Study Material - Youtube

Document Information

- **Generated:** 2025-08-26 08:03:45
- **Source:** https://www.youtube.com/watch?v=5FOz4agFl4M
- **Platform:** Youtube
- **Word Count:** 2,476 words
- **Estimated Reading Time:** ~12 minutes
- **Number of Chapters:** 3
- **Transcript Available:** Yes (analyzed from video content)

Table of Contents

Video Overview

This lecture, titled "Policy Gradient Theorem," is a segment from the "Mathematical Foundations of Generative AI" course. The instructor, Prof. Prathosh A P, provides a detailed mathematical derivation of the Policy Gradient Theorem, a cornerstone of modern reinforcement learning. The primary goal is to develop a method for optimizing a policy, represented by a parameterized function (like a neural network), to maximize the expected total reward. The lecture systematically builds from the fundamental objective function in reinforcement learning to a practical and efficient form of the policy gradient that is widely used in state-of-the-art algorithms, including those for aligning large language models.

Learning Objectives

Upon completing this lecture, students will be able to: - **Understand the Reinforcement Learning Optimization Problem**: Formulate the goal of reinforcement learning as an optimization problem aimed at maximizing an objective function $J(\theta)$. - **Derive the Policy Gradient Theorem**: Follow and explain the step-by-step mathematical derivation of the gradient of the objective function, $\nabla_\theta J(\theta)$. - **Master the Log-Derivative Trick**: Understand the intuition and application of the log-derivative trick, a key mathematical tool used in the derivation. - **Appreciate Variance Reduction Techniques**: Comprehend why the initial form of the policy gradient has high variance and how causality and baselines (like the value function) are used to create a more stable estimator. - **Define and Relate Key RL Functions**: Clearly define and understand the relationships between the Value Function ($V^\pi(s)$), the Q-function ($Q^\pi(s,a)$), and the Advantage Function ($A^\pi(s,a)$). - **Formulate the Practical Policy Gradient**: Arrive at the final, practical expression for the policy gradient involving the advantage function, which is used in advanced algorithms like PPO.

Prerequisites

To fully grasp the concepts in this video, students should have a foundational understanding of: - **Basic Reinforcement Learning (RL)**: Familiarity with core RL concepts such as policy (π), state (s), action (a), reward (r), and trajectory (τ). - **Calculus**: A solid understanding of derivatives, gradients (∇), and the chain rule. - **Probability & Statistics**: Knowledge of probability distributions, expected values ($\mathbb{E}$), and the concept of an unbiased estimator. - **Logarithms**: Basic properties of logarithms, particularly the derivative of a logarithm.

Key Concepts

- **Objective Function** $J(\theta)$
- **Policy Gradient** $\nabla_\theta J(\theta)$
- **Gradient Ascent**
- **Log-Derivative Trick**
- **Trajectory Probability** $p_\theta(\tau)$
- **Reward-to-Go** R_t
- **Baseline Function** $b(s_t)$
- **Value Function** $V^\pi(s_t)$
- **Q-Function (Action-Value Function)** $Q^\pi(s_t, a_t)$
- **Advantage Function** $A^\pi(s_t, a_t)$

The Policy Gradient Method: A Deep Dive

The Optimization Problem in Reinforcement Learning

Intuitive Foundation

In reinforcement learning, our goal is to train an agent to make a sequence of decisions in an environment to achieve the highest possible cumulative reward. We call the agent's decision-making strategy a **policy**, denoted by π. When this policy is defined by a set of parameters θ (for instance, the weights of a neural network), we write it as π_θ.

The agent interacts with the environment over a series of time steps, creating a **trajectory** (or episode) τ, which is a sequence of states and actions: $\tau = (s_0, a_0, s_1, a_1, \dots)$. For each trajectory, the agent receives a total reward, $R(\tau)$. Since the policy and the environment can be stochastic, we can't guarantee the same reward for every trajectory. Therefore, our goal is to find the policy parameters θ that maximize the *expected* total reward over all possible trajectories.

Mathematical Formulation

(00:11) - The instructor formalizes this goal with an **objective function**, $J(\theta)$. This function represents the expected total reward for a trajectory τ sampled according to the policy π_θ.

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)]$$

Where: - $J(\theta)$ is the objective function we want to maximize. - θ represents the parameters of our policy (e.g., neural network weights). - τ is a trajectory. - $p_\theta(\tau)$ is the probability of observing trajectory τ when following policy π_θ. - $R(\tau)$ is the total reward for trajectory τ. - $\mathbb{E}_{\tau \sim p_\theta(\tau)}[\cdot]$ denotes the expectation over all trajectories sampled from the policy.

This expectation can also be written as an integral over the space of all possible trajectories:

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

The ultimate goal is to find the optimal parameters θ^* that define the best possible policy π_{θ^*}:

$$\pi_{\theta^*} = \arg\max_\pi J(\theta)$$

(00:33) - To solve this maximization problem, we can use **gradient-based optimization**. Specifically, since we are maximizing, we use **gradient ascent**. The update rule for the policy parameters at each iteration t is:

$$\theta^{t+1} \leftarrow \theta^t + \alpha \nabla_\theta J(\theta)$$

Here, α is the learning rate. This simple rule forms the basis of policy gradient methods. However, the main challenge is to compute the gradient of the objective function, $\nabla_\theta J(\theta)$. The rest of the lecture is dedicated to deriving a computable expression for this gradient.

The following flowchart illustrates the overall process of policy optimization using gradient ascent.

```
flowchart TD
    A["Start with an initial policy $\pi_{\theta_0}$"] --> B{"Generate trajectories $\tau$ using curren
    B --> C["Estimate the policy gradient $\nabla_\theta J(\theta)$"]
    C --> D["Update policy parameters:<br/>$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$"]
    D --> B
```

Figure 1: The iterative process of policy optimization via gradient ascent.

The Policy Gradient Theorem: Derivation

(02:08) - The **Policy Gradient Theorem** provides a way to compute the gradient of the objective function, $\nabla_\theta J(\theta)$, without needing to know the dynamics of the environment.

Step-by-Step Derivation

1. Start with the definition of the gradient. We begin with the integral form of the objective function and apply the gradient operator ∇_θ.

$$\nabla_\theta J(\theta) = \nabla_\theta \int p_\theta(\tau) R(\tau) d\tau$$

2. Push the gradient inside the integral. (03:50) - Since the integration is over trajectories τ and the differentiation is with respect to the policy parameters θ, we can swap the order of these operations.

$$\nabla_\theta J(\theta) = \int \nabla_\theta p_\theta(\tau) R(\tau) d\tau$$

> **Problem:** This form is not an expectation, as it involves the gradient of the probability distribution, $\nabla_\theta p_\theta(\tau)$, not the distribution itself. We cannot estimate this directly by sampling trajectories.

3. Apply the Log-Derivative Trick. (04:40) - To solve this, we use a clever identity from calculus known as the **log-derivative trick**. The intuition is to re-introduce $p_\theta(\tau)$ into the expression to form an expectation.

The identity states that for any function $f(x)$, the derivative of its logarithm is:

$$\nabla_x \log f(x) = \frac{\nabla_x f(x)}{f(x)}$$

Rearranging this gives:

$$\nabla_x f(x) = f(x) \nabla_x \log f(x)$$

Applying this to our probability distribution $p_\theta(\tau)$:

$$\nabla_\theta p_\theta(\tau) = p_\theta(\tau) \nabla_\theta \log p_\theta(\tau)$$

4. Substitute back into the integral. We replace $\nabla_\theta p_\theta(\tau)$ in our gradient expression with the result from the log-derivative trick.

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \left(\nabla_\theta \log p_\theta(\tau)\right) R(\tau) d\tau$$

5. Convert the integral back to an expectation. This expression is now in the form of an expectation, which can be estimated using Monte Carlo sampling (i.e., by running the policy and collecting trajectories).

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

This is a foundational result. It tells us that the policy gradient is the expected value of the product of the score function ($\nabla_\theta \log p_\theta(\tau)$) and the reward ($R(\tau)$).

Decomposing the Trajectory Probability

(06:07) - To make this expression more practical, we need to break down $\log p_\theta(\tau)$. A trajectory's probability is the product of the initial state probability and the probabilities of subsequent actions and state transitions.

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

Taking the logarithm, the products become sums:

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

Now, we take the gradient with respect to θ. The terms that do not depend on the policy parameters θ—the initial state probability $p(s_0)$ and the environment dynamics $p(s_{t+1}|s_t, a_t)$—have a gradient of zero.

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

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

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

Substituting this back into our gradient expression gives the final form of the Policy Gradient Theorem:

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

This equation is powerful because it allows us to estimate the policy gradient using only samples from the policy's interaction with the environment, without needing to know the environment's transition model.

Improving the Gradient Estimator

The expression derived above is an unbiased estimator of the true gradient, but it suffers from high variance, which can make training unstable. The lecture discusses several ways to improve it.

Causality and Reward-to-Go

(11:51) - A key observation is that an action a_t taken at time t can only affect rewards from that point forward, not rewards that have already been received. This is the principle of **causality**. The current gradient formulation assigns credit for the total reward $R(\tau)$ to every action in the trajectory, which is inefficient.

To fix this, we can replace the total reward $R(\tau)$ with the **reward-to-go** (or go-to reward), which is the sum of rewards from time step t onwards.

$$R_t = \sum_{k=t}^{T} \gamma^{k-t} r(s_k, a_k)$$

The policy gradient expression becomes:

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

This change does not introduce bias but significantly reduces the variance of the gradient estimate.

Variance Reduction with a Baseline

(14:12) - Another powerful technique for variance reduction is to subtract a **baseline** function, $b(s_t)$, from the reward-to-go term. The baseline must only depend on the state s_t and not the action a_t.

The gradient becomes:

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

This modification does not change the expected value of the gradient (it remains unbiased) but can dramatically reduce its variance if $b(s_t)$ is chosen well. A common and effective choice for the baseline is the **value function** $V^\pi(s_t)$.

The Advantage Function

(19:52) - Using the value function as a baseline leads us to the concept of the **Advantage Function**. Let's define the relevant functions:

- **Value Function** $V^\pi(s)$: The expected return starting from state s and following policy π.

$$V^\pi(s_t) = \mathbb{E}_\pi [R_t|s_t]$$

- **Q-Function** $Q^\pi(s, a)$: The expected return starting from state s, taking action a, and then following policy π.

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi [R_t|s_t, a_t]$$

- **Advantage Function** $A^\pi(s, a)$: Measures how much better it is to take a specific action a in state s compared to the average action according to the policy π.

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

(21:50) - The term $(R_t - V^\pi(s_t))$ is a single-sample, unbiased estimator of the advantage function. This is because the expected value of the reward-to-go R_t, given the state s_t and action a_t, is precisely the Q-function.

$$\mathbb{E}[R_t - V^\pi(s_t)|s_t, a_t] = \mathbb{E}[R_t|s_t, a_t] - V^\pi(s_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) = A^\pi(s_t, a_t)$$

Final Practical Form of the Policy Gradient

(24:00) - By replacing the reward-to-go minus baseline term with the advantage function, we arrive at the most common and practical form of the policy gradient:

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

This form is central to many modern RL algorithms, such as Actor-Critic methods, where one network (the actor) represents the policy π_θ and another network (the critic) estimates the advantage function A^π.

The following diagram shows the relationships between these key reinforcement learning functions.

```
graph TD
    subgraph "Policy Optimization"
        J["Objective Function<br/>$J(\theta) = \mathbb{E}[R(\tau)]$"]
        PG["Policy Gradient<br/>$\nabla_\theta J(\theta)$"]
        Pi["Policy<br/>$\pi_\theta(a|s)$"]

        J --"is maximized by"--> Pi
        Pi --"is updated using"--> PG
        PG --"is derived from"--> J
    end

    subgraph "Value Estimation"
        A["Advantage Function<br/>$A^\pi(s,a)$"]
        Q["Q-Function<br/>$Q^\pi(s,a) = \mathbb{E}[R_t | s,a]$"]
        V["Value Function<br/>$V^\pi(s) = \mathbb{E}[R_t | s]$"]

        Q --> A
        V --> A
        A --"is estimated by<br/>$R_t - V^\pi(s_t)$"--> PG
        Q -.-> V
    end

    J --> PG
```

Figure 2: Conceptual relationships between key functions in Policy Gradient methods.

Key Takeaways from This Video

- **Core Idea**: Policy gradient methods optimize a parameterized policy by performing gradient ascent on the expected reward objective function $J(\theta)$.
- **The Policy Gradient Theorem**: Provides a way to calculate the gradient $\nabla_\theta J(\theta)$ as an expectation, making it possible to estimate from sampled trajectories. The fundamental expression is $\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\left(\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) \right) R(\tau) \right]$.
- **Log-Derivative Trick**: A crucial mathematical technique that transforms the gradient of a probability distribution into a more usable form involving the gradient of the log-probability, allowing the expression to be framed as an expectation.
- **Variance Reduction is Key**: The basic policy gradient estimator is often too noisy for stable learning. Techniques like using reward-to-go and subtracting a state-dependent baseline (like the value function) are essential for practical applications.

- **Advantage Function**: The term $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s,a)$ provides an intuitive and effective signal for policy updates, indicating how much better a particular action is than the policy's average behavior in that state.
- **Practical Application**: The final form of the gradient, $\nabla_\theta J(\theta) \approx \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) A^\pi(s_t,a_t)\right]$, is the foundation for many advanced RL algorithms, including those used to align large language models with human preferences.

Self-Assessment for This Video

1. **Objective Function**: What does the objective function $J(\theta)$ represent in reinforcement learning, and why do we want to maximize it?
2. **Log-Derivative Trick**: Explain the log-derivative trick in your own words. Why is it necessary for deriving the policy gradient theorem? What problem does it solve?
3. **Derivation Step**: In the derivation of $\nabla_\theta \log p_\theta(\tau)$, why do the terms related to the initial state distribution $p(s_0)$ and the environment dynamics $p(s_{t+1}|s_t,a_t)$ disappear?
4. **Variance Reduction**:
   - What is "reward-to-go," and why is it a better credit assignment mechanism than the total trajectory reward $R(\tau)$?
   - Explain why subtracting a state-dependent baseline $b(s_t)$ from the reward-to-go does not introduce bias into the gradient estimate.
5. **Advantage Function**:
   - Define the value function $V^\pi(s)$, the Q-function $Q^\pi(s,a)$, and the advantage function $A^\pi(s,a)$.
   - Prove that $R_t - V^\pi(s_t)$ is an unbiased, single-sample estimator of the advantage function $A^\pi(s_t,a_t)$.
6. **Application**: How does the final policy gradient expression, involving the advantage function, provide an intuitive update rule for the policy? (Hint: Consider the sign of the advantage function).