Policy Gradient Methods Tutorial

September 21, 2025

1 Introduction

Policy gradient methods directly optimize a parameterized policy by ascending the gradient of expected return. Unlike value-based methods that derive policies from action-value estimates, policy gradients handle continuous action spaces, stochastic policies, and incorporate constraints such as entropy regularization with ease.

2 Theory and Formulas

2.1 Objective and Policy Gradient Theorem

Let a stochastic policy $\pi_\theta(a \mid s)$ be parameterized by θ. The objective is the expected discounted return

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\Big[\sum_{t=0}^{T} \gamma^t r_{t+1}\Big], \tag{1}$$

where τ denotes trajectories sampled under π_θ. The policy gradient theorem states

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta}\big[\nabla_\theta \log \pi_\theta(a \mid s)\, Q^{\pi_\theta}(s,a)\big], \tag{2}$$

with d^{π_θ} the discounted state visitation distribution.

2.2 Baselines and Advantage Estimates

To reduce variance, a baseline $b(s)$ can be subtracted without bias:

$$\nabla_\theta J(\theta) = \mathbb{E}\big[\nabla_\theta \log \pi_\theta(a \mid s)\,(Q^{\pi}(s,a) - b(s))\big]. \tag{3}$$

Choosing $b(s) = V^{\pi}(s)$ yields the advantage function $A^{\pi}(s,a)$, motivating actor-critic architectures where a value estimator supplies the baseline.

2.3 Update Rule

Monte Carlo policy gradient performs stochastic ascent:

$$\theta_{k+1} = \theta_k + \alpha \sum_{t=0}^{T-1} \nabla_\theta \log \pi_{\theta_k}(a_t \mid s_t)\, \hat{A}_t, \tag{4}$$

where $\hat{A}_t$ is an estimator of the advantage (returns, generalized advantage estimation, etc.). Learning rates α, gradient clipping, and entropy bonuses stabilize optimization.

3 Applications and Tips

- **Continuous control**: robotics, locomotion, and control tasks with continuous actions (e.g., PPO, TRPO variants).
- **Constrained optimization**: incorporate safety constraints via penalty or Lagrangian formulations while still using gradients.
- **Imitation and fine-tuning**: initialize from expert demonstrations and refine via policy gradients.
- **Best practices**: normalize advantages, use adaptive optimizers (Adam), maintain entropy regularization, and monitor gradient norms.

4 Python Practice

The script `gen_policy_gradient_figures.py` trains a softmax policy on a stochastic bandit-style environment using REINFORCE with a learned baseline. It logs episode returns and gradient norms to visualize learning stability.

Listing 1: Excerpt from $\text{gen}_{p}olicy_{g}radient_{f}igures.py$

```
grad = features[action] - (policy @ features)
baseline = baseline + baseline_lr * (G - baseline)
theta += alpha * grad * (G - baseline)
```

5 Result

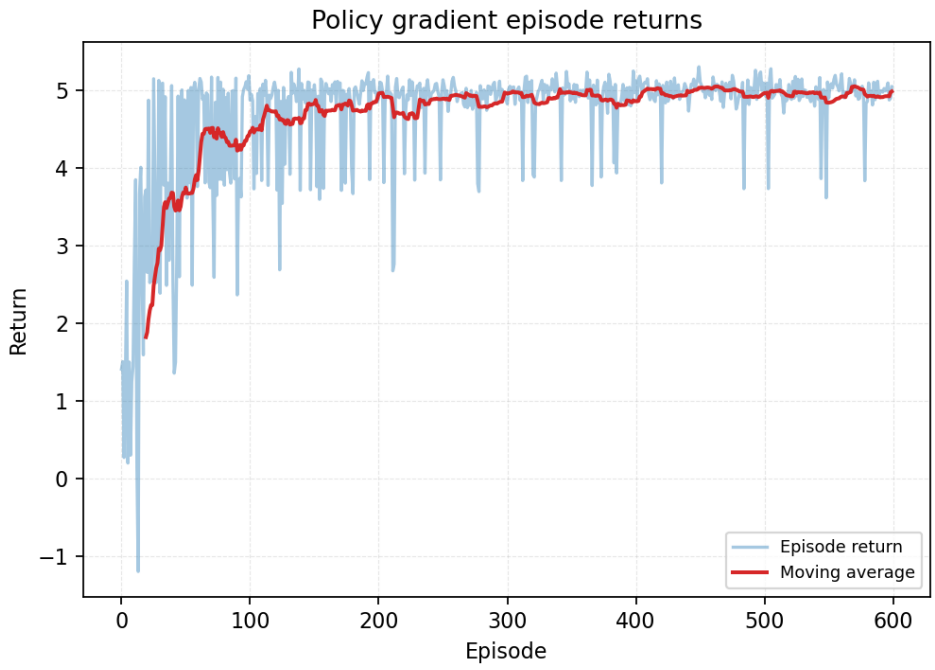

Figure 1: Policy gradient episode returns with moving average smoothing

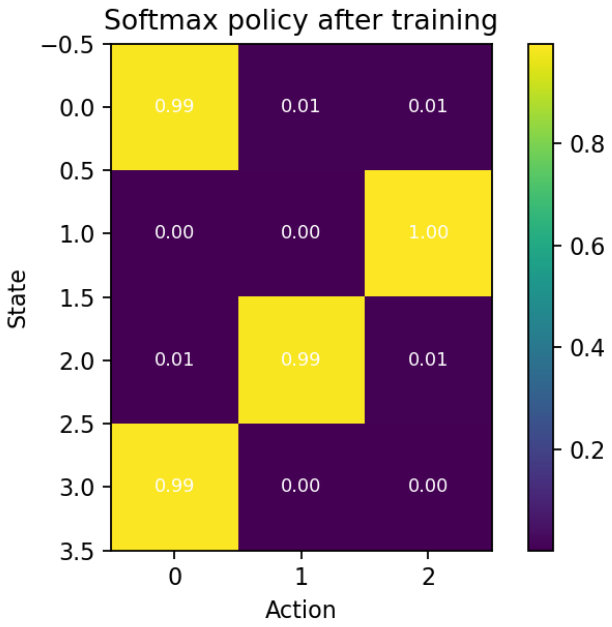

Figure 2: State-action probability heatmap learned by the softmax policy

6 Summary

Policy gradients provide a principled way to optimize stochastic policies through gradient ascent on expected returns. Variance reduction, entropy bonuses, and adaptive optimizers are crucial for stable training. The bandit example highlights how returns improve and how the policy distribution sharpens toward high-reward actions.