

CS-566 Deep Reinforcement Learning

Improved Policy-Based Learning I: Actor-Critic Methods



Nazar Khan
Department of Computer Science
University of the Punjab

Improving Policy Gradients: Reducing High Variance

- ▶ Vanilla (episodic) policy gradient methods suffer from **high variance**.
- ▶ **Key Enhancements to Reduce Variance:**
 - ▶ **Actor-Critic:** Add a value-based critic using TD bootstrapping to guide updates
 - ▶ **Baseline Subtraction:** Use an *advantage function* to reduce variance of returns
 - ▶ **Trust Regions:** Constrain large policy updates for stability (e.g., TRPO, PPO)
 - ▶ **Exploration Strategies:** Encourage high-entropy policies to escape local minima
- ▶ These techniques dramatically improved the practicality and performance of policy-based RL.

Actor-Critic Bootstrapping: Motivation

Actor-Critic combines policy-based and value-based methods:

- ▶ **Actor:** policy $\pi_\theta(a|s)$ (chooses actions)
- ▶ **Critic:** value function $V_\phi(s)$ (evaluates states/actions)

Why Actor-Critic?

- ▶ REINFORCE: low bias but **very high variance**
- ▶ Full-episode sampling \Rightarrow updates vary wildly
- ▶ Actor-Critic keeps low bias, but reduces variance via value bootstrapping

Actor-Critic has become a core RL paradigm (A3C, PPO, SAC, DDPG, . . .).

Where Does Variance Come From?

Sources of variance in policy gradients

1. **Cumulative return variance**: Full episode rewards vary greatly
2. **Gradient estimate variance**: Stochastic action samples \Rightarrow noisy gradient

Solutions:

- ▶ **Bootstrapping** for lower reward variance
- ▶ **Baseline subtraction** (advantage) for lower gradient variance

The critic $V_\phi(s)$ supplies both bootstrapping and baselines.

Network Structure: Actor and Critic

Value function used in Actor-Critic:

$$V_\phi(s)$$

Parameterization options

- ▶ Separate networks: θ (actor), ϕ (critic)
- ▶ Shared body + two heads (policy head + value head)

Notation:

- ▶ Policy parameters θ
- ▶ Value-network parameters ϕ

Actor improves policy; critic provides value estimates for stability.

Temporal Difference Bootstrapping

Sampling full episodes gives high variance: many possible trajectories \Rightarrow updates unstable.

Bootstrapping idea:

- ▶ Use TD targets to estimate returns before episode ends
- ▶ n -step bootstrapped target

$$\hat{Q}_n(s_t, a_t) = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V_\phi(s_{t+n})$$

- ▶ Interpolates between MC (high variance) and TD (higher bias)

Actor-Critic: Learning Updates

Value function loss:

$$\mathcal{L}(\phi) = (\hat{Q}_n(s_t, a_t) - V_\phi(s_t))^2$$

Policy gradient update:

$$\nabla_\theta J(\theta) = \hat{Q}_n(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

Policy uses bootstrapped estimate instead of full return R .

Actor-Critic with Bootstrapping: Pseudocode

Algorithm 1 Actor-Critic with n -step Bootstrapping

Initialize policy π_θ and value network V_ϕ

repeat

for each episode $i = 1..M$ **do**

 Sample trajectory $\tau = \{s_0, a_0, r_0, \dots, s_T\}$

for $t = 0..T - 1$ **do**

$\hat{Q}_n(s_t, a_t) = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V_\phi(s_{t+n})$ \triangleright MC + n-step TD

end for

end for

$\phi \leftarrow \phi - \alpha \nabla_\phi \sum_t (\hat{Q}_n - V_\phi(s_t))^2$

\triangleright Updates in batch mode

$\theta \leftarrow \theta + \alpha \sum_t [\hat{Q}_n \nabla_\theta \log \pi_\theta(a_t | s_t)]$

until convergence

Key Takeaways

Actor-Critic Summary

- ▶ Actor learns policy; critic learns value function
- ▶ Reduces variance vs. vanilla policy gradients
- ▶ Bootstrapping \Rightarrow better value estimates
- ▶ Uses n -step returns between MC and TD
- ▶ Foundation for modern RL (A3C, PPO, SAC, etc.)

Baseline Subtraction with Advantage Function

Goal: Reduce variance of policy gradient estimates.

Key idea: Subtract a baseline from returns to lower variance *without changing the expectation*.

Example:

- ▶ Action returns in a state: 65, 70, 75
- ▶ All positive → vanilla PG pushes **all** action probabilities up
- ▶ Better: push above-average actions up (75) and below-average down (65)

Baseline: Use value function $V(s)$

$$A(s, a) = Q(s, a) - V(s)$$

Advantage: Measures improvement of a over expected value of s .

Advantage Function in Practice

Combine bootstrap estimate with baseline:

$$\hat{A}_n(s_t, a_t) = \hat{Q}_n(s_t, a_t) - V_\phi(s_t)$$

$$\nabla_\theta J(\theta) = \hat{A}_n(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

Interpretation:

- ▶ Positive advantage \rightarrow increase probability of action
- ▶ Negative advantage \rightarrow decrease probability of action
- ▶ Zero advantage \rightarrow no change

Actor-Critic with Baseline + Bootstrapping

Actor critic: Policy + value function

Algorithm structure:

- ▶ Collect trajectory
- ▶ Compute n -step target and advantage
- ▶ Update critic (value function)
- ▶ Update actor (policy)

Losses:

$$\mathcal{L}_V = (\hat{A}_n)^2 \quad \mathcal{L}_\pi = -\hat{A}_n \log \pi_\theta(a|s)$$

Pseudocode: Actor-Critic with Advantage

Initialize policy π_θ , value function V_ϕ

while not converged **do**

for each episode **do**

 Collect trajectory τ

for each time t **do**

$$\hat{Q}_n = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V_\phi(s_{t+n})$$

$$\hat{A}_n = \hat{Q}_n - V_\phi(s_t)$$

end for

end for

$$\phi \leftarrow \phi - \alpha \nabla_\phi \sum_t \hat{A}_n^2$$

$$\theta \leftarrow \theta + \alpha \sum_t \hat{A}_n \nabla_\theta \log \pi_\theta(a_t | s_t)$$

end while

General Policy Gradient Formulation

$$\nabla_{\theta} J(\theta) = \mathbb{E} \left[\sum_t \Psi_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right]$$

Choices for target Ψ_t :

$$\Psi_t = \hat{Q}_{MC} \qquad \qquad \qquad \text{Monte Carlo}$$

$$\Psi_t = \hat{Q}_n \qquad \qquad \qquad n\text{-step bootstrapping}$$

$$\Psi_t = \hat{A}_{MC} \qquad \qquad \qquad \text{Advantage (MC)}$$

$$\Psi_t = \hat{A}_n \qquad \qquad \qquad \text{Advantage + bootstrapping}$$

$$\Psi_t = Q_{\phi}(s, a) \qquad \qquad \qquad \text{Critic estimated } Q$$

A3C (Asynchronous Advantage Actor-Critic)

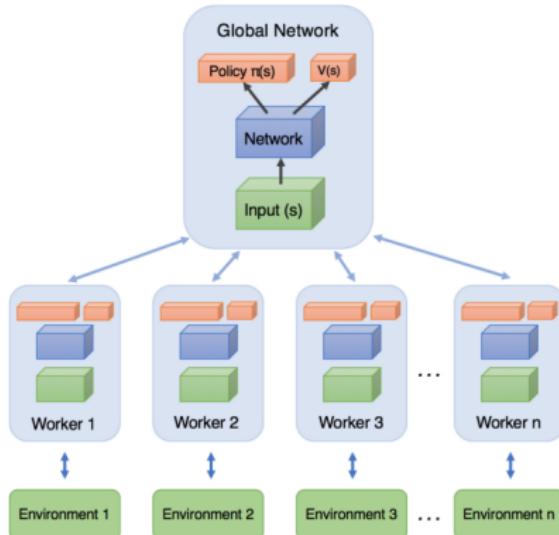
Why A3C?

- ▶ Extends advantage actor-critic
- ▶ Uses many parallel agents to stabilize learning
- ▶ Neural networks estimate both V and A and π
- ▶ Asynchronous updates to shared parameters

Benefits:

- ▶ Efficient experience collection
- ▶ Reduced correlation between samples
- ▶ Strong performance on Atari and continuous control

A3C Architecture



- ▶ Shared CNN feature extractor
- ▶ Separate value and policy heads
- ▶ Parallel actors update global params asynchronously