UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Proximal Policy Optimization

Presented By: Abel Abebe Bzuayene
Presented to: Prof. Alberto Castellini
July 30, 2024

Proximal Policy Optimization (OpenAI)

"PPO has become the default reinforcement learning algorithm at OpenAI because of its ease of use and good performance"

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms.

https://arxiv.org/pdf/1707.06347

https://blog.openai.com/openai-baselines-ppo/

Table Content

1. Introduction

Proximal Policy Optimization (PPO) is a reinforcement learning algorithm designed to improve the stability and efficiency of policy optimization.

Purpose: Enhance scalability, data efficiency, and robustness in reinforcement learning.

Comparison: Superior to deep Q-learning, vanilla policy gradients, and trust region methods.

Key Feature: Uses a clipped objective function for simplified optimization and robust performance.

Method: Samples data and performs multiple optimization steps.

Results: Outperforms previous algorithms in continuous control tasks.

2. **Background**: Policy Optimization
2.1 Policy Gradient Methods

- Policy gradient methods optimize reinforcement learning policies by estimating and applying the gradient of the policy's performance. The gradient estimator is given by:

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right],$$

where π_θ represents a stochastic policy, and $\hat{A}_t$ is an estimator of the advantage function at time t. The expectation $\hat{\mathbb{E}}_t[\cdot]$ denotes averaging over a batch of samples.

To optimize the policy, the objective function $L_{PG}(\theta)$ is:

$$L_{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right].$$

- **What is the limitation of PG method?**
- In policy gradient methods, optimizing the loss function can lead to large and destabilizing policy changes if the same trajectory is used repeatedly for updates.

- **Any solution?** To address this, researchers have focused on using the old policy as a reference and introducing constraints to limit update size.

2.2 **Trust Region Methods**

Trust Region Policy Optimization (TRPO) ensures that policy updates stay within a safe range by limiting the deviation between the old and new policies using KL-divergence.

It optimizes the objective function while respecting this constraint to maintain stability.

TRPO optimizes a policy by maximizing:

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

- **Alternatively,** a penalty approach is suggested

$$\underset{\theta}{\text{maximize}}\, \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right]$$

- **Problems with TRPO?** It requires second-order derivatives , hard to choose a single value of β that performs well across different problems, complicated, and is not compatible with architectures that include noise (such as dropout) or parameter sharing.
- **Any solution?**

3. Clipped Surrogate Objective

The Clipped Surrogate Objective improves upon the Conservative Policy Iteration (CPI) objective to prevent excessively large policy updates.

Conservative Policy Iteration Objective:

$$L_{CPI}(\theta) = \mathbb{E}_t \left[r_t(\theta) \hat{A}_t \right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and $r(\theta_{\text{old}}) = 1$. Without constraints, maximizing L_{CPI} can lead to overly large policy changes.

Clipped Objective:

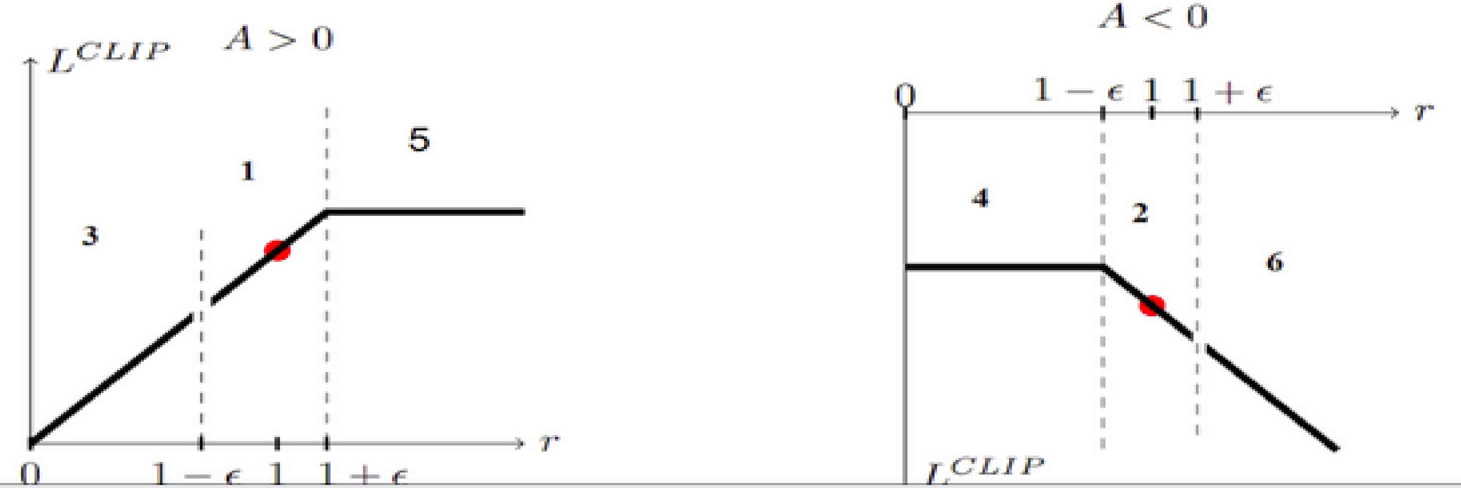
To address this, PPO introduces:

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

where ϵ is a hyperparameter (e.g., $\epsilon = 0.2$). The clipping function limits $r_t(\theta)$ to the interval $[1-\epsilon, 1+\epsilon]$, reducing the incentive for large policy changes. The minimum of the clipped and unclipped objectives ensures that L_{CLIP} provides a lower bound on the original objective and is less sensitive to large deviations in $r_t(\theta)$.

Assumption $p_t(\theta)$ = $r_t(\theta)$

	$p_t(\theta) > 0$	A_t	Return Value of min	Objective is Clipped	Sign of Objective	Gradient
1	$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$+$	$p_t(\theta)A_t$	no	$+$	✓
2	$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$-$	$p_t(\theta)A_t$	no	$-$	✓
3	$p_t(\theta) < 1-\epsilon$	$+$	$p_t(\theta)A_t$	no	$+$	✓
4	$p_t(\theta) < 1-\epsilon$	$-$	$(1-\epsilon)A_t$	yes	$-$	**0**
5	$p_t(\theta) > 1+\epsilon$	$+$	$(1+\epsilon)A_t$	yes	$+$	**0**
6	$p_t(\theta) > 1+\epsilon$	$-$	$p_t(\theta)A_t$	no	$-$	✓



Source : https://huggingface.co/learn/deep-rl-course/unit8/visualize

4. Adaptive KL Penalty Coefficient

- This approach adds a KL divergence penalty to the policy objective, adjusting the penalty coefficient β to achieve a target KL divergence d_targ . Although it performed worse than the clipped surrogate objective in experiments, it remains a useful baseline.

Procedures:

1. Optimize KL-Penalized Objective:

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right]$$

2. Adjust Penalty Coefficient:

$$\text{Compute } d = \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\mathrm{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$$

- If $d < d_{\mathrm{targ}}/1.5$, $\beta \leftarrow \beta/2$
- If $d > d_{\mathrm{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$

- The updated β is used for the next policy update. The factors 1.5 and 2 are heuristic, and β adjusts quickly, making the initial value less critical.

5. Algorithm

The researchers modify the typical policy gradient by using surrogate losses (L_CLIP or L_KLPEN) instead of L_PG and apply multiple steps of stochastic gradient ascent.

In their neural network architecture, which shares parameters between the policy and value function, they combine: Policy Surrogate Objective either L_CLIP or L_KLPEN, Value Function Error Term (squared error loss), Entropy Bonus(for exploration).

The resulting objective function integrates these components to guide the optimization.

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$$

- where L(VF) is a squared-error loss (V — V^{targ})^2 and S denotes an entropy bonus.
- The advantage of policy gradient is:

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$$

- where t specifies the time index in [0, T], the researchers use generalized advantage estimate (GAE), which reduces the equation to:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$
$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

- Finally, a pseudo code of the algorithm is:

Algorithm 1 PPO, Actor-Critic Style

for iteration$=1, 2, \ldots$ **do**
    for actor$=1, 2, \ldots, N$ **do**
        Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
        Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
    $\theta_{\text{old}} \leftarrow \theta$
end for

5.1 Compare REINFORCE with Bs and PPO

Algorithm 2 Episodic REINFORCE with baseline

```
Initialize policy network π_θ
Initialize baseline network b_w
for iteration = 1, 2, ..., num_episodes do
    Generate an episode S_0, A_0, R_1, S_1, ..., S_{T-1}, A_{T-1}, R_t following π_θ
    for t = 0, 1, 2, ..., T - 1 do
        G_t = Σ_{k=t+1}^{T} γ^{k-t-1} R_k
    end for
    L(θ) = -(1/T) Σ_{t=0}^{T-1} (G_t - b_w(S_t)) ln π_θ(A_t|S_t)
    L(w) = -(1/T) Σ_{t=0}^{T-1} (G_t - b_w(S_t))^2
    Update π_θ using Adam(∇_θ L(θ))
    Update b_w using Adam(∇_w L(w))
end for
```

Where:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$L(\theta) = -\frac{1}{T} \sum_{t=0}^{T-1} (G_t - b_w(S_t)) \ln \pi_\theta(A_t|S_t)$$

$$L(w) = -\frac{1}{T} \sum_{t=0}^{T-1} (G_t - b_w(S_t))^2$$

- Source: https://medium.com/@ym1942/proximal-policy-optimization-tutorial-f722f23beb83

Algorithm 5 Episodic PPO

Initialize policy network π_θ

Initialize baseline network b_w

for $iteration = 1, 2, \ldots, num\_episodes$ **do**

  Generate an episode $s_0, a_0, r_1, s_1, \ldots, s_{T-1}, a_{T-1}, r_T$ following π_θ

  for $t = 0, 1, 2, \ldots, T-1$ **do**

    $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

  end for

  Compute advantage estimates $A_t = (G_t - b_w(s_t))$

  for $epoch = 1, 2, \ldots, num\_epochs$ **do**

    Compute the objective function

    $L^{CLIP}(\theta) = \mathbb{E}_\pi[min(r(\theta)A_t, clip(r(\theta), 1-\epsilon, 1+\epsilon)A_t)]$

    $L(w) = -\frac{1}{T}\sum_{t=0}^{T-1}(G_t - V_w(S_t))^2$

    Update π_θ using $Adam(\nabla_\theta L^{CLIP}(\theta))$

    Update b_w using $Adam(\nabla_w L(w))$

  end for

end for

Observation

- Objective Function: PPO uses a clipped objective for policy updates, unlike REINFORCE with baseline.
- Parameter Updates: PPO updates model parameters multiple times per training episode (multiple epochs), which is challenging for traditional policy gradient methods due to instability from policy deviations.
- Clipping Advantage: PPO's clipped objective prevents excessive policy changes, making multiple epoch training more stable and efficient.

6. Experiments

6.1 Comparison between Clipping and KL Penalty

- The researchers compared several different surrogate objectives using different hyperparameters.

No clipping or penalty:	$L_t(\theta) = r_t(\theta)\hat{A}_t$
Clipping:	$L_t(\theta) = \min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta)), 1-\epsilon, 1+\epsilon)\hat{A}_t$
KL penalty (fixed or adaptive)	$L_t(\theta) = r_t(\theta)\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\mathrm{old}}}, \pi_\theta]$

Policy Model Structure: Fully connected MLP with two hidden layers (64 units each) and tanh activations.

Outputs: Mean of a Gaussian distribution with variable standard deviations.

No Parameter Sharing: Policy and value functions did not share parameters.

No Entropy Bonus: Not used in the model.

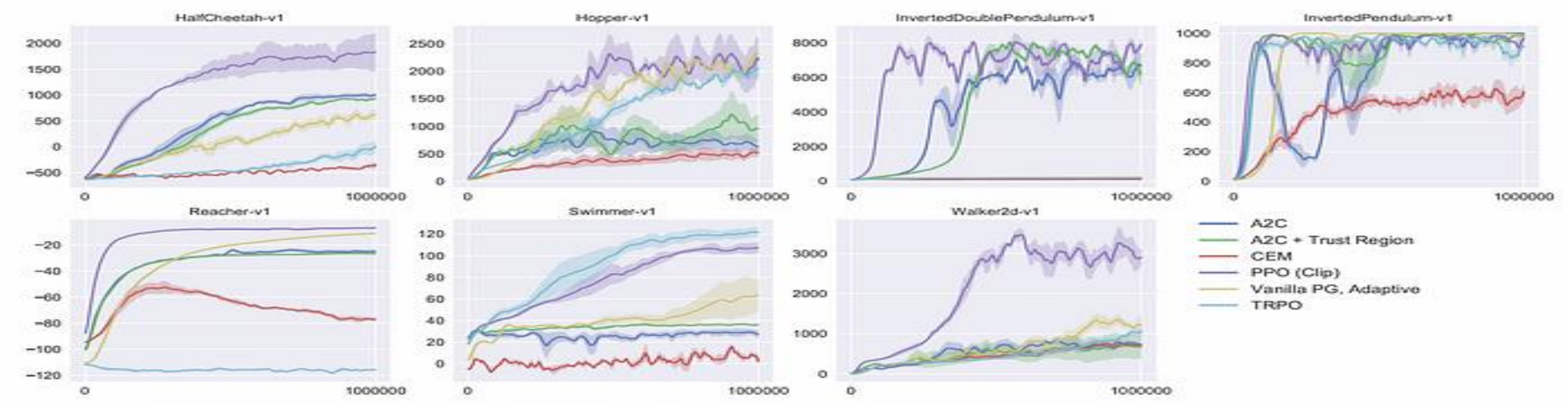
Tasks: Each algorithm was run on all 7 environments, with 3 random seeds on each

Performance Evaluation: Average reward of the last 100 episodes, scaled (random policy = 0, best result = 1).

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

6.2 Comparison to Other Algorithms in the Continuous Domain

- The researchers compared PPO with the clipped surrogate objective and $\epsilon = 0.2$ to several other methods considered effective for continuous problems



6.3 Comparison to Other Algorithms on the Atari Domain

- They also ran PPO in the Atari environment and compared the average episode reward over the entire training period and the average episode reward over the last 100 episodes.

	A2C	ACER	PPO	Tie
(1) avg. episode reward over all of training	1	18	**30**	0
(2) avg. episode reward over last 100 episodes	1	**28**	19	1

7. Conclusion

Innovative Algorithm: PPO is a key advancement in the policy gradient domain.

Problems Addressed: Tackles issues of instability and sample inefficiency in previous policy gradient methods.

Key Feature: Introduces clipping to stabilize policy updates and improve efficiency.

Impact: Enabled the development of many new policy gradient-based algorithms, significantly advancing reinforcement learning.


Thank you
35
30%
20%
10%