Deep Learning (1470)

Randall Balestriero

Class 21: TRPO and PPO

Recap!

On-Policy and Off-Policy Learning

RL algorithms collect experiences and learn from these experiences

On-Policy Algorithms have to collect experiences with the policy they are learning

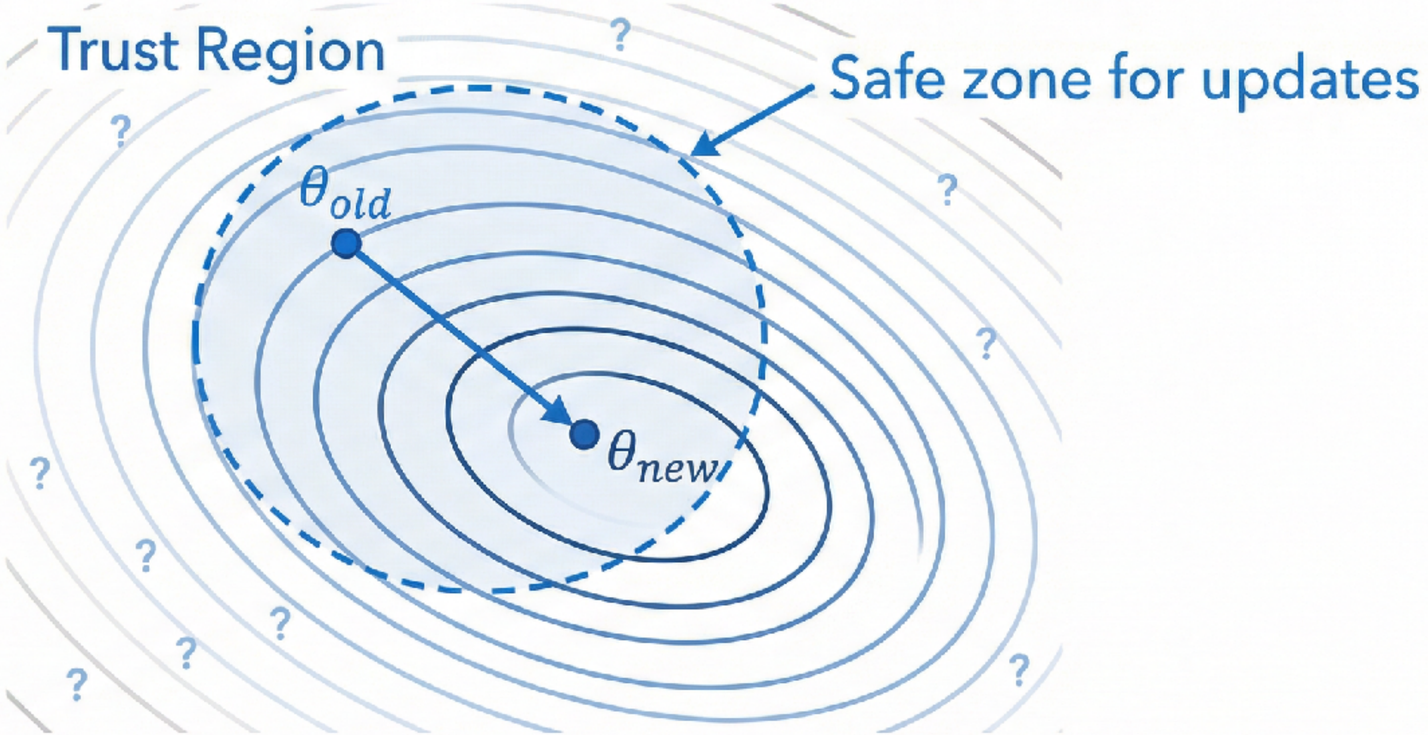
Off-Policy Algorithms can use **any** policy to collect experiences

Review: On + Off-Policy Learning

	On-Policy	Off Policy
Summary	Learns policy/value function based on policy used during training	Learns policy independent of policy used to collect experiences during training
Algorithms	SARSA, Policy Gradient, Actor Critic, PPO	Q-Learning, Off-policy Actor-Critic, Deep Deterministic Policy Gradient (DDPG)

Any Idea?

The Key Insight: Trust Regions



Core Idea: Only trust our local approximation within a small region

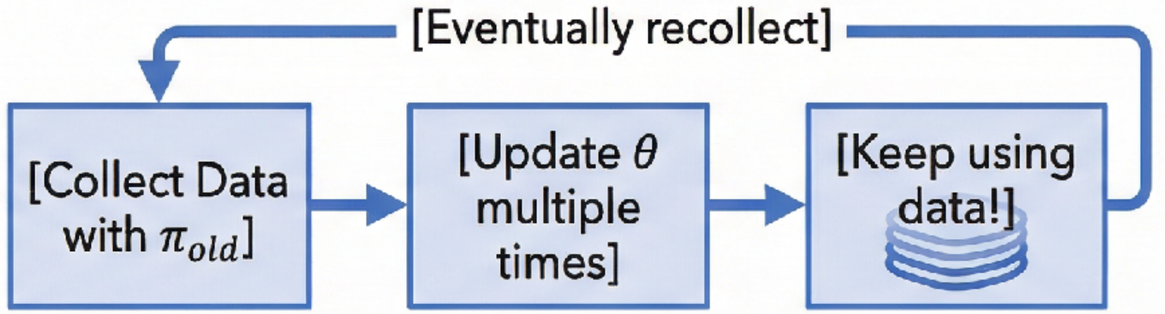
- Local linear/quadratic approximations are only accurate nearby
- Constrain updates to stay within the "trusted" neighborhood
- Guarantees monotonic improvement (with high probability)

The Off-Policy Trick: Reusing Old Data

Pure On-Policy (e.g., REINFORCE, A2C)

- Must collect new data after EVERY update
- Old trajectories are "stale" - from wrong distribution
- Sample inefficient
- Simple but wasteful

Off-Policy via **Importance Sampling** (PPO/TRPO)



- Reuse data for K epochs of updates
- Correct distribution mismatch via importance ratio
- $r(\theta) = \pi_\theta(a|s) \,/\, \pi_{old}(a|s)$
- Much more sample efficient!

The importance ratio $r(\theta)$ corrects for the fact that data came from π_{old}, not π_θ

TRPO: Trust Region Policy Optimization

Schulman et al., 2015

$$\underset{\theta}{\text{maximize}} \quad L(\theta) = \mathbb{E}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta^{\text{old}}}(a|s)} \cdot A(s,a)\right]$$

$$\text{subject to:} \quad D_{\text{KL}}(\pi_{\theta^{\text{old}}} \,\|\, \pi_\theta) \leq \delta$$

Importance sampling ratio $r(\theta)$

KL divergence constraint (trust region!)

Advantage function

Maximum allowed divergence (e.g., 0.01)

The KL constraint ensures the new policy doesn't deviate too far from the old one

Importance Sampling: The Math

We want to estimate:

$$\mathbb{E}_{a\sim\pi_\theta}[A(s,a)]$$

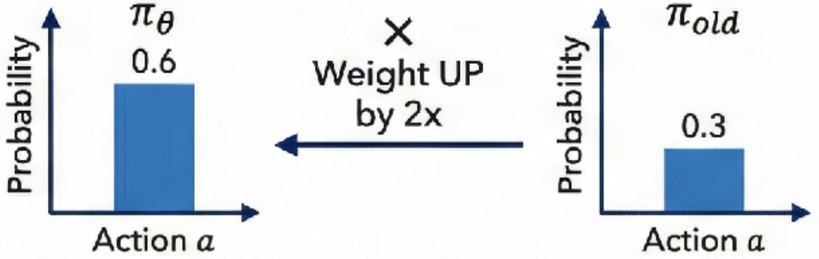
⚠ But our data came from π_{old}!

✓ Solution: Importance Sampling

$$\mathbb{E}_{a\sim\pi_\theta}[A(s,a)] = \mathbb{E}_{a\sim\pi_{old}}\left[\frac{\pi_\theta(a|s)}{\pi_{old}(a|s)}\cdot A(s,a)\right]$$
$$= \mathbb{E}_{a\sim\pi_{old}}[r(\theta)\cdot A(s,a)]$$

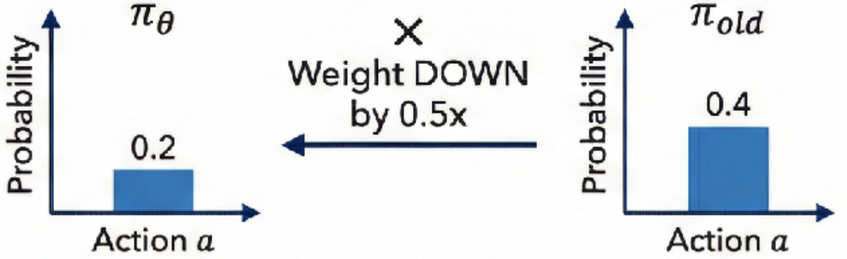
Intuition

If $\pi_\theta(a) = 0.6$ and $\pi_{old}(a) = 0.3$, then $r(\theta) = 2$



Action a was sampled less often than π_θ would
→ weight it UP by 2x

If $\pi_\theta(a) = 0.2$ and $\pi_{old}(a) = 0.4$, then $r(\theta) = 0.5$



Action a was sampled more often than π_θ would
→ weight it DOWN by 0.5x

⚠ **Danger:** If π_θ drifts too far from π_{old}, importance weights explode!
That's why we need clipping (PPO) or KL constraint (TRPO)

Enter PPO: A Simpler Approach

Schulman et al., 2017

TRPO Recap

- Constrained optimization
- Second-order methods (Fisher matrix)
- Conjugate gradient solver
- Line search
- Complex implementation

Simplification

PPO Solution

- First-order methods only!
- Simple clipping mechanism
- No KL constraint needed
- Standard SGD/Adam
- Easy to implement

"PPO attains the data efficiency and reliable performance of TRPO while using only first-order optimization"

The Complete PPO Objective

$$L^{\text{PPO}}(\theta) = \mathbb{E}\Big[\, L^{\text{CLIP}}(\theta) - c_1 \cdot L^{\text{VF}}(\theta) + c_2 \cdot S[\pi_\theta] \,\Big]$$

$L^{\text{CLIP}}(\theta)$ - Policy Loss

The clipped surrogate objective

Encourages good actions, discourages bad

$L^{\text{VF}}(\theta)$ - Value Function Loss

$$L^{\text{VF}} = (V_\theta(s) - V_{\text{target}})^2$$

Trains the critic network

c_1 typically = 0.5

$S[\pi_\theta]$ - Entropy Bonus

$$S = -\sum \pi(a|s) \log \pi(a|s)$$

Encourages exploration

c_2 typically = 0.01

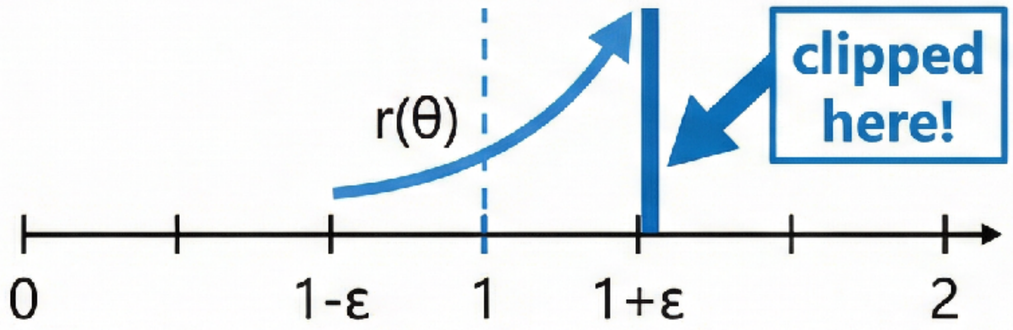
In actor-critic setup, policy and value networks often share parameters

PPO: The Clipped Objective

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r(\theta)\cdot A, \text{clip}(r(\theta), 1-\varepsilon, 1+\varepsilon)\cdot A\right)\right]$$

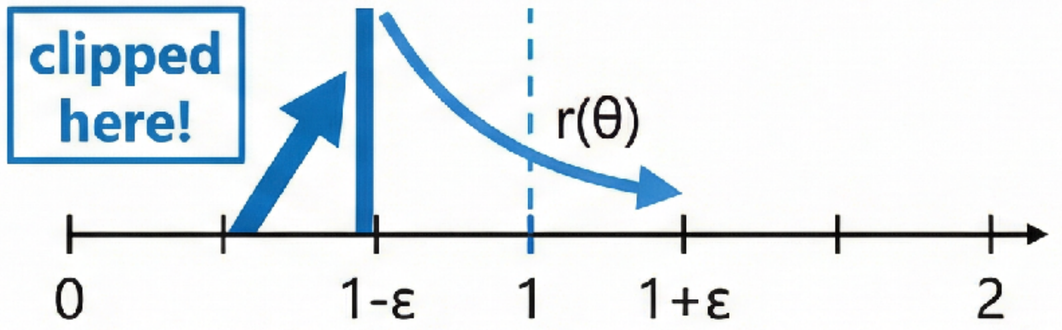
$$\text{where } r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}$$

When A > 0 (good action)



Can't get more credit than (1+ε)·A ✓

When A < 0 (bad action)



Can't be penalized more than (1-ε)·A ✗

Typical value: ε = 0.2

TRPO vs PPO: Head-to-Head

	TRPO	PPO
Constraint	Hard KL constraint $D_{KL} \leq \delta$	Soft clipping $r(\theta) \in [1-\varepsilon, 1+\varepsilon]$
Optimization	Second-order (Natural gradient)	First-order (Adam/SGD)
Implementation	Complex ~500 lines	Simple ~50 lines
Computation	Expensive (Fisher matrix, CG)	Cheap (Standard backprop)
Sample Efficiency	Similar	Similar
Performance	Strong	Strong (often slightly better)
Popularity	Research	Industry standard (OpenAI, etc.)

PPO: Same benefits as TRPO, much simpler to implement and tune

The Evolution of Policy Optimization

1992:

1992: REINFORCE (Williams)

- The OG policy gradient
- Simple but high variance

$$\nabla J = \mathbb{E}[\nabla \log \pi \cdot R]$$

Add baseline to reduce variance

2000s:

2000s: Actor-Critic

- Critic estimates value
- Reduces variance
- But still on-policy, unstable step sizes

Constrain the updates!

2015:

2015: TRPO (Schulman)

- Trust region constraint
- Monotonic improvement guarantee
- Complex: needs conjugate gradient

Simplify!

2017:

2017: PPO (Schulman)

- Simple clipping trick
- First-order optimization
- The industry standard ⭐

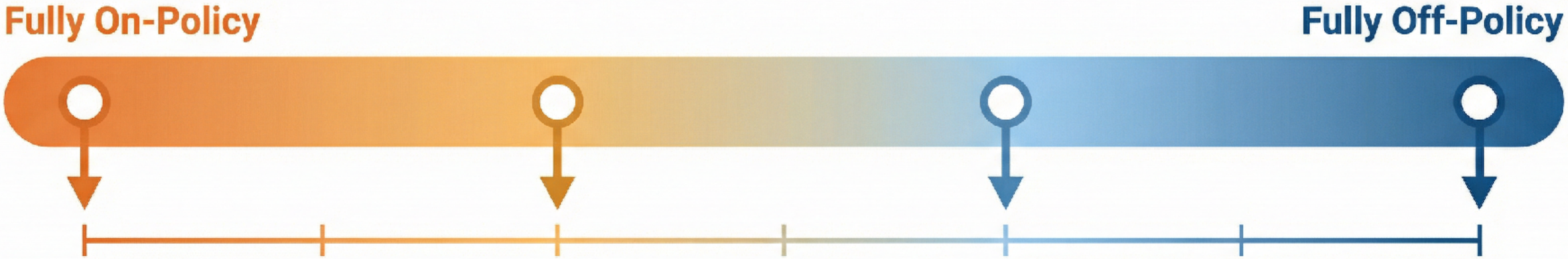
Scale it up!

2019+:

2019+: PPO at Scale

- OpenAI Five, RLHF
- ChatGPT, Claude, etc.
- Distributed, massive compute

The On-Policy to Off-Policy Spectrum



REINFORCE, A2C

- Data used once, then discarded
- No importance correction

TRPO, PPO

- Data reused for K epochs
- Importance sampling correction
- π_{old} is recent (same iteration)

Off-policy Actor-Critic

- Replay buffer of past data
- π_{old} can be many updates old

DQN, SAC, DDPG

- Large replay buffers
- Data from very old policies
- No explicit importance correction (Q-learning trick)

Key trade-off: More off-policy → more sample efficient, but harder to stabilize

Algorithm Selection Guide: Which One to Use?

Algorithm	Action Space	Sample Eff.	Stability	Best For
DQN	Discrete ONLY	High ⭐⭐⭐ (replay)	Medium	• Atari, discrete games • Simple to implement
A2C	Both	Low ⭐ (on-policy)	High ⭐⭐⭐	• Fast prototyping • Parallel environments
PPO	Both	Medium ⭐⭐	High ⭐⭐⭐	• General purpose ✅ • **DEFAULT CHOICE**
TRPO	Both	Medium ⭐⭐	High ⭐⭐⭐	• Research/theory • When guarantees needed
DDPG/TD3 SAC	Continuous ONLY	High ⭐⭐⭐ (replay)	Low Med-High	• Robotics, control • Continuous actions

🎯 **Default Choice: Start with PPO**
- Works for both discrete and continuous
- Stable and reliable
- Easy to tune
- Industry standard

See you on Friday!