Week 7: Policy Gradient

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

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

1 Policy Improvement in an Inexact Way

In reinforcement learning (RL), the ultimate goal is to find a policy π that maximizes the agent's reward, such as the discounted sum of rewards, average reward per time step, or total reward. A common approach is to iteratively improve an existing policy π^k to obtain a new policy π^{k+1} such that

$$v(\pi^{k+1}) \;\geq\; v(\pi^k),$$

where $v(\pi)$ denotes the value (under some optimality criterion) of policy π.

In many practical scenarios, we do not fully solve for an optimal policy at each iteration; instead, we perform *inexact* or partial improvements. As long as these updates monotonically increase or maintain the policy value, the overall process can converge to a better policy over time.

1.1 Policy Gradient Methods

In RL, one popular realization of inexact policy improvement is via *policy gradient methods*. The idea is to perform gradient *ascent* on the objective function $v_x(\pi)$ where x indicates the chosen optimality criterion (e.g., discounted return, average reward, total reward, or bias). Concretely, we seek

$$\nabla_\pi \, v_x(\pi) \quad \text{(or its approximation)},$$

and then update the policy in the direction of this gradient to increase performance:

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

where α is a step size (learning rate).

A direct application of this concept, called *direct policy parameterization*, attempts to compute partial derivatives of $v_x(\pi)$ with respect to the *policy* π itself. However, this approach can be difficult to use in practice because:

- Policies may not have a simple, differentiable form if they are defined implicitly (e.g., large tables or non-differentiable structures).
- Directly working in the space of all possible policies can be extremely high-dimensional and unstructured, making optimization unwieldy.

These challenges motivate an alternative known as *explicit policy parameterization*.

2 Motivation for Explicit Policy Parameterization

To address the difficulties of direct policy parameterization, we introduce a parameter vector

$$\theta \;\in\; \Theta \subseteq \mathbb{R}^d,$$

where d is the dimension of the parameter space. This parameter vector defines a *family* of policies:

$$\pi_\theta(a \mid s),$$

so that the value of the policy can be written as

$$v(\pi_\theta) \;=\; v(\theta).$$

By focusing on θ, we transform the problem of improving π into the (often more tractable) problem of optimizing θ via gradient ascent:

$$\theta^{k+1} \;\leftarrow\; \theta^k \;+\; \alpha \nabla_\theta \, v(\theta^k).$$

This shift from direct policy parameterization to exact policy parameterization has several advantages:

1. High-Dimensional Function Approximation: Using neural networks or other flexible approximators for π_θ allows us to represent complex policies that can generalize well across large state spaces.
2. Systematic Exploration Control: Stochastic parameterizations can naturally incorporate an exploration mechanism (similar in spirit to ε-greedy), enabling more principled control over the exploration-exploitation trade-off.
3. Compatibility with Gradient-Based Optimization: Once the policy is explicitly parameterized and differentiable, powerful optimization algorithms can be applied. Methods such as Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), and other variants can be readily used.
4. Empirical Success in State-of-the-Art (SOTA) RL: Many of the best-performing RL algorithms in continuous control and high-dimensional settings rely on explicit policy parameterization. These algorithms have shown remarkable success in benchmarks such as robotic control tasks and complex games.

3 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:

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

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 parameter vector,
- $\mathscr{A}$ 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 (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 Logistic Function. In the case of two actions a_1 and a_2, this softmax reduces to a logistic (sigmoid) function:

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

which is widely used in statistics and neural networks. This emphasizes the connection between policy gradients and logistic regression models.

Gradient of the Softmax Policy. When applying policy gradient methods, we need the gradient of the log-policy:

$$\nabla_{\boldsymbol{\theta}} \log \pi(a \mid s; \boldsymbol{\theta}) = \phi(s,a) - \sum_{a' \in A} \pi(a' \mid s; \boldsymbol{\theta}) \phi(s,a').$$

This expression follows from the quotient rule and is essential for computing the policy gradient in REINFORCE and actor-critic methods.

Interpretation. This softmax parameterization encourages exploration: even suboptimal actions have non-zero probability of being selected. The scale of $\boldsymbol{\theta}^\top \phi(s,a)$ can influence how deterministic or stochastic the policy behaves.

4 Gradient Ascent for Policy Improvement

As mentioned in the previous chapters, one of the way to achieve inexact policy improvement is via *gradient ascent*. Concretely, if $v(\theta)$ denotes the performance measure (e.g., expected return) of a parameterized policy π_θ, then a gradient ascent step can be written as:

$$\theta \;\leftarrow\; \theta \;+\; \alpha \, \nabla_\theta \, v(\theta), \tag{1}$$

where α is a step-size (learning rate) and $\nabla_\theta v(\theta)$ is the gradient of the performance measure with respect to the policy parameters θ. By moving in the direction of the gradient, we *increase* $v(\theta)$, thus improving the policy in an inexact yet systematic manner.

4.1 Policy Gradient Formula

A key question is how to compute $\nabla_\theta \, v(\theta)$ when the policy $\pi_\theta(a \mid s)$ is stochastic. Under certain regularity conditions, the *policy gradient theorem* states:

$$\nabla_\theta \, v(\theta) = \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) \sum_{a \in \mathcal{A}} q^{\pi_\theta}(s,a) \, \nabla_\theta \, \pi_\theta(a \mid s) \tag{2}$$

$$= \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) \sum_{a \in \mathcal{A}} q^{\pi_\theta}(s,a) \, \pi_\theta(a \mid s) \, \nabla_\theta \log \pi_\theta(a \mid s), \tag{3}$$

where:

- $d^{\pi_\theta}(s)$ is the (stationary) distribution of states when following policy π_θ.
- $q^{\pi_\theta}(s,a)$ is the action-value function under π_θ, i.e., the expected return starting from state s, taking action a, and thereafter following π_θ.
- $\log \pi_\theta(a \mid s)$ is used to transform $\nabla_\theta \, \pi_\theta(a \mid s)$ into a form more amenable to sampling.

In words, to compute the gradient of the policy performance, we weight each action a in state s by both its likelihood under π_θ and its corresponding value $q^{\pi_\theta}(s,a)$.

4.2 Sampling-Based (Unbiased) Estimation

Directly summing over all states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ is often intractable, especially in large or continuous spaces. Instead, we typically approximate (3) via samples from the current policy π_θ. That is, we collect trajectories $\{(s_t, a_t, r_t)\}$ by interacting with the environment using π_θ, and then estimate:

$$\nabla_\theta \, v(\theta) \;\approx\; \frac{1}{N} \sum_{i=1}^{N} \Big[q^{\pi_\theta}\big(s_i, a_i\big) \, \nabla_\theta \log \pi_\theta\big(a_i \mid s_i\big) \Big], \tag{4}$$

where N is the number of sampled state-action pairs (or episodes).

If $q^{\pi_\theta}(s_i, a_i)$ is itself replaced by an unbiased return estimate (e.g., Monte Carlo returns, bootstrapped estimates, or advantage functions), then (4) is an *unbiased* stochastic gradient of $v(\theta)$. The entire procedure is thus referred to as *stochastic gradient ascent (SGA)*.

4.3 Biased vs. Semi-Stochastic Approaches

In practice, some approximations introduce bias. For instance, $q^{\pi_\theta}(s, a)$ may be approximated by a learned critic or truncated returns. Moreover, during learning, θ may change faster than we can gather fully on-policy samples. This mismatch between the policy used to gather data and the updated policy parameters can create a *semi-stochastic* gradient approach. While theoretically these approximations may introduce some bias, in many real-world tasks they are sufficiently accurate to yield strong performance and stable learning.

4.4 Putting It All Together

The gradient ascent update rule for inexact policy improvement can be summarized as:

$$\theta_{k+1} \;=\; \theta_k \;+\; \alpha \, \widehat{\nabla_\theta v(\theta_k)}, \tag{5}$$

where $\widehat{\nabla_\theta v(\theta_k)}$ is an approximation of the true gradient, computed via sampled trajectories or batches of experience. This approach has the following key characteristics:

- Inexact but Monotonically Improving: Each update step aims to increase $v(\theta)$, though not necessarily by an exact or optimal amount.
- Unbiased or Semi-Unbiased Estimates: Depending on how $q^{\pi_\theta}(s, a)$ is estimated, the gradient update may be unbiased (e.g., Monte Carlo returns) or partially biased (e.g., bootstrapped estimates, off-policy data).
- Computational Feasibility: Sampling-based methods are often the only viable approach in high-dimensional or continuous RL problems.
- Compatibility with Practical Hyperparameters: Even if the procedure is not strictly on-policy or if there is some mismatch in distributions, these methods still tend to work well in practice, especially with appropriate hyperparameter tuning.

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:

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

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,
- $\mathbf{P}_\pi^* = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$ is the discounted resolvent (inverse Bellman operator),
- $\mathbf{r}_\pi \in \mathbb{R}^{|\mathcal{S}|}$ is the reward vector under policy π.

However, the distribution over states under this discounted formulation is no longer a proper probability distribution. Instead, it is a weighted occupancy distribution that reflects how often states are visited, discounted over time. Formally, this state distribution is:

$$\mu_\gamma(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr(S_t = s),$$

which corresponds to a *geometric distribution* over state visitation (trial until first success). This reflects that recent states are weighted more heavily than distant ones.

5.1 How to Sample a State from the Discounted Distribution

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

1. Sample a length $L \sim \text{Geo}(1-\gamma)$, 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 $\{S_0, \dots, S_{L-1}\}$ 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.

5.2 Gradient of the Discounted Performance

When differentiating the performance objective with respect to parameters $\boldsymbol{\theta}$, a challenge arises:

- The derivative of the state distribution $\mu_\gamma(s)$ with respect to $\boldsymbol{\theta}$ is unknown.
- This term is not sampling friendly, which makes gradient estimation difficult.

To illustrate the problem, consider the gradient of the value function from a start state s_0:

$$\nabla_{\boldsymbol{\theta}} v^{\pi}_{\gamma}(s_0) = \sum_{s\in\mathcal{S}}(\mathbf{P}^{\gamma}_{\pi})(s \mid s_0)\sum_{a\in\mathcal{A}} q^{\pi}_{\gamma}(s,a)\nabla_{\boldsymbol{\theta}}\log\pi(a \mid s;\boldsymbol{\theta}).$$

This formulation shows two key difficulties:

- The discounted transition matrix $\mathbf{P}^{\gamma}_{\pi}$ is not a proper probability distribution over states because its rows do not sum to 1.
- The dependency on future state visitation makes $\nabla_{\boldsymbol{\theta}} v^{\pi}_{\gamma}(s_0)$ nontrivial to estimate via sampling, since it relies on **off-policy-like expectations** over future trajectories.

Despite this, we can still derive a useful result. The **Policy Gradient Theorem** states:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu_\gamma(s)\sum_a \nabla_{\boldsymbol{\theta}}\pi(a|s;\boldsymbol{\theta})q^{\pi}(s,a),$$

where the gradient does *not* involve the derivative of the state distribution $\mu_\gamma(s)$, and the proportionality constant depends on γ.

This result allows us to construct stochastic estimates of the policy gradient using sample trajectories.

5.3 Sampling-Friendly Estimation

Using the identity:

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

we can rewrite the gradient as:

$$\nabla J(\boldsymbol{\theta}) \propto \mathbb{E}_{\pi}[q^{\pi}(s,a)\nabla_{\boldsymbol{\theta}}\log\pi(a \mid s;\boldsymbol{\theta})],$$

which forms the basis of REINFORCE and other policy gradient algorithms.

5.4 REINFORCE with Discounted Return

In practice, for episodic tasks, we define the update rule as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi(A_t \mid S_t; \boldsymbol{\theta}_t),$$

where $G_t = \sum_{k=t}^{T} \gamma^{k-t} R_{k+1}$ is the sampled discounted return. This form emphasizes the time-weighted contribution of each reward.

5.5 Geometric Distribution Intuition

The discounted state distribution $\mu_\gamma(s)$ resembles a geometric distribution with success probability $1-\gamma$, as it defines the likelihood of visiting a state within discounted trials. This interpretation helps justify the form of the weighting in the gradient expression.

5.6 Conclusion

Although the discounted reward setting does not induce a proper probability distribution over states, the policy gradient theorem provides a powerful result that avoids this difficulty by leveraging the structure of the expected return and enabling sampling-based learning algorithms like REINFORCE.

6 Dependence of the Optimal Policy on the Initial State Distribution

We consider a Markov Decision Process (MDP)

$$\big(\mathcal{S},\, \mathcal{A},\, P,\, r,\, \gamma,\, p^0\big),$$

where

- $\mathcal{S}$ is the finite set of states,
- $\mathcal{A}$ is the finite set of actions,
- $P(s' \mid s, a)$ 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 $S_0 \sim p^0$.

6.1 State–Value and Action–Value Functions

For any (possibly stochastic) policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$, define:

$$v^\pi(s) \;=\; \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t\, r(S_t, A_t) \;\Big|\; S_0 = s,\, \pi\Big],$$

$$q^\pi(s,a) \;=\; \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t\, r(S_t, A_t) \;\Big|\; S_0 = s,\, A_0 = a,\, \pi\Big].$$

6.2 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.

6.3 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.

6.4 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.

6.5 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.

Conclusion. The optimal policy for an MDP 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.

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:

- the initial state distribution, and
- the stochastic nature of the policy.

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

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

2. Definition of the TD Error

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

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 first equality shows that subtracting $v_b(S)$ from the action-value $q^\pi(S,A)$ does not introduce bias, because $\mathbb{E}_{A\sim\pi}[\nabla \log \pi(A \mid S)] = 0$.
- 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 kernel 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.

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

We observe that:

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

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

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

Final 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