pi^k)$. In RL, this can be done with policy gradient, specifically with gradient ascent method to keep adjusting the policy to maximize its value. Remember that gradient is having role to guide below components so that the maximization of the objective function can be reached as soon as possible:

- Direction - where should the parameter update go
- Magnitude - Not only the step size parameter, the gradient is also having the role to determine how big the step that should be taken for the parameter update. The bigger the gradient norm, the more sensitive the objective function w.r.t the change of the gradient. In other words, the bigger the norm of the gradient, the bigger the step, and vice versa.

Concretely, to do the inexact policy improvement, we could:

a Find the gradient of the policy's value w.r.t the policy, i.e., $\nabla_\pi v_x(\pi^k) = \frac{\partial}{\partial \pi} v_x(\pi^k)$.
b Use the gradient ascent method to update the policy:

$$\pi^{k+1} \leftarrow \pi^k + \alpha \nabla_\pi v_x(\pi^k), \tag{3}$$

However, at the above method, we use direct policy parameterization, where the gradient is computed directly w.r.t the policy. This method is not practical to be used in RL setting, because when the state and action space is large, then the parameter needed to represent the policy is also large (each of the state-action pair needs to be represented by a separate parameter). This makes the computation infeasible, and optimization hard to be done.

As a remedy, we can use explicit policy parameterization, where we define a parameter vector $\theta \in \Theta = \mathbb{R}^{dim(\theta)}$, so that it becomes the parameter for the policy:

$$\pi(\theta) = \pi(a \mid s; \theta) \tag{4}$$

And thus, the value of the policy can be written as:

$$v(\pi(\theta)) = v(\theta) \tag{5}$$

One example: $\pi \sim \mathcal{N}(\mu = 0, \sigma^2 = \theta)$

This brings several benefits:

- We can use high-dimensional representation for the policy, e.g., using neural networks or other flexible approximators for $\pi(\theta)$ allows us to represent complex policies that can generalize well across large state spaces.
- Explicit policy parameterizations can naturally incorporate an exploration mechanism (similar in spirit to ε-greedy).
- Compatibility with gradient-based optimization: Once the policy is explicitly parameterized and differentiable, powerful optimization algorithms can be applied such as Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO).

Many of the proven empirical success, State of The Art (SOTA), and best practices in RL comes from this explicit policy parameterization method, such as PPO algorithm.

2 Policy Parameterization for Discrete Action Spaces

In reinforcement learning with discrete action spaces, we often represent the policy $\pi(a \mid s; \boldsymbol{\theta})$ using a parameterized categorical distribution. One commonly used approach is the softmax (Gibbs/Boltzmann) parameterization, where the probability of selecting an action a in state s is defined as:

$$\pi(a \mid s; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta}^\top \phi(s,a))}{\sum_{a' \in A} \exp(\boldsymbol{\theta}^\top \phi(s,a'))}, \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$$

where:

- $\phi(s,a) \in \mathbb{R}^d$ is a feature vector for state-action pair (s,a),
- $\boldsymbol{\theta} \in \mathbb{R}^d$ is the policy parameter vector,
- $\mathcal{A}$ is the set of all possible discrete actions.

This softmax function ensures that:

- Each action probability is positive: $\pi(a \mid s; \boldsymbol{\theta}) > 0$,
- The probabilities sum to one: $\sum_{a \in A} \pi(a \mid s; \boldsymbol{\theta}) = 1$.

The distribution $\pi(\cdot \mid s; \boldsymbol{\theta})$ is also known as a categorical distribution over actions, and this formulation aligns with the Gibbs distribution in statistical physics, which favors higher-scoring actions (higher $\boldsymbol{\theta}^\top \phi(s,a)$).

Alternative Interpretation: Preference-Based Parameterization:

Sutton and Barto (2018) also presents a preference-based view of softmax policy parameterization, where each action a is assigned a scalar preference $H_t(a) \in \mathbb{R}$. These preferences determine the policy via:

$$\pi_t(a) = \frac{\exp(H_t(a))}{\sum_{b \in A} \exp(H_t(b))}.$$

Only the relative differences between preferences matter. For example, adding a constant to all $H_t(a)$ values has no effect on the resulting action probabilities. This highlights the invariance property of the softmax formulation. Initially, all preferences are often set equally (e.g., $H_1(a) = 0$) to induce uniform exploration.

Connection to the Logistic (Sigmoid) Function

When there are exactly two actions, say a_1 and a_2, the softmax formula

$$\pi(a_i) = \frac{\exp\bigl(H(a_i)\bigr)}{\exp\bigl(H(a_1)\bigr) + \exp\bigl(H(a_2)\bigr)}, \quad i = 1, 2,$$

reduces to the familiar sigmoid form. For a_1:

$$\pi(a_1) = \frac{\exp\bigl(H(a_1)\bigr)}{\exp\bigl(H(a_1)\bigr) + \exp\bigl(H(a_2)\bigr)} = \frac{1}{1 + \exp\bigl(H(a_2) - H(a_1)\bigr)} = \frac{1}{1 + \exp\bigl(-(H(a_1) - H(a_2))\bigr)} = \sigma\bigl(H(a_1) - H(a_2)\bigr)$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the logistic sigmoid.

3 Gradient Ascent

After specifying the explicit policy parameterization, we can now use the gradient ascent method to update the policy:

$$\theta^{k+1} \leftarrow \theta^k + \alpha \nabla_\theta v_x(\theta)\Big|_{\theta = \theta^k} \tag{6}$$

Where:

- k is the iteration index
- θ^k is the parameter vector at iteration k.
- α is the (positive) step size or learning rate. This usually becomes hyperparameter.
- $\nabla_\theta v_x(\theta)$ is the gradient of the value function with respect to the parameter vector θ under some optimality criterion x (e.g., discounted return, average reward, total reward)
- θ^{k+1} is the updated parameter vector after applying the gradient ascent step

Now, the big question to get everything to be working is on how to find the $\nabla_\theta v_x(\theta)$. And this is going to be discussed in the next section.

4 Policy Gradient Theorem

As discussed in previous section, the most important thing to make the gradient ascent method works is to find the $\nabla_\theta v_x(\theta)$. This is where the policy gradient theorem comes in.

First, let's set the optimality criterion x to be the average reward or gain. Then, we can write the value function as:

$$v_g(\theta) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p^\star_{\pi(\theta)}(s) \cdot \pi(a \mid s; \theta) \cdot r(s, a) \tag{7}$$

Introducing the gradient operator ∇_θ at both sides:

$$\nabla_\theta v_g(\theta) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p^\star_{\pi(\theta)}(s) \cdot \pi(a \mid s; \theta) \cdot r(s, a) \tag{8}$$

Exchanging the order of sum and gradient, applying gradient to θ-dependent terms:

$$\nabla_\theta v_g(\theta) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} r(s, a) \cdot \nabla_\theta \Bigl(p^\star_{\pi(\theta)}(s) \cdot \pi(a \mid s; \theta)\Bigr) \tag{9}$$

Using the product rule:

$$\nabla_\theta v_g(\theta) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} r(s,a) \cdot \Big(\nabla_\theta p^\star_{\pi(\theta)}(s) \cdot \pi(a \mid s; \theta) + p^\star_{\pi(\theta)}(s) \cdot \nabla_\theta \pi(a \mid s; \theta) \Big) \tag{10}$$

The above equation is not sampling-friendly, because it involves the gradient of the state distribution $p^\star_{\pi(\theta)}(s)$, which is not easy to compute.

Now, recall that the score function identity is defined as:

$$\nabla_\theta \log \pi(a \mid s; \boldsymbol{\theta}) = \frac{\nabla \pi(a \mid s; \boldsymbol{\theta})}{\pi(a \mid s; \boldsymbol{\theta})} \tag{11}$$

Using the above identity, we can rewrite the gradient of the policy as:

$$\nabla_\theta \pi(a \mid s; \boldsymbol{\theta}) = \pi(a \mid s; \boldsymbol{\theta}) \cdot \nabla_\theta \log \pi(a \mid s; \boldsymbol{\theta}) \tag{12}$$

Utilizing the score function into (10):

$$\nabla_\theta v_g(\theta) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} r(s,a) \cdot \Big(p^\star_{\pi(\theta)}(s) \cdot \nabla_\theta \log p^\star_{\pi(\theta)}(s) \cdot \pi(a \mid s; \theta) + p^\star_{\pi(\theta)}(s) \cdot \pi(a \mid s; \boldsymbol{\theta}) \cdot \nabla_\theta \log \pi(a \mid s; \boldsymbol{\theta}) \Big) \tag{13}$$

$$= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p^\star_{\pi(\theta)}(s) \cdot \pi(a \mid s; \boldsymbol{\theta}) \cdot r(s,a) \cdot \Big(\nabla_\theta \log p^\star_{\pi(\theta)}(s) + \nabla_\theta \log \pi(a \mid s; \boldsymbol{\theta}) \Big) \tag{14}$$

Note that $p^\star_{\pi(\theta)}(s)$ appearing as a multiplicative term is not problematic as it can be handled under expectation. However, $\nabla_\theta \log p^\star_{\pi(\theta)}(s)$ is unknown and difficult to compute. While this form is sampling-friendly, in the RL setting $p^\star_{\pi(\theta)}(s)$ itself is unknown to the agent.

To get rid of the above problem, Sutton et al. (1999) proposed that we can encode the $\nabla_\theta \log p_\pi(\theta)^\star(s)$ and $r(s,a)$ into $q^\pi_b(s,a)$:

$$\nabla_\theta v_g(\theta) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p^\star_{\pi(\theta)}(s) \cdot \pi(a \mid s; \boldsymbol{\theta}) \cdot q^\pi_b(s,a) \cdot \nabla_\theta \log \pi(a \mid s; \boldsymbol{\theta}) \tag{15}$$

$$= \mathbb{E}_{S \sim p^\star_{\pi(\theta)}(.), A \sim \pi(.\mid S; \boldsymbol{\theta})} [q^\pi_b(S,A) \cdot \nabla_\theta \log \pi(A \mid S; \boldsymbol{\theta})] \tag{16}$$

Remember that expectation is equivalent to mean μ, thus we can get an unbiased approximation of the gradient by using sample mean $\hat{\mu}$ (which is also a random variable) and the expectation of it is equal to the mean.

That $p^\star_{\pi(\theta)}$ is unknown but appears under expectation. Hence, is practice, we simply sample states from it. But it just exist when $t >= t_{mix}$. Before that, we sample from $p^t_{\pi(\theta)}$, which result in a biased approximation and thus called Semi Stochastic Gradient Ascent. When the sampling is obtained from the true distribution $p^\star_{\pi(\theta)}$, the approximation is unbiased, thus we can call it as Stochastic Gradient Ascent.

5 Discounted Reward and the Policy Gradient Theorem

In reinforcement learning, when working with continuing tasks, we often aim to maximize the expected *discounted return*. This leads to defining the performance objective as:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right] = \mathbf{P}^\gamma_\pi \mathbf{r}_\pi,$$

where:

- $\boldsymbol{\theta} \in \mathbb{R}^d$ is the policy parameter,

- $\gamma \in [0, 1)$ is the discount factor,
- $\pi_{\boldsymbol{\theta}}$ is the parameterized policy,
- $\mathbf{P}_\pi^\gamma = \lim_{t_{max}\to\infty} \sum_{t=0}^{t_{max}-1} \big(\gamma\,\mathbf{P}_\pi\big)^t = (\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}$ is the state transition distribution matrix under policy π and discount factor γ,
- $\mathbf{r}_\pi \in \mathbb{R}^{|\mathcal{S}|}$ is the reward vector under policy π.

5.1 Why $\mathbf{P}_\pi^\gamma$ Is Not a Proper Stochastic Matrix

From the definition

$$\mathbf{P}_\pi^\gamma(s \mid s_0) \;=\; \sum_{t=0}^{\infty} \gamma^t\, P_\pi^t(s \mid s_0),$$

let us check whether each column $s_0 \mapsto \mathbf{P}_\pi^\gamma(\cdot \mid s_0)$ sums to one.

$$\sum_{s\in\mathcal{S}} \mathbf{P}_\pi^\gamma(s \mid s_0) = \sum_{s\in\mathcal{S}} \sum_{t=0}^{\infty} \gamma^t\, P_\pi^t(s \mid s_0) = \sum_{t=0}^{\infty} \gamma^t \underbrace{\sum_{s\in\mathcal{S}} P_\pi^t(s \mid s_0)}_{=1} = \sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma} \;\neq\; 1.$$

Hence,

$$\sum_{s} \mathbf{P}_\pi^\gamma(s \mid s_0) = \frac{1}{1-\gamma},$$

so $\mathbf{P}_\pi^\gamma$ does not satisfy the column-sum-to-one property of a stochastic matrix.

5.2 Constructing a Discounted Occupancy Distribution

To get rid of the previous problem where $\mathbf{P}_\pi^\gamma$ is not a proper distribution, we must renormalize:

$$d_\pi(s \mid s_0) \;=\; (1-\gamma)\,\mathbf{P}_\pi^\gamma(s \mid s_0) \quad\Longrightarrow\quad \sum_{s} d_\pi(s \mid s_0) = 1.$$

Here $d_\pi(s \mid s_0)$ is the (normalized) *discounted state-occupancy measure*, often denoted simply $d_\pi(s)$ when averaged over an initial distribution.

Interpretation:

- $\mathbf{P}_\pi^\gamma(s \mid s_0)$ counts (in expectation) the *discounted* number of visits to s starting from s_0.
- Multiplying by $(1-\gamma)$ turns those counts into a valid probability distribution over states.
- In policy-gradient methods, one often uses $d_\pi(s)$ to weight gradient estimates by how frequently states are encountered under π.

5.3 From the Unnormalized Resolvent to a Proper Sampling Distribution

We begin with the "raw" (unnormalized) form of the discounted policy gradient:

$$\nabla_{\boldsymbol{\theta}} V_\gamma(\pi, s_0) = \sum_{s\in\mathcal{S}} \underbrace{\mathbf{P}_\pi^\gamma(s \mid s_0)}_{\text{not a dist.}} \sum_{a\in\mathcal{A}} q_\pi(s,a)\,\nabla_{\boldsymbol{\theta}} \log \pi(a \mid s; \boldsymbol{\theta}).$$

Since $\mathbf{P}_\pi^\gamma$ does not sum to one over s, it is not sampling-friendly. We therefore multiply both sides by $(1-\gamma)$, yielding

$$(1-\gamma)\,\nabla_{\boldsymbol{\theta}} V_\gamma(\pi, s_0) = \sum_{s\in\mathcal{S}} \underbrace{(1-\gamma)\,\mathbf{P}_\pi^\gamma(s \mid s_0)}_{d_\pi(s|s_0)} \sum_{a\in\mathcal{A}} q_\pi(s,a)\,\nabla_{\boldsymbol{\theta}} \log \pi(a \mid s; \boldsymbol{\theta}).$$

By definition,

$$d_\pi(s \mid s_0) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P_\pi^t(s \mid s_0), \qquad \sum_{s} d_\pi(s \mid s_0) = 1,$$

so d_π is a proper distribution over states. The price we pay is a scaled gradient, $(1-\gamma)\,\nabla_{\boldsymbol{\theta}} V_\gamma(\pi, s_0)$.

Note that if we still use p_π^γ instead of d_π, then the above equation will not be sampling-friendly, because p_π^γ is not a proper distribution.

5.4 How to Sample a State from the Discounted Distribution

Let $d_\pi = \widetilde{P}_\pi^\gamma \propto (1-\gamma)P_\pi^\gamma$ denote the normalized discounted state distribution. To sample a single state $S \sim \widetilde{P}_\pi^\gamma$, follow this procedure:

1. Sample a length $L \sim \mathrm{Geo}(1-\gamma)$, representing the number of steps to run the episode. This reflects the discounted weighting of time steps.
2. Run an episode (trajectory) from $t=0$ to $t=L-1$ using the current policy $\pi_{\boldsymbol{\theta}}$.
3. Let the sampled state be:
$$S_{L-1}.$$
All previous states can be discarded. However, in practice, people often use *all* states $\{S_0, \ldots, S_{L-1}\}$ to construct an unbiased gradient estimate more efficiently.

This method allows sampling from the discounted state distribution using a simple rejection-free procedure based on the geometric distribution.

5.5 Dependence of the Optimal Policy on the Initial State Distribution

Consider a Markov Decision Process (MDP) in discounted reward optimality criterion, which is defined by the tuple
$$\big(\mathcal{S},\, \mathcal{A}, \mathcal{T},\, P,\, r,\, \gamma,\, p^0\big),$$
where

- $\mathcal{S}$ is the finite set of states,
- $\mathcal{A}$ is the finite set of actions,
- $\mathcal{T}$ is the time set,
- $P(s' \mid s, a)$ is the transition kernel,
- $r(s,a)$ (or $r(s)$) is the (possibly discounted) reward function,
- $\gamma \in [0,1)$ is the discount factor,
- p^0 is the distribution over the initial state $S_0 \sim p^0$.

5.6 State–Value and Action–Value Functions

For any (possibly stochastic) policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$, define:
$$v^\pi(s) \;=\; \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t\, r(S_t, A_t) \;\Big|\; S_0 = s,\, \pi\Big],$$
$$q^\pi(s,a) \;=\; \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t\, r(S_t, A_t) \;\Big|\; S_0 = s,\, A_0 = a,\, \pi\Big].$$

5.7 Performance Objective

We measure policy performance by the expected start–state value under p^0:
$$J(\pi) \;=\; \mathbb{E}_{S_0 \sim p^0}\big[v^\pi(S_0)\big] \;=\; \sum_{s \in \mathcal{S}} p^0(s)\, v^\pi(s).$$
The *optimal policy* is then
$$\pi^*(p^0) \;=\; \arg\max_\pi \; J(\pi) \;=\; \arg\max_\pi \; \mathbb{E}_{S_0 \sim p^0}\big[v^\pi(S_0)\big].$$
Key point: the optimizer $\pi^*(p^0)$ generally *depends* on the choice of p^0.

5.8 State–Wise vs. Distribution–Averaged Optimality

1. State–Wise Optimality. For each fixed initial state s_0, one can define a *state–specific* optimal policy
$$\pi^*_{s_0} \;=\; \arg\max_\pi \; v^\pi(s_0).$$
This policy maximizes return when *starting* exactly in s_0. However, it may perform poorly if the true start state differs.

2. Distribution–Averaged Optimality. When the start state is sampled from p^0, the relevant criterion is

$$\bar{\pi} \;=\; \arg\max_{\pi} \; \mathbb{E}_{S_0 \sim p^0}\big[v^{\pi}(S_0)\big].$$

This $\bar{\pi}$ may sacrifice performance in unlikely start states to improve performance where p^0 places more mass.

5.9 Example: Two Initial Distributions

Let $\mathcal{S} = \{s_0, s_1, s_2, s_3\}$. Consider two different initial distributions:

$$p^0_{(1)} = [1 \quad 1 \quad 0 \quad 0]^{\mathsf{T}}, \quad p^0_{(2)} = [0 \quad 1 \quad 1 \quad 1]^{\mathsf{T}}.$$

- Under $p^0_{(1)}$, only s_0, s_1 ever occur; the optimal policy $\pi^*(p^0_{(1)})$ will focus on maximizing $v^{\pi}(s_0)$ and $v^{\pi}(s_1)$.
- Under $p^0_{(2)}$, the process starts in s_1, s_2, s_3; hence $\pi^*(p^0_{(2)})$ may differ substantially, since it can ignore s_0 entirely.

5.10 Uniformly Optimal Policy

A policy is *uniformly optimal* (or *non–distributional*) if it maximizes $v^{\pi}(s)$ *for every* $s \in \mathcal{S}$:

$$\pi^{\text{uni}} \;=\; \arg\max_{\pi} \; \min_{s \in \mathcal{S}} v^{\pi}(s).$$

Such a policy does not depend on p^0, but need not exist unless the MDP admits a single policy that is simultaneously optimal in all start states.

The optimal policy for an MDP in discounted reward optimality criterion is *not* unique in general and *varies* with the initial state distribution p^0, unless a uniformly optimal policy exists. Hence, when designing or evaluating algorithms, one must specify (or be aware of) the assumed p^0 under which optimality is judged.

6 Policy Gradient: Average vs Discounted

In comparing average-reward and discounted-reward policy-gradient methods, two key differences are:

6.1 Sampling Distributions

Average-reward:

$$S \sim d^*_{\pi} \quad A \sim \pi(\cdot \mid S).$$

Discounted-reward:

$$T_{\max} \sim \text{Geo}(1-\gamma), \quad S \sim P^{\gamma}_{\pi}(\cdot \mid s_0), \quad A \sim \pi(\cdot \mid S).$$

6.2 Dependence on the Initial State

- **Average-reward optimal policy:**

$$\pi^{\star}_g = \arg\max_{\pi} \; v_g(\pi)$$

  is *uniformly optimal* across all start states in a unichain MDP — it does not depend on any particular initial distribution.
- **Discounted-reward optimal policy:**

$$\pi^{\star}_{\gamma} = \arg\max_{\pi} \; \mathbb{E}_{S_0 \sim p^0}\left[V_{\gamma}(\pi, S_0)\right]$$

  generally *depends* on the chosen initial-state distribution p^0. Different p^0 can yield different optimal π^*_{γ}.

7 High Variance in Policy Gradient Estimation

7.1 Issue

The sample-based estimator of the policy gradient is unbiased once the underlying Markov chain has sufficiently mixed. However, it often suffers from high variance because it draws randomness from two independent sources (random variables):

- the initial state distribution $p_\pi^\star$, and
- the stochastic nature of the policy π.

Each of these contributes to the overall variance, which can slow or destabilize learning.

7.2 Variance Reduction Techniques

To reduce variance without introducing bias, two complementary approaches are widely used:

1. **Baseline / Control Variate.** Add a baseline function that depends only on the state (not on the action). Subtracting this baseline from the action-value estimate preserves unbiasedness while lowering variance. A common choice for the baseline is the state-value function, leading to the notion of an advantage function.
2. **Actor–Critic Methods.** Introduce a parametric *critic* that learns a value-function approximation online (e.g. via temporal-difference updates). The critic's estimate serves as an adaptive baseline or advantage for the actor, further stabilizing and reducing the variance of the policy update.

By combining a state-dependent baseline with an actor–critic architecture, one can maintain unbiased gradient estimates while dramatically reducing estimator variance, thereby improving convergence speed and stability.

7.3 Baseline as a Control Variate and TD–Based Advantage Estimation

Building on the basic policy-gradient theorem, we introduce a state-dependent baseline v_b and show how to replace the true advantage by a one-step TD error. The handwritten notes give the following formulas:

1. Policy-Gradient with a Baseline

$$\nabla v_g(\theta) = \mathbb{E}\big[\{\, q_b^\pi(S, A) \;-\; v_b(S)\}\, \nabla_\theta \log \pi_\theta(A \mid S)\big] = \mathbb{E}\big[\delta_v(S, A, S')\, \nabla_\theta \log \pi_\theta(A \mid S)\big].$$

2. Definition of the TD Error

$$\delta_v(s, a, s') = \mathbb{E}_{s' \sim P(\cdot|s,a)}\big[r(s, a) \;-\; v_b \;+\; v_b(s') \;-\; v_b(s) \;\big|\; s, a\big].$$

3. Single-Sample Approximation

$$\nabla v_g(\theta) \;\approx\; \delta_v(s, a, s')\, \nabla_\theta \log \pi_\theta(a \mid s) \quad \text{(using one sample of } (s, a, s')\text{)}.$$

Explanation

- The second equality replaces the true advantage $q^\pi(S, A) - v_b(S)$ by the TD-based quantity $\delta_v(S, A, S')$, which has the same expectation under the transition matrix P.
- Finally, we approximate the full expectation by a single sampled transition (s, a, s'), yielding a low-variance stochastic gradient step in which the critic's baseline and the actor's gradient are combined in one update.

7.4 Actor–Critic: Parametric Value Approximation

Actor–Critic (AC) methods further reduce variance by introducing a *parametric value approximator* (the *critic*), at the cost of a small bias–variance trade-off. Let $w \in \mathcal{W}$ denote the critic parameters.

Critic as a State–Value Baseline One can form a parametric baseline

$$\hat{v}_b(s; w),$$

which is then used inside the TD-error

$$\delta_v(s, a, s') \;=\; r(s, a) \;+\; \gamma\, \hat{v}_b(s'; w) \;-\; \hat{v}_b(s; w)$$

to approximate the advantage in the policy-gradient update

$$\nabla v_g(\theta) \;\approx\; \mathbb{E}\big[\delta_v(S, A, S')\, \nabla_\theta \log \pi_\theta(A \mid S)\big].$$

Here we are *combining* (1) the baseline control-variate trick and (2) the actor–critic TD update.

Critic as an Action–Value Approximator Alternatively, one may learn a parametric action-value critic

$$\hat{q}_b(s, a; w),$$

and plug it directly into the original policy-gradient theorem:

$$\nabla v_g(\theta) = \mathbb{E}\big[\{\hat{q}_b(S, A; w) \;-\; v_b(S)\}\, \nabla_\theta \log \pi_\theta(A \mid S)\big] \;\approx\; \mathbb{E}\big[\hat{q}_b(S, A; w)\, \nabla_\theta \log \pi_\theta(A \mid S)\big].$$

In this form we apply only the actor–critic mechanism (no separate baseline subtraction).

8 Compatible Action Value Approximation and Natural Policy Gradient

We now show in detail how a *compatible* parametric critic yields both:

- an unbiased replacement for the true action-value in the policy gradient, and
- the *natural* policy gradient as a by-product and it reduces the problem to regression.

1. Definition of the Compatible Critic

Let the critic be parameterized by $w \in \mathcal{W}$ as:

$$\hat{q}^\pi_b(s, a; w) \doteq w^\top f_\theta(s, a),$$

where the feature vector is defined as:

$$f_\theta(s, a) \doteq \nabla_\theta \log \pi_\theta(a \mid s),$$

which satisfies the *compatibility condition*, namely:

$$\nabla_w \hat{q}^\pi_b(s, a; w) = f_\theta(s, a).$$

This ensures that the critic is *linear in the parameters* w, with features aligned to the policy gradient.

2. Mean-Squared Error (MSE) Objective

We fit w by minimizing the MSE between the approximated and true action-values:

$$e^q_{\text{MS}}(w) \doteq \mathbb{E}\Big[\big(\hat{q}^\pi_b(S, A; w) - q^\pi_b(S, A)\big)^2\Big].$$

The optimal w^* satisfies the stationary condition:

$$\nabla_w e^q_{\text{MS}}(w^*) = 0,$$

which implies:

$$\mathbb{E}[(\hat{q}^\pi_b(S, A; w^*) - q^\pi_b(S, A)) f_\theta(S, A)] = 0.$$

Thus, the regression error is orthogonal to the feature vector f_θ.

3. Benefit 1: Unbiased Advantage Replacement

From the policy gradient theorem:

$$\nabla_\theta v_g(\theta) = \mathbb{E}[q^\pi_b(S, A)\, f_\theta(S, A)],$$

and using the orthogonality condition:

$$\mathbb{E}[(\hat{q}^\pi_b(S, A; w^*) - q^\pi_b(S, A)) f_\theta(S, A)] = 0,$$

we can write:

$$\nabla_\theta v_g(\theta) = \mathbb{E}[\hat{q}^\pi_b(S, A; w^*) f_\theta(S, A)].$$

That is, we can use $\hat{q}^\pi_b(\cdot; w^*)$ directly in the gradient expression without introducing bias.

4. Benefit 2: Emergence of the Natural Gradient That Reduces The Problem To Regression

We observe that:

$$\mathbb{E}[\hat{q}_b^{\pi}(S, A; w^*) f_\theta(S, A)] = \mathbb{E}\big[{w^*}^\top f_\theta(S, A) f_\theta(S, A)\big] = \Phi w^*,$$

where:

$$\Phi \doteq \mathbb{E}\big[f_\theta(S, A) f_\theta(S, A)^\top\big]$$

is the *Fisher information matrix* of the policy.

Thus:

$$\Phi w^* = \nabla_\theta v_g(\theta) \quad \Rightarrow \quad w^* = \Phi^{-1} \nabla_\theta v_g(\theta).$$

This reveals that w^*, learned from regression, is equal to the natural gradient vector. In practice, this leads to the following actor update:

$$\theta \leftarrow \theta + \alpha\, w^*, \quad \text{or equivalently} \quad \theta \leftarrow \theta + \alpha\, \Phi^{-1} \nabla_\theta v_g(\theta),$$

where w^* serves as the preconditioned gradient direction and Φ^{-1} acts as a conditioning matrix.

5. Actor–Critic Update Summary

An actor–critic algorithm using compatible function approximation proceeds as follows:

1. **Critic step:** Regress $\hat{q}_b^{\pi}(s, a; w)$ toward the true $q_b^{\pi}(s, a)$, minimizing $e^{q}_{\text{MS}}(w)$, and obtain w^*.
2. **Actor step:** Update the policy using:

$$\theta \leftarrow \theta + \alpha\, w^*.$$

Additional Notes

- The actor update

$$\theta \;\leftarrow\; \theta + \alpha\, w^*$$

  is *agnostic to the specific policy parameterization*, because w^* comes from a simple linear regression problem on the critic.
- In general one may write a preconditioned gradient step as

$$\theta \;\leftarrow\; \theta + \alpha\, C\, \nabla_\theta v_g(\theta),$$

  where C is any symmetric positive-definite matrix.
  - For *vanilla* policy gradient, $C = I$.
  - Here, by compatible function approximation, $C = \Phi^{-1}$, so that

$$w^* = \Phi^{-1}\, \nabla_\theta v_g(\theta).$$

- Using $C = \Phi^{-1}$ (the *natural gradient*) yields **faster convergence** in parameter space and **lower sample complexity** than the unconditioned update $C = I$.

9 Citations and References

References

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press. (page 2)

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12. (page 4)