Study Material - Youtube

Document Information

- **Generated:** 2025-08-26 08:07:48
- **Source:** https://www.youtube.com/watch?v=mvANXESrhqc
- **Platform:** Youtube
- **Word Count:** 2,403 words
- **Estimated Reading Time:** ~12 minutes
- **Number of Chapters:** 4
- **Transcript Available:** Yes (analyzed from video content)

Table of Contents

Video Overview

This lecture, titled "Mathematical Foundations of Generative AI: Proximal Policy Optimization (PPO)," provides a detailed introduction to the PPO algorithm. It is presented as an advanced version of the standard policy gradient method, specifically tailored for aligning Large Language Models (LLMs) with human preferences, a process known as Reinforcement Learning from Human Feedback (RLHF).

The instructor begins by recapping the fundamentals of the policy gradient algorithm, highlighting its primary limitation: being **on-policy**. This means it requires fresh data samples for every single parameter update, which is extremely inefficient and computationally expensive for large models. To address this, the lecture introduces **Importance Sampling** as a statistical technique to enable off-policy learning, allowing the reuse of data collected from older policies. The core of the lecture is a step-by-step mathematical derivation of the importance sampling-based surrogate objective function and an explanation of why its gradient matches the original policy gradient at the point of the old policy.

Finally, the lecture identifies the key challenge with naive importance sampling—high variance and training instability when the new and old policies diverge significantly. This sets the stage for PPO, which solves this problem by constraining the policy update, ensuring that the new policy remains "proximal" or close to the old one.

Learning Objectives

Upon completing this lecture, a student will be able to: - Understand the limitations of on-policy policy gradient methods, particularly their sample inefficiency. - Grasp the core concept and mathematical formulation of Importance Sampling. - Derive the surrogate objective function for policy gradients using Importance Sampling. - Explain why the gradient of the surrogate objective is equivalent to the standard policy gradient under specific conditions. - Identify the problem of high variance and instability in off-policy methods when policies diverge. - Understand the fundamental motivation behind Proximal Policy Optimization (PPO) as a method to constrain policy updates for stable training.

Prerequisites

To fully understand the content of this lecture, students should have a foundational knowledge of: - **Reinforcement Learning (RL):** Basic concepts such as policy (π), state (s), action (a), and the RL framework. - **Policy Gradient Methods:** Familiarity with the policy gradient theorem and the REINFORCE

algorithm. - **Advantage Function:** Understanding of the advantage function, $A(s, a)$, and its role in reducing variance in policy gradients. - **Calculus and Probability:** Knowledge of gradients (∇), expectation ($\mathbb{E}$), probability distributions, and the log-derivative trick.

Key Concepts Covered

- Proximal Policy Optimization (PPO)
- Policy Gradient Algorithm
- On-Policy vs. Off-Policy Learning
- Importance Sampling
- Surrogate Loss Function
- Policy Ratio (r_t)
- Gradient Variance and Training Instability

Proximal Policy Optimization (PPO) - Deep Understanding

Introduction and Motivation

(00:11) The lecture introduces two advanced, improvised versions of policy gradient algorithms used for aligning language models with human preferences: **Proximal Policy Optimization (PPO)** and **Direct Preference Optimization (DPO)**. These are currently two of the most popular and effective algorithms for this task. This lecture will focus on the first of these, PPO.

Recap: Policy Gradient for Language Model Alignment

(01:04) To understand PPO, we must first recall the standard policy gradient framework as applied to language models.

1. **The Language Model as a Policy:** In the RLHF framework, an auto-regressive language model is treated as the **policy**, denoted by π_θ. This policy takes a state s_t (the current context or prompt) and produces a probability distribution over possible actions a_t (the next token to generate). The parameters of the language model are the policy parameters, θ.
2. **The Policy Gradient Algorithm:** The goal is to update the parameters θ to maximize a reward signal (which is derived from human preferences). This is done using a gradient ascent update rule:

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

   where α is the learning rate and $\nabla_\theta J(\theta_t)$ is the gradient of the objective function.
3. **The Policy Gradient Theorem:** The gradient of the objective function $J(\theta)$ is given by the policy gradient theorem. The video presents its core formulation:

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

   - **Intuitive Explanation:** This equation tells us how to adjust the policy parameters θ. We look at state-action pairs (s_t, a_t) that occurred in trajectories τ generated by our current policy π_θ.
     - The term $\nabla_\theta \log \pi_\theta(a_t|s_t)$ indicates the direction in parameter space that will increase the probability of taking action a_t in state s_t.
     - The **Advantage Function** $A^\pi(s_t, a_t)$ measures how much better or worse taking action a_t in state s_t was compared to the average action from that state.
     - **The Update Logic:** If the advantage $A^\pi(s_t, a_t)$ is positive (a good action), we update θ to make that action more likely. If the advantage is negative (a bad action), we update θ to make it less likely. The expectation $\mathbb{E}$ averages this effect over all possible trajectories.

The Problem: On-Policy Learning and Sample Inefficiency

(07:02) A critical limitation of the standard policy gradient algorithm is that it is **on-policy**.

> **On-Policy Learning:** An algorithm is on-policy if the data used for learning must be generated by the exact same policy that is being improved.

This creates a significant problem for training large models: 1. **Data Generation:** We generate a batch of data (trajectories, or in the case of LLMs, prompt-response pairs) using the current policy π_{θ_t}. 2. **Gradient Calculation:** We use this batch to compute a single estimate of the gradient, $\hat{g}$. 3. **Parameter Update:** We perform one gradient ascent step: $\theta_{t+1} \leftarrow \theta_t + \alpha\hat{g}$. 4. **Data Invalidation:** The policy has now changed from π_{θ_t} to $\pi_{\theta_{t+1}}$. Because the algorithm is on-policy, the entire batch of data we just generated is now considered "stale" and **must be discarded**. 5. **Repeat:** To perform the next update, we must go back to step 1 and generate a completely new batch of data with the new policy $\pi_{\theta_{t+1}}$.

This cycle is extremely **sample-inefficient**. Generating responses from a large language model is computationally expensive, and having to discard all data after a single update makes the alignment process prohibitively slow and costly.

The following Mermaid diagram illustrates this inefficient on-policy loop.

```
flowchart TD
    A["Start with policy  _ _t"] --> B["Generate trajectories  <br/>(Expensive Rollouts)"];
    B --> C["Estimate Advantage Â_t<br/>and compute Gradient ĝ"];
    C --> D["Update Policy:<br/> _t+1 ←  _t +  ĝ"];
    D --> E{"Is training complete?"};
    E -->|No| F["Discard all trajectories  "];
    F --> A;
    E -->|Yes| G["End"];
```

Figure 1: The on-policy learning loop of the standard policy gradient algorithm, highlighting the costly data generation and discard cycle.

Off-Policy Learning with Importance Sampling

To overcome the inefficiency of on-policy learning, we need a way to reuse data from older policies. This is the goal of **off-policy** methods. The key statistical tool that enables this is **Importance Sampling**.

Intuitive Foundation of Importance Sampling

(10:22) Importance sampling is a technique to estimate the expected value of a function with respect to one probability distribution, $p(x)$, while using samples generated from a different distribution, $q(x)$.

Imagine you want to find the average height of people in Country A ($p(x)$), but you only have a list of heights from people in Country B ($q(x)$). You can't just average the heights from Country B, as the populations might be different. However, if you know *how* the height distributions differ (e.g., people in Country B are, on average, 1.1 times taller), you can correct for this difference. Importance sampling formalizes this correction.

The correction factor is the ratio of the probabilities of an event under the two distributions, known as the **importance weight**.

Mathematical Analysis of Importance Sampling

(11:08) Let's formalize the intuition. Our goal is to compute:

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

However, we can only draw samples from a different distribution, $q(x)$.

Derivation: 1. **Introduce $q(x)$:** We can multiply and divide the integrand by $q(x)$ without changing the value (assuming $q(x) \neq 0$ where $p(x)f(x) \neq 0$):

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

2. **Rearrange Terms:** We can regroup the terms to form a new expectation:

$$\int_x \left(\frac{p(x)}{q(x)}f(x)\right) q(x)dx$$

3. **Formulate New Expectation:** This integral is now the definition of the expectation of a new function, $\frac{p(x)}{q(x)}f(x)$, with respect to the distribution $q(x)$:

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

The term $w(x) = \frac{p(x)}{q(x)}$ is the **importance weight**. It re-weights the samples from $q(x)$ so that their expectation matches the one we would have gotten from $p(x)$.

Applying Importance Sampling to Policy Gradients

(17:00) We can apply this technique to the policy gradient objective. We want to update the current policy π_θ but use data (trajectories) collected from an older, fixed policy $\pi_{\theta_{old}}$.

1. **Define the Surrogate Objective:** We define a new surrogate objective function, $L_{IS}(\theta)$, using importance sampling. Here, the target distribution is the new policy π_θ and the sampling distribution is the old policy $\pi_{\theta_{old}}$.

$$L_{IS}(\theta) = \mathbb{E}_{t\sim\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right]$$

2. **The Policy Ratio:** The importance weight is the ratio of the probabilities of taking action a_t in state s_t under the new and old policies. This is called the **policy ratio**, $r_t(\theta)$:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

3. **Simplified Surrogate Objective:** The objective can now be written concisely:

$$L_{IS}(\theta) = \mathbb{E}_{t\sim\pi_{\theta_{old}}}[r_t(\theta)\hat{A}_t]$$

Why does this work? (23:05) The crucial property is that the gradient of this surrogate objective $L_{IS}(\theta)$ is the same as the gradient of the original objective $J(\theta)$ when evaluated at $\theta = \theta_{old}$.

Proof: It can be verified that:

$$\nabla_\theta L_{IS}(\theta)\Big|_{\theta=\theta_{old}} = \nabla_\theta J(\theta)\Big|_{\theta=\theta_{old}}$$

This means that for small updates around θ_{old}, optimizing the surrogate objective is equivalent to optimizing the true objective. This allows us to take multiple optimization steps on $L_{IS}(\theta)$ using the same batch of data collected from $\pi_{\theta_{old}}$, making the process far more sample-efficient.

The Instability of Unconstrained Importance Sampling

(27:22) While importance sampling allows for off-policy updates, it introduces a new problem: **instability**.

- If the new policy π_θ deviates "too much" from the old policy $\pi_{\theta_{old}}$, the policy ratio $r_t(\theta)$ can become very large or close to zero.
- A very large ratio will cause the gradient estimate to have extremely **high variance**, as a few samples with large weights can dominate the entire batch.
- This leads to destructively large policy updates, causing the training process to become unstable and diverge.

To prevent this, we must ensure that the new policy π_θ stays "close" or **proximal** to the old policy $\pi_{\theta_{old}}$. This is the central idea that leads to Proximal Policy Optimization.

The Core Idea of PPO

(29:22) PPO addresses the instability of the importance sampling-based surrogate objective by adding a **constraint** or a **penalty** to the objective function. This term discourages the new policy π_θ from moving too far away from the old policy $\pi_{\theta_{old}}$.

The objective function is modified to restrict the deviation, often using a distributional divergence metric like KL-divergence. This ensures that the policy ratio $r_t(\theta)$ remains within a reasonable range, which stabilizes the training process.

The specific form of the PPO objective function, which typically involves a clipped surrogate objective, will be detailed in subsequent material. The key takeaway here is the motivation:

> **PPO = Policy Gradient + Importance Sampling (for sample efficiency) + A Constraint (for stability)**

This combination makes PPO a robust, sample-efficient, and stable algorithm, which is why it has become a cornerstone of modern RLHF for aligning large language models.

Key Mathematical Concepts

Policy Gradient Update

The fundamental update rule for policy parameters θ.

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

Policy Gradient Theorem (Expectation Form)

The gradient of the objective function $J(\theta)$.

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

Policy Gradient (Sample Estimate)

The practical estimation of the gradient from a batch of data.

$$\hat{g} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i - 1} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{A}_{i,t}$$

Importance Sampling Transformation

Relates the expectation under a target distribution $p(x)$ to a sampling distribution $q(x)$.

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

Surrogate Objective with Importance Sampling

The off-policy objective function used in PPO's derivation.

$$L_{IS}(\theta) = \mathbb{E}_{t\sim\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right] = \mathbb{E}_{t\sim\pi_{\theta_{old}}}[r_t(\theta)\hat{A}_t]$$

where $r_t(\theta)$ is the policy ratio.

Self-Assessment for This Video

1. **Conceptual Understanding:**
   - What does it mean for a reinforcement learning algorithm to be "on-policy"? Why is this a problem for aligning large language models?
   - Explain the intuition behind Importance Sampling using an analogy. What is the "importance weight"?
   - How does Importance Sampling allow us to create an "off-policy" algorithm from the policy gradient method?
   - What is the main risk or instability introduced by using the importance sampling surrogate objective? Why does this happen?
2. **Mathematical Reasoning:**
   - Write down the formula for the policy ratio $r_t(\theta)$. What do the numerator and denominator represent?
   - Show that the gradient of the surrogate loss function $\nabla_\theta L_{IS}(\theta)$ is equal to the true policy gradient $\nabla_\theta J(\theta)$ when evaluated at $\theta = \theta_{old}$.
   - Describe a scenario where the policy ratio $r_t(\theta)$ would become very large. What is the consequence of this for the gradient update?
3. **Application and Synthesis:**
   - What is the core principle that PPO adds to the importance-sampled policy gradient to ensure stability?
   - Based on the lecture, draw a diagram that shows the progression of ideas from the standard Policy Gradient algorithm to the motivation for PPO.

Key Takeaways from This Video

- **Standard Policy Gradient is Inefficient:** It is an on-policy algorithm, requiring new data for every update, which is too expensive for large models.
- **Importance Sampling Enables Off-Policy Updates:** By re-weighting samples from an old policy with the policy ratio $r_t(\theta) = \frac{\pi_\theta}{\pi_{\theta_{old}}}$, we can reuse old data for multiple updates, dramatically improving sample efficiency.
- **Off-Policy Methods Can Be Unstable:** If the new policy π_θ becomes too different from the old policy $\pi_{\theta_{old}}$, the policy ratio can explode, leading to high variance in gradient estimates and unstable training.

- **PPO Constrains the Policy Update:** The fundamental innovation of PPO is to constrain the surrogate objective function, ensuring the updated policy stays "proximal" to the old one, thereby gaining sample efficiency without sacrificing stability.