Lecture on Proximal Policy Optimization

Introduction
In 2017, OpenAI introduced a new family of policy gradient methods for reinforcement learning called **Proximal Policy Optimization (PPO)**. This algorithm quickly became one of the most widely used RL methods due to its balance of simplicity, stability, and efficiency.

PPO optimizes a **surrogate objective** with **clipped probability ratios**, ensuring that policy updates remain stable without requiring complex second-order optimization. Unlike standard policy gradient methods, which update policies once per data sample, PPO allows multiple **minibatch updates**, improving sample efficiency. Empirical results have shown that PPO outperforms other policy gradient methods in tasks like robotic control and Atari games. It is also the foundation of widely used AI models like **ChatGPT**.

To fully understand why PPO was introduced, we first need to look at its predecessor: **Trust Region Policy Optimization (TRPO).**

What is TRPO and Why Was It Inefficient?

Before PPO, **Trust Region Policy Optimization (TRPO)** was a leading approach for improving policy gradient methods. TRPO was designed to ensure **stable updates** by preventing the new policy from deviating too much from the previous one.

How TRPO Works

- In standard policy gradient methods, large updates can cause drastic policy shifts, leading to instability.
- TRPO solves this by introducing a **trust region constraint**, which limits how much the policy can change in each update.
- It does this using **Kullback-Leibler (KL) divergence**, a measure of how different two probability distributions are.

Mathematically, TRPO optimizes the objective:

maxθE[πθ(a|s)πθold(a|s)A(s,a)]\max_{\theta} \mathbb{E} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A(s, a) \right]

subject to the constraint:

DKL(πθ||πθold)≤δD_{KL}(\pi_{\theta} || \pi_{\theta_{\text{old}}}) \leq \delta

where:

- πθ\pi_{\theta} is the new policy, and πθold\pi_{\theta_{\text{old}}} is the previous policy.
- A(s,a)A(s, a) is the **advantage function**, which tells how good an action is compared to the average.

Rishikesh Vadodaria

- DKLD_{KL} is the KL divergence, ensuring the update stays within a safe limit δ\delta.

Why Was TRPO Inefficient?

While TRPO improved stability, it had some drawbacks:

1. **Complexity** – It requires **second-order optimization** (computing Hessians), which is computationally expensive.
2. **Hard to Implement** – The KL divergence constraint requires a **constrained optimization solver**, making it harder to apply.
3. **Inefficient Data Use** – TRPO processes data once per batch, meaning it doesn't fully utilize available training samples.

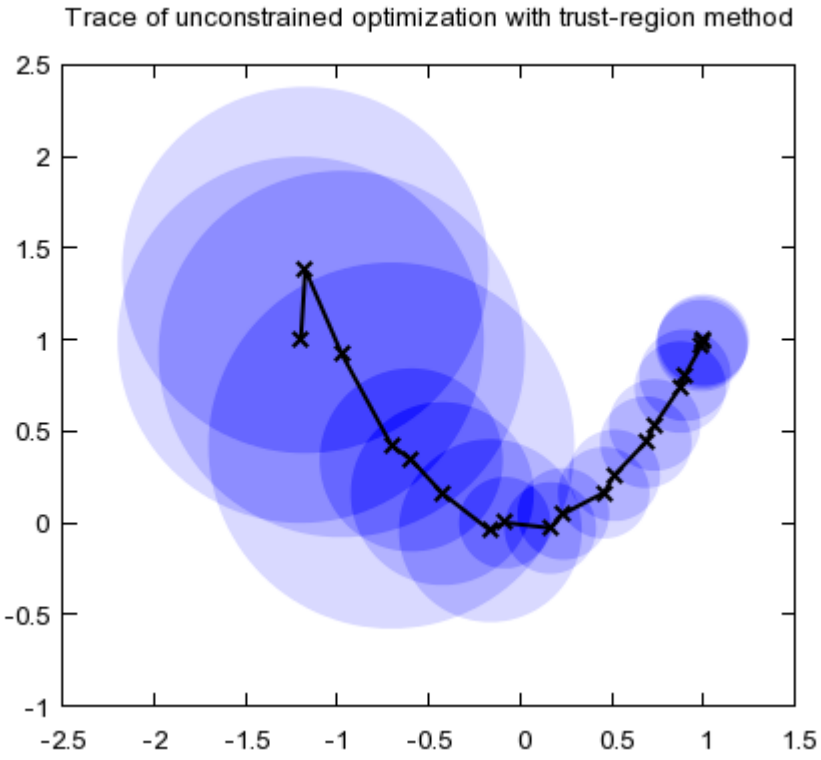
These limitations led to the development of **PPO**, which retains the benefits of TRPO while being simpler and more efficient.

Transition to PPO

PPO builds upon TRPO by **removing the need for complex constraints** and using **clipping** to stabilize updates. Instead of explicitly enforcing a trust region, PPO introduces a modified objective function that **penalizes excessive policy updates** while allowing multiple minibatch updates for better sample efficiency.

In the next section, we will dive deeper into **how PPO works** and why it became the standard for modern RL applications.

This script keeps the lecture engaging, structured, and easy to follow. Let me know if you want any modifications!


Trace of unconstrained optimization with trust-region method
2.5
2
1.5
1
0.5
0
-0.5
-1
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5