From First Principles to Modern Practice: A Theoretical Deep Dive into Policy Gradient Guarantees

By Taha Majlesi

July 17, 2025

Abstract

This report provides a comprehensive, first-principles exploration of the theoretical guarantees underpinning modern policy gradient methods in reinforcement learning. The analysis progresses from the foundational Policy Gradient Theorem and the REINFORCE algorithm to the advanced theoretical constructs that ensure stable and monotonic policy improvement. We delve into the challenges of variance and off-policy learning, and show how solutions to these problems lead directly to state-of-the-art algorithms like Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO).

1 The Foundations of Policy Gradient Methods

Policy gradient methods represent a fundamental class of algorithms in reinforcement learning (RL) that directly optimize an agent's decision-making strategy, or policy. Unlike value-based methods that learn the value of states or state-action pairs and then derive a policy from these values, policy gradient methods parameterize the policy itself and use optimization techniques, such as gradient ascent, to improve it. This direct approach is particularly powerful in environments with high-dimensional or continuous action spaces, where computing and storing value functions for every action is intractable. Furthermore, policy gradient methods can learn stochastic policies, which are often optimal in partially observable environments or when facing strategic opponents. This section establishes the foundational principles of policy gradient methods, beginning with a formal definition of the optimization objective, deriving the central theorem that makes this optimization possible, and introducing the seminal REINFORCE algorithm as its most direct implementation.

1.1 The Objective Function: A Formal Goal for RL

The primary goal of any reinforcement learning agent is to learn a policy that maximizes the cumulative reward it receives over time. To apply mathematical optimization, this goal must be formalized as an objective function that can be differentiated with respect to the policy's parameters. The policy, denoted as $\pi_\theta(a|s)$, is a parameterized function (e.g., a neural network with parameters θ) that outputs a probability distribution over actions a given a state s.

The performance of a policy is measured by the expected total discounted reward. This is captured by the objective function, $J(\theta)$, which is defined as the expectation of the sum of discounted rewards over all possible trajectories that could be generated by following the policy π_θ. A trajectory, τ, is a sequence of states and actions, $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$, where T is the horizon of the task.

The objective function can be expressed formally as:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{1}$$

where:

- $\mathbb{E}_{\tau \sim p_\theta(\tau)}[\cdot]$ denotes the expectation over trajectories τ sampled from the distribution $p_\theta(\tau)$.
- $p_\theta(\tau)$ is the probability of observing a specific trajectory τ when the agent follows policy π_θ. This probability is a product of the initial state distribution $p(s_0)$, the policy's action probabilities $\pi_\theta(a_t|s_t)$, and the environment's transition dynamics $p(s_{t+1}|s_t, a_t)$.

- $r(s_t, a_t)$ is the reward received at timestep t after taking action a_t in state s_t.
- $\gamma \in [0, 1]$ is the discount factor, which prioritizes immediate rewards over future ones. It ensures that the infinite sum of rewards remains finite and mathematically tractable.

This formulation casts the reinforcement learning problem as an optimization problem: find the parameters θ^* that maximize $J(\theta)$. The most common family of algorithms for solving such problems is gradient-based optimization, where the parameters are iteratively updated in the direction of the gradient of the objective function: $\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k)$, where α is a step-size parameter or learning rate.

1.2 The Policy Gradient Theorem: Making the Objective Tractable

The primary challenge in optimizing Equation (1) is that the expectation is taken with respect to a distribution, $p_\theta(\tau)$, that itself depends on the parameters θ we wish to optimize. This prevents a naive differentiation where the gradient operator could simply be passed inside the expectation. The Policy Gradient Theorem provides a crucial reformulation of the gradient $\nabla_\theta J(\theta)$ that resolves this issue and makes the objective tractable for estimation.

The derivation of this theorem hinges on a mathematical identity known as the **log-derivative trick** or REINFORCE trick. For any differentiable, positive function $f(x)$, its gradient can be expressed in terms of the gradient of its logarithm: $\nabla_x f(x) = f(x) \nabla_x \log f(x)$. This identity is derived directly from the chain rule for derivatives: $\nabla_x \log f(x) = \frac{1}{f(x)} \nabla_x f(x)$.

The derivation of the Policy Gradient Theorem proceeds as follows:

1. **Expand the Objective Function:** The expectation in $J(\theta)$ is written explicitly as an integral (or sum in the discrete case) over all possible trajectories:

$$J(\theta) = \int_\tau p_\theta(\tau) R(\tau) d\tau$$

   where $R(\tau) = \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$ is the total reward of a trajectory τ.

2. **Differentiate the Objective:** We take the gradient of $J(\theta)$ with respect to the policy parameters θ. Assuming we can swap the gradient and integral operators, we get:

$$\nabla_\theta J(\theta) = \int_\tau \nabla_\theta p_\theta(\tau) R(\tau) d\tau \tag{2}$$

   At this point, the gradient is applied to the trajectory probability $p_\theta(\tau)$, which depends on the unknown environment dynamics. This form is not yet useful for practical estimation.

3. **Apply the Log-Derivative Trick:** The log-derivative trick is now applied to the term $\nabla_\theta p_\theta(\tau)$, rewriting it as $p_\theta(\tau) \nabla_\theta \log p_\theta(\tau)$. Substituting this back into Equation (2) yields:

$$\nabla_\theta J(\theta) = \int_\tau p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d\tau \tag{3}$$

   This step is the core of the derivation. It reintroduces the probability distribution $p_\theta(\tau)$ inside the integral, allowing the entire expression to be rewritten as an expectation.

4. **Rewrite as an Expectation:** Equation (3) is now in the form of an expectation with respect to the original trajectory distribution $p_\theta(\tau)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) R(\tau) \right] \tag{4}$$

5. **Simplify the Log-Probability Term:** The final step is to simplify the term $\nabla_\theta \log p_\theta(\tau)$. The probability of a trajectory is given by:

$$p_\theta(\tau) = p(s_0) \prod_{t=0}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

Taking the logarithm separates this product into a sum:

$$\log p_\theta(\tau) = \log p(s_0) + \sum_{t=0}^{T} (\log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t))$$

When we take the gradient with respect to θ, the terms that do not depend on the policy parameters—namely, the initial state distribution $p(s_0)$ and the environment dynamics $p(s_{t+1}|s_t, a_t)$—vanish. This is a critical result, as it means the final gradient expression does not require knowledge of the environment's model. This leaves us with:

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Final Form of the Policy Gradient Theorem: Substituting this simplified expression back into Equation (4) gives the final form of the Policy Gradient Theorem:

Theorem 1 (Policy Gradient Theorem)**.** *The gradient of the objective function $J(\theta)$ with respect to the policy parameters θ is given by:*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left(\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right) \right] \tag{5}$$

This theorem provides a powerful bridge between theory and practice. It transforms a seemingly intractable optimization problem—differentiating an expectation over a changing distribution that depends on unknown dynamics—into a tractable estimation problem. The right-hand side is an expectation of a quantity that can be computed using only information from a trajectory sampled by running the policy. This allows us to estimate the gradient using Monte Carlo methods.

1.3 The REINFORCE Algorithm: A First Practical Implementation

The REINFORCE algorithm, also known as Monte Carlo Policy Gradient, is the most direct and intuitive practical implementation of the Policy Gradient Theorem. It operates by approximating the expectation in Equation (5) with a sample mean over a batch of trajectories collected by executing the current policy π_θ in the environment.

The REINFORCE algorithm follows a simple three-step iterative process:

1. **Generate Samples:** Run the current policy π_θ for N episodes (trajectories) to collect a dataset $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_N\}$.

2. **Estimate the Gradient:** For each trajectory τ_i in the dataset, compute the gradient estimate. A common and more effective variant of the gradient estimator recognizes that the action taken at timestep t can only influence rewards from timestep t onwards. This principle of causality allows us to replace the total trajectory reward $R(\tau)$ with the "reward-to-go," $\hat{Q}^\pi_{i,t} = \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{i,t'}, a_{i,t'})$. The resulting gradient estimator is:

$$\nabla_\theta J(\theta) \approx \hat{g} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}^\pi_{i,t} \tag{6}$$

3. **Update the Policy:** Update the policy parameters using gradient ascent with a learning rate α:

$$\theta \leftarrow \theta + \alpha \hat{g}$$

Interpretation of the REINFORCE Update Rule: The update rule in Equation (6) has a highly intuitive interpretation.

- The term $\nabla_\theta \log \pi_\theta(a|s)$ is the "score function." It is a vector in the parameter space that points in the direction that most increases the log-probability of taking action a from state s.
- The term $\hat{Q}^\pi_{i,t}$ is the empirical estimate of the expected future return starting from that state-action pair. It acts as a scalar weight for the score function vector.

The update rule effectively "reinforces" actions based on the outcome they produce. If the reward-to-go $\hat{Q}^{\pi}_{i,t}$ is high (a positive outcome), the policy parameters are pushed significantly in the direction that makes the action $a_{i,t}$ more likely in the future. If the reward-to-go is low or negative (a poor outcome), the parameters are pushed in the opposite direction, making that action less likely. This simple mechanism of increasing the probability of "good" actions and decreasing the probability of "bad" ones allows the agent to gradually improve its policy and converge towards one that maximizes the objective function.

2 Taming the Gradient: Variance Reduction with Baselines

While the REINFORCE algorithm is theoretically sound and provides an unbiased estimate of the policy gradient, it suffers from a major practical drawback: high variance in the gradient estimates. This high variance can lead to slow, unstable, and inefficient learning, often requiring a very large number of samples to achieve convergence. This section explores the root cause of this variance problem and introduces the standard, theoretically justified technique for mitigating it: the use of a baseline, most commonly in the form of the advantage function.

2.1 The Problem of High Variance

The variance in the REINFORCE gradient estimator (Equation 6) arises from the reward-to-go term, $\hat{Q}^{\pi}_{i,t}$. The absolute magnitude of this term can vary significantly depending on the environment's reward structure and the stochasticity of the policy and transitions.

Consider an environment where all rewards are positive, such as a game where the agent receives a score of $+1$ for each step it survives. In this scenario, the reward-to-go $\hat{Q}^{\pi}_{i,t}$ will always be positive for every state-action pair in any sampled trajectory. According to the REINFORCE update rule, this means that every action taken will be reinforced (i.e., its probability will be increased). The algorithm will still learn, as actions in trajectories with higher total rewards will be reinforced more strongly, but the learning signal is extremely noisy. The optimizer struggles to differentiate between "good" actions and "very good" actions because it lacks a meaningful reference point. This absolute scaling of rewards, rather than a relative comparison, is a primary source of high variance.

This high variance means that the estimated gradient $\hat{g}$ can fluctuate wildly from one batch of samples to the next. Such fluctuations can cause the policy to take erratic steps in the parameter space, potentially undoing previous learning progress and dramatically slowing down convergence.

2.2 The Advantage Function as a Baseline

To address the high variance issue, a baseline function, $b(s_t)$, is subtracted from the reward-to-go term in the policy gradient estimator. The most common and effective choice for this baseline is an estimate of the state-value function, $V^{\pi}(s_t)$. The state-value function represents the expected return an agent can achieve starting from state s_t and following policy π thereafter.

When the state-value function is used as a baseline, the resulting term is known as the **Advantage Function**, $A^{\pi}(s_t, a_t)$:

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) \tag{7}$$

where $Q^{\pi}(s_t, a_t)$ is the action-value function, representing the expected return after taking action a_t in state s_t and following policy π. In the context of Monte Carlo estimation, this corresponds to replacing $\hat{Q}^{\pi}_{i,t}$ with $\hat{A}^{\pi}_{i,t} = \hat{Q}^{\pi}_{i,t} - \hat{V}(s_{i,t})$, where $\hat{V}(s_{i,t})$ is an estimate of the value function (often learned by a separate neural network, known as a "critic").

The modified policy gradient update becomes:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \hat{A}^{\pi}_{i,t} \tag{8}$$

This formulation is often found in **Actor-Critic** methods, where the "actor" is the policy π_{θ} and the "critic" is the value function estimator $\hat{V}$ used to compute the advantage.

Intuitive Interpretation: The advantage function fundamentally changes the question the agent asks during learning. Instead of asking, "Was the outcome of this action good in an absolute sense?" (the REINFORCE approach), it asks, "Was the outcome of this action better or worse than what I expected to happen on average from this state?".

- If $A^\pi(s,a) > 0$, it means action a led to a better-than-average outcome. The policy should increase the probability of selecting a in state s.
- If $A^\pi(s,a) < 0$, it means action a led to a worse-than-average outcome. The policy should decrease the probability of selecting a in state s.

By centering the learning signal around zero, the advantage function provides a much clearer and more stable signal for optimization. It decouples the value of a state from the specific benefit of an action taken in that state, allowing for more precise credit assignment and significantly reducing the variance of the gradient estimate.

2.3 The Unbiased Nature of Baselines

A critical theoretical property of using a state-dependent baseline is that it reduces the variance of the gradient estimator without introducing any bias. This means that the expected value of the new gradient estimator (using the advantage function) is identical to the expected value of the original REINFORCE estimator. The policy update is still, on average, pointing in the correct direction of ascent for $J(\theta)$.

This can be proven by showing that the expectation of the gradient component associated with the baseline is zero. Let the baseline be $b(s_t)$. The term we are subtracting from the gradient estimator is proportional to $\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)$. Its expectation is:

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

We can analyze the expectation for a single timestep t. Since the baseline $b(s_t)$ depends only on the state s_t, we can condition the expectation on s_t:

$$\begin{aligned}&\mathbb{E}_{s_t\sim p_\theta(s_t)}\left[\mathbb{E}_{a_t\sim\pi_\theta(a_t|s_t)}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right]\right]\\ =&\mathbb{E}_{s_t\sim p_\theta(s_t)}\left[b(s_t)\mathbb{E}_{a_t\sim\pi_\theta(a_t|s_t)}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\right]\right]\end{aligned}$$

Now we focus on the inner expectation, $\mathbb{E}_{a_t\sim\pi_\theta(a_t|s_t)}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$. We can write this as a sum (or integral):

$$\sum_{a_t} \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

Using the log-derivative trick in reverse ($\nabla_\theta f(x) = f(x)\nabla_\theta \log f(x)$), this becomes:

$$\sum_{a_t} \nabla_\theta \pi_\theta(a_t|s_t) = \nabla_\theta \sum_{a_t} \pi_\theta(a_t|s_t)$$

Since $\pi_\theta(a_t|s_t)$ is a probability distribution, the sum of probabilities over all actions must equal 1. Therefore:

$$\nabla_\theta \sum_{a_t} \pi_\theta(a_t|s_t) = \nabla_\theta(1) = 0$$

Because the inner expectation is zero, the entire expression evaluates to zero. This proves that subtracting any baseline $b(s_t)$ that depends only on the state does not change the expected gradient. Its only effect is on the variance of the estimator. A good baseline, one that is highly correlated with the reward-to-go, will dramatically reduce this variance, leading to more stable and efficient learning.

3 A Deeper View: Policy Gradients as Generalized Policy Iteration

While policy gradient methods and classical dynamic programming methods like Policy Iteration (PI) appear distinct at first glance, a deeper theoretical analysis reveals a profound connection. Policy gradient methods can be understood as a "soft" or continuous form of Generalized Policy Iteration (GPI), the fundamental algorithmic template that underlies most reinforcement learning methods. GPI refers to the general idea of letting two interacting processes—policy evaluation and policy improvement—work towards a common goal of finding an optimal policy. This section establishes the formal link between these two paradigms, culminating in a key identity that provides a theoretical guarantee for policy improvement.

3.1 The Duality of Evaluation and Improvement

Policy Iteration is a classic algorithm that finds an optimal policy by alternating between two distinct steps:

1. **Policy Evaluation:** Given a policy π, compute its action-value function, $Q^\pi(s, a)$. This step involves calculating the expected return for taking each action in each state and then following policy π thereafter.
2. **Policy Improvement:** Given the value function Q^π, create a new, improved policy π' by acting greedily with respect to Q^π in every state. That is, for each state s, the new policy selects the action that maximizes $Q^\pi(s, a)$.

This process is guaranteed to converge to the optimal policy. Now, consider the structure of a modern policy gradient algorithm, particularly an actor-critic variant that uses an advantage function baseline:

1. **"Evaluation" Step:** Generate samples by running the current policy π_θ. Use these samples to fit a value function estimator $\hat{V}(s)$ (the critic) and compute the advantage estimates $\hat{A}^\pi(s, a) = \hat{Q}^\pi(s, a) - \hat{V}(s)$. This step is analogous to policy evaluation, as it aims to estimate the value of the current policy's actions.
2. **"Improvement" Step:** Update the policy parameters θ by taking a gradient step in the direction of $\nabla_\theta J(\theta) \propto \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s, a)]$. This step is analogous to policy improvement. Instead of a "hard" greedy update, it performs a "soft" update, gently pushing the policy in the direction of actions with positive advantages.

This parallel structure reveals that policy gradient methods are an instance of Generalized Policy Iteration. The "evaluation" step estimates the advantage of the current policy, and the "improvement" step updates the policy to be "more greedy" with respect to those advantages. This interpretation provides a conceptual bridge between the continuous optimization of policy gradients and the discrete, iterative nature of classical PI.

3.2 The Policy Improvement Identity: A Formal Bridge

The connection between policy gradients and policy iteration is not merely conceptual; it is mathematically precise. A fundamental identity proves that the change in the objective function $J(\theta)$ when moving from an old policy π_θ to a new policy $\pi_{\theta'}$ is directly related to the expected advantage of the new policy's actions, where the advantage is calculated with respect to the old policy.

Theorem 2 (Policy Improvement Identity)**.** *The difference in performance between a new policy $\pi_{\theta'}$ and an old policy π_θ is given by:*

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t) \right] \tag{9}$$

This equation is a cornerstone for understanding theoretical guarantees in policy gradient methods. It asserts that to improve a policy, one should seek a new policy $\pi_{\theta'}$ that, on average, takes actions with a positive advantage over the old policy π_θ. The proof of this identity is a crucial piece of the theoretical puzzle.

Proof. The derivation proceeds by relating the objective function $J(\theta)$ to the value function $V^\pi(s)$ and the advantage function $A^\pi(s, a)$.

1. Start with the definition of the advantage function: $A^{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$.
2. Rearrange to express the reward: $r(s_t, a_t) = A^{\pi_\theta}(s_t, a_t) + V^{\pi_\theta}(s_t) - \gamma V^{\pi_\theta}(s_{t+1})$.

3. Consider the objective function for the new policy, $J(\theta')$, and substitute the expression for the reward. The expectation is over trajectories τ sampled from the new policy $\pi_{\theta'}$.

$$\begin{aligned} J(\theta') &= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \\ &= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t (A^{\pi_\theta}(s_t, a_t) + V^{\pi_\theta}(s_t) - \gamma V^{\pi_\theta}(s_{t+1})) \right] \\ &= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t) \right] + \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} (\gamma^t V^{\pi_\theta}(s_t) - \gamma^{t+1} V^{\pi_\theta}(s_{t+1})) \right] \end{aligned}$$

4. The second term is a telescoping sum:

$$\begin{aligned} &\sum_{t=0}^{\infty} (\gamma^t V^{\pi_\theta}(s_t) - \gamma^{t+1} V^{\pi_\theta}(s_{t+1})) \\ =&(V^{\pi_\theta}(s_0) - \gamma V^{\pi_\theta}(s_1)) + (\gamma V^{\pi_\theta}(s_1) - \gamma^2 V^{\pi_\theta}(s_2)) + \dots \\ =&V^{\pi_\theta}(s_0) \end{aligned}$$

5. The expectation of this term is $\mathbb{E}_{s_0 \sim p(s_0)}[V^{\pi_\theta}(s_0)]$, which is exactly the definition of the objective function for the old policy, $J(\theta)$.

6. Substituting this back, we get:

$$J(\theta') = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t) \right] + J(\theta)$$

7. Rearranging gives the final identity:

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

□

This identity is profound. It provides a direct, formal guarantee for policy improvement: any new policy $\pi_{\theta'}$ that achieves a positive expected advantage over the old policy π_θ is guaranteed to be a better policy in terms of the overall objective $J(\theta)$. However, this identity also reveals a critical practical challenge. The expectation is taken over trajectories sampled from the new policy, $\tau \sim p_{\theta'}(\tau)$, which is the very policy we are trying to find. We cannot evaluate this expectation without first having the new policy. This "chicken-and-egg" problem is the central theoretical tension that motivates the development of more advanced policy gradient algorithms. It necessitates a method for estimating this expected advantage using only samples collected from the old policy, π_θ.

4 The Off-Policy Challenge and the Importance Sampling Solution

The policy improvement identity (Equation 9) provides a powerful theoretical guarantee, but its practical application is hindered by a fundamental mismatch: it requires evaluating the expected advantage under the new policy's trajectory distribution ($p_{\theta'}$), while we only have access to samples generated by the old policy (p_θ). This is a classic off-policy evaluation problem. The solution lies in a statistical technique called Importance Sampling (IS), which allows us to correct for this distributional mismatch. However, while IS provides a theoretical solution, it introduces its own set of practical challenges related to variance.

4.1 The State Distribution Mismatch

The core of the off-policy problem lies in the expectation $\mathbb{E}_{\tau\sim p_{\theta'}(\tau)}[\cdot]$. This expectation depends on θ' in two distinct ways:

1. **Action Probabilities:** The actions a_t are sampled from the new policy, $\pi_{\theta'}(a_t|s_t)$.
2. **State Visitation Distribution:** The states s_t are visited according to the distribution induced by running policy $\pi_{\theta'}$, which we denote as $p_{\theta'}(s_t)$. Changing the policy changes the decisions the agent makes, which in turn changes the sequence of states it encounters.

The policy improvement identity can be expanded to make this explicit:

$$J(\theta') - J(\theta) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t\sim p_{\theta'}(s_t)} \left[\mathbb{E}_{a_t\sim\pi_{\theta'}(a_t|s_t)}[A^{\pi_\theta}(s_t, a_t)]\right] \tag{10}$$

To make progress, we need a way to estimate this quantity using data collected under π_θ. This requires correcting for the mismatch in both the action selection and the state visitation distribution.

4.2 Importance Sampling (IS) for Off-Policy Correction

Importance Sampling is a general Monte Carlo technique used to estimate the properties of a target distribution p using samples generated from a different proposal or behavior distribution q. The fundamental principle of IS is to re-weight the samples from q to account for the difference in probabilities between the two distributions.

The expected value of a function $f(x)$ under the target distribution p can be written as:

$$\mathbb{E}_{x\sim p}[f(x)] = \int p(x)f(x)dx$$

By multiplying and dividing by the behavior distribution $q(x)$, we get:

$$= \int q(x)\frac{p(x)}{q(x)}f(x)dx$$

This expression is now an expectation under the behavior distribution q:

$$\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right] \tag{11}$$

The term $w(x) = \frac{p(x)}{q(x)}$ is called the **importance weight** or importance ratio. It corrects for the fact that we are sampling from q instead of p. If a sample x is more likely under p than q, its importance weight is greater than 1, and its contribution to the estimate is up-weighted. Conversely, if it is less likely under p, it is down-weighted. A critical condition for IS to be valid is that the support of q must cover the support of p; that is, if $p(x) > 0$, then $q(x)$ must also be greater than 0.

4.3 Reformulating the Objective with Importance Sampling

We can apply the principle of importance sampling to the inner expectation of Equation (10), which deals with the action selection mismatch. Here, the target policy is $\pi_{\theta'}$ and the behavior policy is π_θ. Applying Equation (11), we get:

$$\mathbb{E}_{a_t\sim\pi_{\theta'}(a_t|s_t)}[A^{\pi_\theta}(s_t, a_t)] = \mathbb{E}_{a_t\sim\pi_\theta(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}A^{\pi_\theta}(s_t, a_t)\right] \tag{12}$$

Substituting this back into Equation (10) gives:

$$J(\theta') - J(\theta) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t\sim p_{\theta'}(s_t)}\left[\mathbb{E}_{a_t\sim\pi_\theta(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}A^{\pi_\theta}(s_t, a_t)\right]\right] \tag{13}$$

This expression is still not fully tractable because the outer expectation is over the new state distribution, $p_{\theta'}(s_t)$. However, a common approach in policy gradient methods is to make an approximation: assume

that the state visitation distribution does not change much for a small change in policy, i.e., $p_{\theta'}(s_t) \approx p_\theta(s_t)$. This leads to the creation of a **surrogate objective function**, $L_\theta(\theta')$, which approximates the true performance improvement but can be estimated using data from the old policy π_θ:

$$L_\theta(\theta') = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim p_\theta(s_t)} \left[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)} \left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} A^{\pi_\theta}(s_t, a_t) \right] \right] \tag{14}$$

This can be written more compactly as an expectation over states and actions sampled from the old policy:

$$L_\theta(\theta') = \mathbb{E}_{s_t \sim p_\theta, a_t \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} A^{\pi_\theta}(s_t, a_t) \right] \tag{15}$$

The goal now becomes to find a new policy $\pi_{\theta'}$ that maximizes this surrogate objective. By optimizing $L_\theta(\theta')$, we hope to improve the true objective $J(\theta')$. The validity of this approximation hinges on the assumption that $\pi_{\theta'}$ remains "close" to π_θ.

This introduces a new, severe practical problem. Importance sampling is a theoretically sound technique, but it is notoriously prone to high variance. If the new policy $\pi_{\theta'}$ becomes significantly different from the old policy π_θ, the importance ratio $\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}$ can become very large or very close to zero for certain state-action pairs. When this happens, the variance of the gradient estimate for $L_\theta(\theta')$ can explode, as the entire estimate may be dominated by a few samples with extreme weights. This can completely destabilize the learning process. This observation is the primary motivation for constraining how much the new policy can diverge from the old one, leading directly to the concept of a "trust region."

5 Theoretical Guarantees for Monotonic Improvement

The use of a surrogate objective (Equation 15) is a practical necessity, but it rests on the crucial approximation that the state visitation distribution does not change significantly ($p_{\theta'} \approx p_\theta$). For policy gradient methods to be reliable, we need a formal guarantee that maximizing this surrogate objective will lead to an improvement in the true objective, $J(\theta)$. This requires establishing a rigorous mathematical relationship between the two. The theory presented in the lecture slides provides exactly this, by first bounding how much the state distribution can change as a function of the policy change, and then using this bound to derive a lower bound on the true performance improvement. This formalizes the concept of a "trust region," within which the surrogate objective is a faithful approximation.

5.1 Bounding the Change in State Distribution

The first step is to quantify how the "closeness" of two policies, π_θ and $\pi_{\theta'}$, affects the "closeness" of their induced state visitation distributions, $p_\theta(s_t)$ and $p_{\theta'}(s_t)$.

Let's define the maximum difference in action probabilities between the two policies as ϵ:

$$\max_{s,a} |\pi_{\theta'}(a|s) - \pi_\theta(a|s)| \leq \epsilon$$

This means that at any state, the probability of taking any action under the new policy differs from the old policy by at most ϵ. A key lemma from probability theory states that if two distributions have a total variation distance of ϵ, they can be coupled such that they agree with probability $1 - \epsilon$. In this context, it implies that at any step, the new policy $\pi_{\theta'}$ will choose a "different" action from what π_θ would have chosen with a probability of at most ϵ.

Consider the state distribution at timestep t. The distribution $p_{\theta'}(s_t)$ can be thought of as a mixture. With probability $(1-\epsilon)^t$, no "different" actions have been taken up to step t, so the agent's trajectory has evolved identically to one under π_θ, and the state distribution is $p_\theta(s_t)$. With probability $1-(1-\epsilon)^t$, at least one "different" action has been taken, leading to a trajectory that has diverged and results in some other state distribution, which can be called $p_{\text{mistake}}(s_t)$. This gives the relation:

$$p_{\theta'}(s_t) = (1-\epsilon)^t p_\theta(s_t) + (1-(1-\epsilon)^t) p_{\text{mistake}}(s_t)$$

To bound the difference between the state distributions, we take the absolute difference:

$$\begin{aligned} |p_{\theta'}(s_t) - p_\theta(s_t)| &= |(1-\epsilon)^t p_\theta(s_t) + (1-(1-\epsilon)^t) p_{\text{mistake}}(s_t) - p_\theta(s_t)| \\ &= |-(1-(1-\epsilon)^t) p_\theta(s_t) + (1-(1-\epsilon)^t) p_{\text{mistake}}(s_t)| \\ &= (1-(1-\epsilon)^t) |p_{\text{mistake}}(s_t) - p_\theta(s_t)| \end{aligned}$$

The maximum possible total variation distance between any two probability distributions is 2. Therefore, $|p_{\text{mistake}}(s_t) - p_\theta(s_t)| \leq 2$. This gives us a bound:

$$|p_{\theta'}(s_t) - p_\theta(s_t)| \leq 2(1 - (1-\epsilon)^t)$$

Using the inequality $(1-x)^n \geq 1 - nx$ for $x \in [0, 1]$, we can further simplify this to a linear bound:

$$|p_{\theta'}(s_t) - p_\theta(s_t)| \leq 2(1 - (1 - \epsilon t)) = 2\epsilon t \tag{16}$$

Although this bound may not be tight, it formally establishes that if the change in policy is small (small ϵ), the change in the state visitation distribution is also small and grows linearly with the time horizon t.

5.2 Deriving a Lower Bound on Performance (The Surrogate Objective)

With the bound on the state distribution change, we can now connect the true objective improvement $J(\theta') - J(\theta)$ to the surrogate objective $L_\theta(\theta')$. Recall the exact expression for the improvement:

$$J(\theta') - J(\theta) = \sum_t \gamma^t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)}[f(s_t)]$$

where $f(s_t) = \mathbb{E}_{a_t \sim \pi_{\theta'}(a_t|s_t)}[A^{\pi_\theta}(s_t, a_t)]$.

We can relate the expectation under $p_{\theta'}$ to the expectation under p_θ using the following inequality for any function $f(s)$:

$$\begin{aligned}
\mathbb{E}_{s \sim p_{\theta'}(s)}[f(s)] &= \sum_s p_{\theta'}(s) f(s) \\
&\geq \sum_s p_\theta(s) f(s) - \sum_s |p_{\theta'}(s) - p_\theta(s)| \max_s |f(s)| \\
\mathbb{E}_{s \sim p_{\theta'}(s)}[f(s)] &\geq \mathbb{E}_{s \sim p_\theta(s)}[f(s)] - \max_s |f(s)| \sum_s |p_{\theta'}(s) - p_\theta(s)|
\end{aligned}$$

The sum of absolute differences is the total variation distance, which we bounded by $2\epsilon t$. Let $C_t = \max_{s_t} |f(s_t)|$. Then:

$$\mathbb{E}_{s_t \sim p_{\theta'}(s_t)}[f(s_t)] \geq \mathbb{E}_{s_t \sim p_\theta(s_t)}[f(s_t)] - 2\epsilon t C_t$$

Applying this to the full performance improvement sum:

$$J(\theta') - J(\theta) \geq \sum_t \gamma^t (\mathbb{E}_{s_t \sim p_\theta(s_t)}[f(s_t)] - 2\epsilon t C_t)$$

The first term inside the sum is exactly the surrogate objective $L_\theta(\theta')$. The second term is an error penalty. This gives us the final lower bound on performance improvement:

$$J(\theta') - J(\theta) \geq L_\theta(\theta') - C \cdot D_{\text{TV}}(\pi_\theta, \pi_{\theta'}) \tag{17}$$

where C is a constant that depends on the maximum advantage and the horizon, and D_{TV} is a measure of the total variation distance between the policies (related to our ϵ).

This is the key theoretical result. It guarantees that maximizing the surrogate objective $L_\theta(\theta')$ also maximizes a lower bound on the true objective improvement $J(\theta') - J(\theta)$. The approximation is sound as long as the error term, which grows with the distance between the policies, is kept under control.

5.3 The "Trust Region" Concept

The performance bound in Equation (17) gives rise to the concept of a **trust region**. A trust region is a neighborhood around the current policy π_θ within which the surrogate objective $L_\theta(\theta')$ is a reasonably good approximation of the true objective improvement.

The theory reveals a fundamental trade-off in policy optimization related to the size of the policy update step.

- **Small Step:** If we take a very small step, such that $\pi_{\theta'}$ is very close to π_θ, then the error term in Equation (17) is negligible. In this case, maximizing the surrogate objective reliably improves the true objective. This guarantees stable, monotonic improvement but at the cost of slow learning and low sample efficiency.

- **Large Step:** If we take a large step, we might learn faster. However, we risk leaving the trust region where the surrogate approximation is valid. The error term could become very large, meaning that even if we find a new policy with a high surrogate objective value, its true performance could be catastrophically worse due to the large, negative error term.

This trade-off is central to modern policy gradient methods. The challenge is no longer just to find a direction of improvement, but to find the largest possible step in that direction that does not violate the trust region constraint. This ensures both rapid learning and stable, monotonic improvement. Algorithms like Trust Region Policy Optimization (TRPO) are designed to solve this constrained optimization problem directly.

6 From Theory to Practice: TRPO and PPO

The theoretical framework developed in the preceding sections—connecting policy gradients to policy iteration and deriving performance bounds that justify a trust region—provides the foundation for modern, high-performance reinforcement learning algorithms. This section examines two of the most influential algorithms that emerged from this line of reasoning: Trust Region Policy Optimization (TRPO), which provides a formal, rigorous solution to the constrained optimization problem, and Proximal Policy Optimization (PPO), a more pragmatic and widely used approximation that achieves similar performance with greater simplicity.

6.1 Trust Region Policy Optimization (TRPO): The Formal Solution

Trust Region Policy Optimization (TRPO) is the direct and rigorous implementation of the theoretical principles laid out in Section 5. It aims to maximize the surrogate objective function while explicitly constraining the new policy to remain within a trust region around the old policy. This constraint prevents destructively large policy updates and promotes monotonic performance improvement.

The TRPO optimization problem is formulated as follows:

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & L_{\theta_{\text{old}}}(\theta) = \mathbb{E}_{s,a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}\hat{A}_{\theta_{\text{old}}}(s,a)\right] \\ \text{subject to} \quad & \bar{D}_{\text{KL}}(\pi_{\theta_{\text{old}}}||\pi_\theta) \leq \delta \end{aligned} \tag{18}$$

where:

- $L_{\theta_{\text{old}}}(\theta)$ is the surrogate objective we aim to maximize.
- $\bar{D}_{\text{KL}}(\pi_{\theta_{\text{old}}}||\pi_\theta) = \mathbb{E}_{s\sim\pi_{\theta_{\text{old}}}}[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)||\pi_\theta(\cdot|s))]$ is the average Kullback-Leibler (KL) divergence between the old and new policies over the states visited by the old policy. The KL divergence is a more robust measure of the "distance" between two probability distributions than a simple parameter-space distance.
- δ is a small scalar hyperparameter that defines the "radius" of the trust region.

Solving this constrained optimization problem is non-trivial. TRPO approximates the objective with a linear function and the KL-divergence constraint with a quadratic function (via Taylor expansion). This results in a quadratic program that can be solved efficiently. In practice, this is accomplished using the conjugate gradient algorithm. The conjugate gradient method allows for the efficient calculation of the term $H^{-1}g$ (where H is the Fisher Information Matrix, the second-order approximation of the KL-divergence, and g is the policy gradient) without needing to explicitly form, store, and invert the large matrix H. This makes the update feasible for large neural network policies. After finding the update direction with conjugate gradient, a backtracking line search is performed to ensure that the KL constraint is satisfied and that the actual objective improves.

While TRPO is founded on strong theoretical guarantees and often produces stable, monotonic improvements, its implementation is complex due to the reliance on second-order optimization methods and the line search procedure.

6.2 Proximal Policy Optimization (PPO): The Pragmatic Approximation

Proximal Policy Optimization (PPO) was introduced as a simpler alternative to TRPO that achieves comparable or better performance with a much simpler implementation. PPO avoids the complex second-order optimization of TRPO by using a novel **clipped surrogate objective function**. This new objective implicitly constrains the size of the policy update, effectively creating a trust region using only first-order optimization.

The core idea of PPO is to modify the surrogate objective to penalize policy changes that move the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ too far from 1. The PPO-Clip objective function is:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{19}$$

where:

- $\hat{A}_t$ is the estimated advantage at timestep t.
- ϵ is a small hyperparameter (e.g., 0.2) that defines the clipping range.
- The clip function constrains the ratio $r_t(\theta)$ to lie within the interval $[1-\epsilon, 1+\epsilon]$.

The min operator is the key to the algorithm's behavior:

- **When Advantage $\hat{A}_t$ is positive:** An action was better than expected. The objective is $\min(r_t(\theta)\hat{A}_t, (1+\epsilon)\hat{A}_t)$. This means the objective increases as the policy ratio $r_t(\theta)$ increases, but the increase is "clipped" once the ratio exceeds $1+\epsilon$. This removes the incentive for making the policy update too large, as there is no further gain in the objective.
- **When Advantage $\hat{A}_t$ is negative:** An action was worse than expected. The objective is $\min(r_t(\theta)\hat{A}_t, (1-\epsilon)\hat{A}_t)$. Since $\hat{A}_t$ is negative, the second term $(1-\epsilon)\hat{A}_t$ is larger (less negative) than $(1+\epsilon)\hat{A}_t$. The objective is maximized by making $r_t(\theta)\hat{A}_t$ as large as possible, which means making $r_t(\theta)$ smaller. However, the gain is clipped once $r_t(\theta)$ goes below $1-\epsilon$. This prevents the policy from being updated too aggressively in response to a bad action.

By taking the minimum of the normal objective and the clipped objective, PPO ensures that the policy update is conservative, effectively keeping the new policy "proximal" to the old one. This simple clipping mechanism can be optimized with standard gradient ascent algorithms like Adam, making PPO much easier to implement and tune than TRPO.

6.3 Synthesis and Comparison

The evolution from the basic REINFORCE algorithm to the sophisticated TRPO and PPO methods illustrates a clear trajectory in reinforcement learning research: a continuous effort to balance theoretical rigor, sample efficiency, stability, and implementation complexity.

- **REINFORCE** provided the foundational insight but was too unstable for most practical applications.
- **TRPO** addressed the stability problem with strong theoretical guarantees of monotonic improvement, but its implementation complexity limited its widespread adoption.
- **PPO** struck a remarkable balance. It sacrificed the hard theoretical guarantees of TRPO for a much simpler, first-order algorithm that proved to be empirically robust and highly effective. This pragmatic trade-off has made PPO the default reinforcement learning algorithm for a wide range of applications, from game playing to robotics.

The success of PPO highlights a crucial lesson in applied AI research: a theoretically "less pure" but more scalable and practical algorithm can often have a greater impact than a more rigorous but complex counterpart. PPO's clipped objective is a heuristic, yet its ability to capture the essential spirit of the trust region constraint in a simple, first-order-friendly manner has made it a cornerstone of modern deep reinforcement learning.

The following table provides a concise summary and comparison of these key policy gradient algorithms.

Table 1: Comparative Analysis of Policy Gradient Algorithms

Feature	REINFORCE	Trust Region Policy Optimization (TRPO)	Proximal Policy Optimization (PPO)
Core Idea	Direct Monte-Carlo gradient of the objective.	Maximize surrogate objective within a "trust region" defined by KL-divergence.	Simplify TRPO's constraint with a clipped surrogate objective.
Objective/Update Rule	$\nabla J \propto \mathbb{E}[\nabla \log \pi \cdot \hat{Q}]$	$\max_\theta L(\theta)$ s.t. $D_{KL}(\pi_{old}\|\|\pi_{new}) \leq \delta$	$\max_\theta L^{\text{CLIP}}(\theta)$
Sample Efficiency	Low (fully on-policy, no data reuse between updates).	Moderate (allows multiple optimization steps on collected data).	Moderate-to-High (simpler updates allow for more data reuse per batch).
Stability	Low (high variance, no monotonic guarantees).	High (theoretical monotonic improvement guarantees under certain conditions).	High (empirically stable, though lacks TRPO's hard guarantees).
Implementation Complexity	Simple (direct gradient calculation).	Complex (requires second-order methods, e.g., conjugate gradient).	Moderate (first-order optimization, easy to implement in standard frameworks).
Theoretical Foundation	Policy Gradient Theorem.	Constrained optimization on a theoretical lower bound of policy performance.	A heuristic/practical approximation of the TRPO objective.

7 Conclusions

This report has provided a comprehensive, first-principles exploration of the theoretical guarantees underpinning modern policy gradient methods in reinforcement learning. The analysis progressed from the foundational Policy Gradient Theorem and the REINFORCE algorithm to the advanced theoretical constructs that ensure stable and monotonic policy improvement.

The key conceptual journey can be summarized as follows:

- **The Gradient Problem and its Solution:** The Policy Gradient Theorem, through the log-derivative trick, transforms the intractable problem of differentiating an objective function with respect to a changing probability distribution into a tractable expectation that can be estimated via Monte Carlo sampling. The REINFORCE algorithm is the direct embodiment of this principle.
- **The Variance Problem and its Solution:** The high variance of the REINFORCE estimator is mitigated by subtracting a state-dependent baseline, most effectively the state-value function. This gives rise to the advantage function, which provides a more stable, relative learning signal without introducing bias.
- **The Off-Policy Problem and its Solution:** Viewing policy gradients as a form of policy iteration reveals a fundamental challenge: the policy improvement guarantee requires evaluating performance under a new, unknown policy. Importance sampling provides a theoretical correction, leading to a surrogate objective function that can be optimized using data from the current policy.
- **The Stability Problem and its Solution:** The use of importance sampling and the approximation of the state distribution introduce the risk of instability if policy updates are too large. The derivation of a formal lower bound on policy performance justifies constraining the policy update to a "trust region" where the surrogate objective is a faithful approximation of true performance.
- **Practical Implementation:** This theoretical framework culminates in practical algorithms. TRPO provides a rigorous, second-order solution to the constrained optimization problem, while PPO offers a simpler, first-order approximation with its clipped objective that has proven to be remarkably effective and scalable, becoming a de facto standard in the field.