Chapter 11: Policy Gradients

Harsh Murdeshwar (210641)
Dhruv (210338)
Rishi Poonia (210851)
Om Singh (210686)
Ayush Himmatsinghka (210245)

Indian Institute of Technology, Kanpur

Video Presentation

Google Docs link containing the video presentation by our group:
`GoogleDocs Link`

Policy-Based vs Value-Based Methods

Policy-Based Methods (e.g., REINFORCE, PPO):

- Learn the policy $\pi_\theta(a \mid s)$ directly.
- Naturally handle stochastic policies.
- Better suited for continuous control problems.
- Require more samples to converge due to high gradient variance.

Value-Based Methods (e.g., DQN, Q-Learning):

- Learn value functions $Q(s, a)$ or $V(s)$, derive policy from them.
- Typically use greedy or ϵ-greedy action selection.
- Excel in environments with discrete, manageable action spaces.
- Lower variance, but can suffer from instability (e.g., due to max operator).

Policy Gradient Objective

- Consider a stochastic policy $\pi_\theta(a \mid s)$ parameterized by θ.
- The goal is to maximize the expected return:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$$

where $\tau = (s_0, a_0, r_0, \dots)$ is a trajectory and $R(\tau)$ is the return.

Policy Gradient Theorem

- Using the log-derivative trick:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(\tau) R(\tau) \right]$$

- Decomposing over timesteps:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) R(\tau) \right]$$

- By causality, only future rewards are affected by a_t:

$$\Rightarrow \nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \gamma^t G_t \right]$$

where $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ is the return from timestep t onward.

Note

- The theoretical gradient of the expected return is given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \gamma^t G_t \right]$$

- However, in practice, sometimes the γ^t term is dropped and the following expression is used as the gradient of $J(\theta)$

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t \right]$$

REINFORCE Algorithm

- Monte Carlo estimation of the policy gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) G_t^i$$

  where $G_t^i = \sum_{k=t}^{T_i} \gamma^{k-t} r_k^i$

- Update rule:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Variants of REINFORCE (Part 1)

- **REINFORCE:**

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t$$

  where $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$

- **Baseline (Variance Reduction):**

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t)(G_t - b(s_t))$$

- **Q-function Replacement:**

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^\pi(s_t, a_t)$$

- Justification: $\mathbb{E}[G_t \mid s_t, a_t] = Q^\pi(s_t, a_t)$

Variants of REINFORCE (Part 2)

- **Advantage Form:**

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_t \mid s_t)(Q^\pi(s_t, a_t) - V(s_t))$$

- **Generalized Advantage Estimation (GAE):**

$$A_t^{\text{GAE}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

- All variants preserve the expected gradient but may differ in variance and bias.

Variance Reduction with Baseline

- Subtracting a baseline $b(s_t)$ from the return does not affect the expected policy gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)(G_t - b(s_t))\right]$$

- **Proof that the baseline does not bias the gradient:**

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) b(s_t)\right] = \mathbb{E}_{s_t}\left[b(s_t)\, \mathbb{E}_{a_t \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]\right]$$

- Using the identity:
  $\mathbb{E}_{a \sim \pi}[\nabla_\theta \log \pi_\theta(a \mid s)] = \nabla_\theta \sum_a \pi_\theta(a \mid s) = \nabla_\theta 1 = 0$

$$\Rightarrow \mathbb{E}_{a_t \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] = 0$$

- Hence,

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) b(s_t)\right] = 0$$

Common Baselines in Policy Gradient

- **Constant Baseline:**
  - A fixed value (e.g., mean of the discounted rewards over episodes).
  - Simple but can help reduce variance.
- **Moving Average Baseline:**
  - The running average of recent G_t values.
  - Adapts over time, tracks training progress.
- **State Value Function $V(s_t)$:**
  - Learned critic network estimates expected return from state s_t.
  - Common in Actor-Critic and Advantage-based methods.

Entropy Regularization for Exploration

- Even with the policy represented as the probability distribution, there is a high chance that the agent will converge to some locally optimal policy and stop exploring the environment.
- To encourage exploration, an entropy bonus is added to the objective:

$$J'(\theta) = J(\theta) + \beta\, \mathbb{E}_{s \sim d^{\pi}} \left[\mathcal{H}(\pi_\theta(\cdot \mid s)) \right]$$

  where:
  - $\mathcal{H}(\pi(\cdot \mid s)) = -\sum_a \pi(a \mid s) \log \pi(a \mid s)$
  - $\beta > 0$ is a hyperparameter controlling exploration vs. exploitation.
- **Updated policy gradient:**

$$\nabla_\theta J'(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \left(\nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t + \beta \nabla_\theta \mathcal{H}(\pi_\theta(\cdot \mid s_t)) \right) \right]$$

Policy Gradient: Advantages and Disadvantages

Advantages:

- Can learn stochastic policies.
- Works well in high-dimensional or continuous action spaces.
- Avoids the max operator, reducing instability.
- Easier to integrate constraints or auxiliary objectives (e.g., entropy).

Disadvantages:

- High variance in gradient estimates.
- Often requires large sample sizes to learn effectively.
- Typically slower to converge compared to value-based methods.
- Sensitive to hyperparameters (learning rate, entropy weight, etc.).

The CartPole Environment

- **Goal:** Balance a pole on a moving cart by applying left or right forces.
- **Episode ends when:**
  - The pole falls beyond a certain angle ($> \pm 12^\circ$).
  - The cart moves out of bounds ($> \pm 2.4$ units).
  - Maximum timestep (typically 500) is reached.
- **Observation space (state):**
  - Cart position x
  - Cart velocity $\dot{x}$
  - Pole angle θ
  - Pole angular velocity $\dot{\theta}$
- **Action space:**
  - Discrete: $\{0 = \text{push left}, 1 = \text{push right}\}$
- **Reward:** $+1$ for each timestep the pole is balanced.

Baseline vs Training Steps Plot Comparison

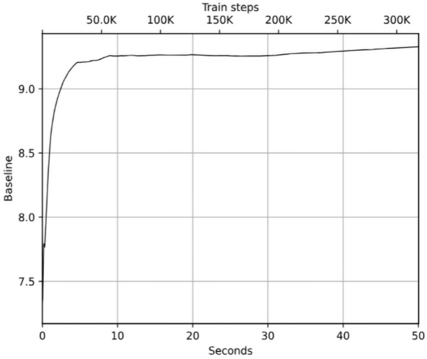

Fig 1.1: Baseline from

`04_cartpole_pg.py`

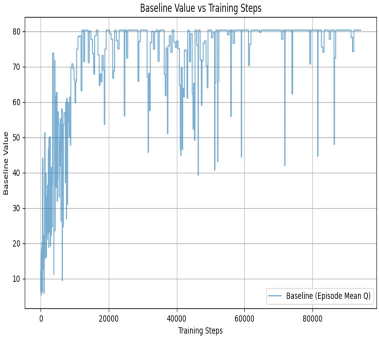

Fig 1.2: Baseline from

`03_cartpole_reinforce_baseline.py`

Policy Entropy vs Training Steps Plot Comparison

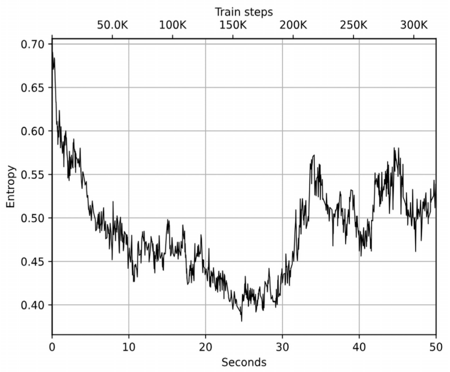

Fig 2.1: Entropy from `04_cartpole_pg.py`

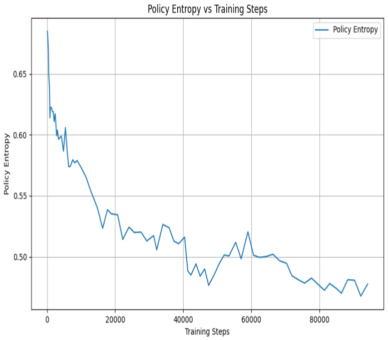

Fig 2.2: Entropy from `03_cartpole_reinforce_baseline.py`

Reward vs Training Steps Plot Comparison

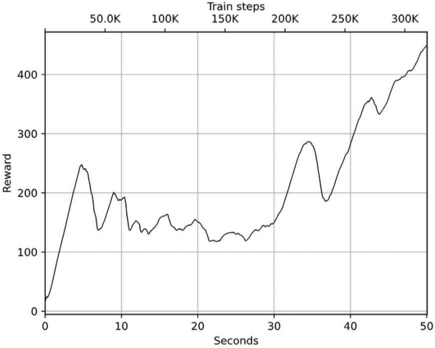

Fig 3.1: Rewards from

`04_cartpole_pg.py`

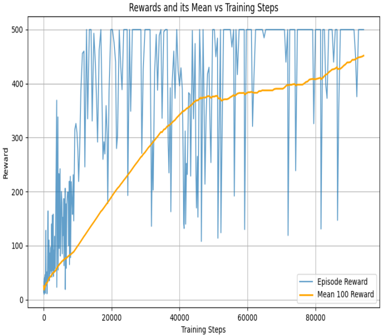

Fig 3.2: Rewards from

`03_cartpole_reinforce_baseline.py`