Week 3 Report

Mu Junrong

Abstract

This report focuses on the mathematics involved, the background about the environment (CartPole), as well as the results obtained.

1 Mathmetics involved

From Week 2's Report section 4, in Reinforcement Learning, the agent learns a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes expected return (Sutton & Barto (2018)):

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The main objective of Reinforcement Learning (RL) is to find an optimal policy π that maximizes the reward (i.e. maximizes the state-value function and action-value function) (Sutton et al. (1999)):

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | \pi \right]$$

1.1 Objective function

From section 5.1 and 5.2, the objective function of Reinforcement Learning is defined as

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

where τ represent a trajectory (i.e., the sequence of states, actions, rewards):

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots)$$

By differentiating the objective function $J(\theta)$:

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \int p_\theta(\tau) R(\tau) \, d\tau \\ &= \int \nabla_\theta p_\theta(\tau) R(\tau) \, d\tau \\ &= \mathbb{E}_{\tau \sim \pi_\theta} \left[\nabla_\theta \log p_\theta(\tau) R(\tau) \right] \end{aligned}$$

1.2 Basic policy gradient

From section 5.3, the trajectory probability under the policy is given by

$$p_\theta(\tau) = p(s_0) \prod_{t=0}^{T} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t)$$

$$\log p_\theta(\tau) = \sum_{t=0}^{T} \log \pi_\theta(a_t | s_t) + \mathrm{C}$$

By substituting into the objective function, the basic policy gradient formula is derived

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

Therefore, in the algorithm, the policy parameters are updated using gradient ascent

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

where α is the learning rate.

2 Background of the environment

CartPole is a physics-based control problem. It simulates a cart on a track which can move left and right. There is a pole (e.g. inverted pendulum) that starts upright on the cart, but can fall to either side. The goal of the agent is to keep the pole balanced upright by moving the cart left or right.

2.1 Key definitions in CartPole

The state space $\mathcal{S}$ is a finite set of 4-dimensional vectors:

$$\mathcal{S} \subseteq \mathbb{R}^4 \quad \text{where} \quad \mathbf{s}_t = \begin{pmatrix} x_t & \dot{x}_t & \theta_t & \dot{\theta}_t \end{pmatrix}^\top$$

with components:

- x_t: Cart position
- $\dot{x}_t$: Cart velocity
- θ_t: Pole angle (in radians)
- $\dot{\theta}_t$: Pole angular velocity

The action space $\mathcal{A}$ is a discrete set:

$$\mathcal{A} = \{0, 1\} \quad \text{where} \quad \begin{cases} 0 = \text{“move left”} \\ 1 = \text{“move right”} \end{cases}$$

The reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ provides:

$$r(\mathbf{s}_t, a_t) = +1 \quad \text{for every timestep the pole remains balanced}$$

A trajectory $\tau = (\mathbf{s}_0, a_0, \mathbf{s}_1, a_1, \dots)$ terminates when either:

- The pole falls: $|\theta_t| > \theta_{\max}$ ($\approx 12^\circ$ or 0.209 rad)
- The cart moves out of bounds: $|x_t| > x_{\max}$ (typically 2.4 m)

2.2 Aim of the agent

The agent must learn a policy (a way to choose left or right) that keeps the pole upright as long as possible. Since it gets a reward of +1 per timestep, the agent is incentivized to stay balanced longer.

3 Result of the experiment

This experiment evaluates the performance of Vanilla policy gradient (VPG) methods in the CartPole-v0 environment, focusing on two key design choices: Reward-to-go (RTG) vs. non-RTG return estimation and small batch (1,000 steps) vs. large batch (4,000 steps) training. The goal is to analyze how RTG and batch size affect sample efficiency (i.e., learning speed), final performance (i.e., maximum achievable return) and training stability (i.e., variance in returns).

In conclusion, large batch data with reward-to-go (RTG) yield the best performance (i.e., highest return, low variance and less fluctuation). In general, large batch data and reward-to-go (RTG) yield better results than smaller batch data and without reward-to-go.

3.1 Variance in return

Based on Figure 1 and Figure 2, the learning curves of data with reward-to-go are smoother, with fewer oscillations and swings, indicating lower variance. Thus, the reward-to-go method reduces the variance, ensuring stability.

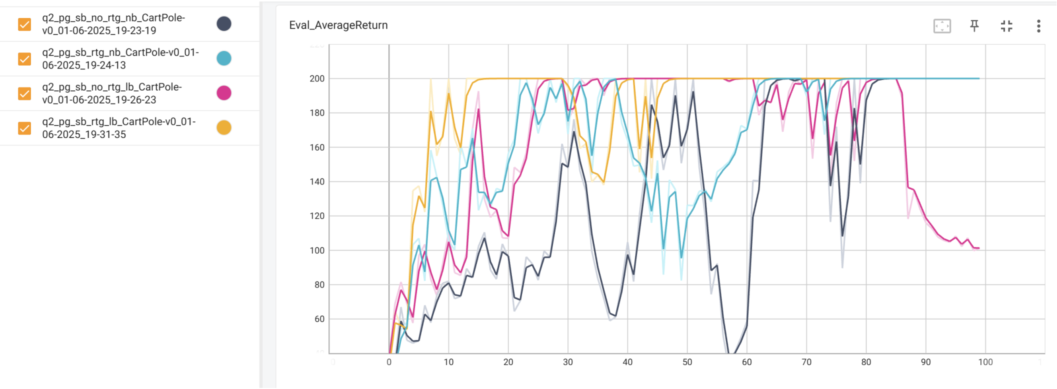

Figure 1: Learning curve of average return over training steps.

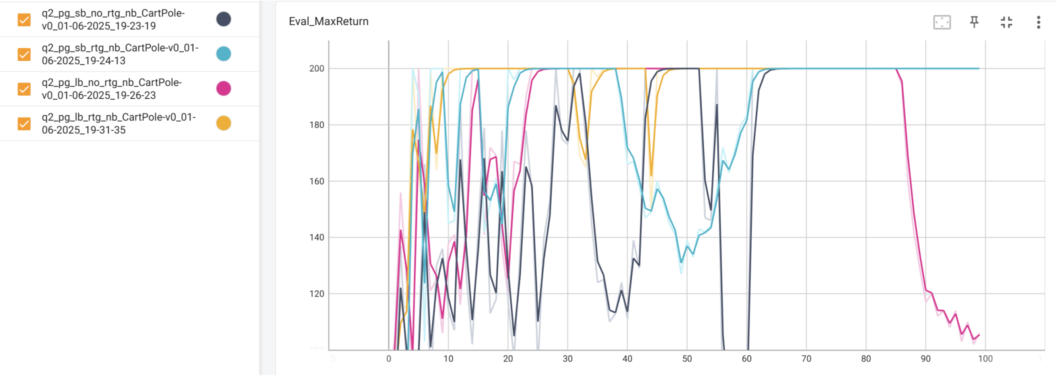

Figure 2: Learning curve of maximum return over training steps.

3.2 Learning speed and final performance

Based on Figure 1 and Figure 2, all four experiments reach 200 (the maximum achievable for CartPole-v0) as the training steps increase. Based on Figure 3, the actor loss is decreasing steadily since the policy is learning to increase the expected return. Especially for large batch data without reward-to-go, the curve experiences a sharp decrease around the time step = 80, possibly because of a quick learning where the reward is. However, large batch data with reward-to-go converges to 200 faster, and the loss decreases at a faster rate.

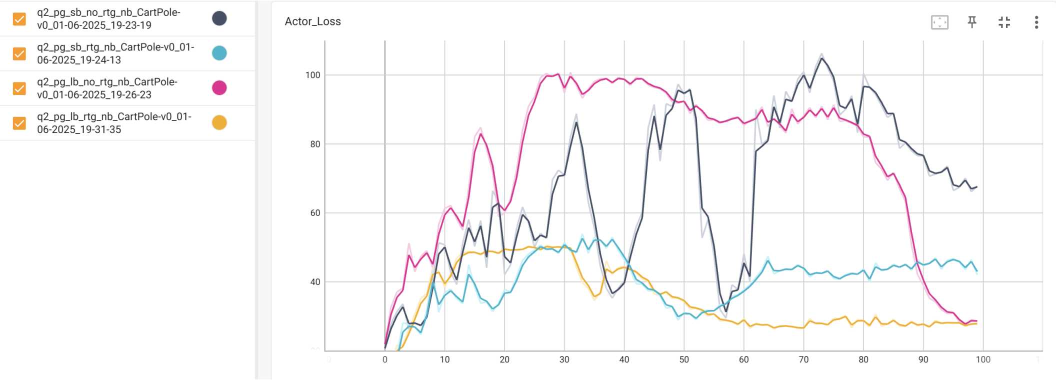
☑ q2_pg_sb_no_rtg_nb_CartPole-v0_01-06-2025_19-23-19 ●
☑ q2_pg_sb_rtg_nb_CartPole-v0_01-06-2025_19-24-13 ●
☑ q2_pg_lb_no_rtg_nb_CartPole-v0_01-06-2025_19-26-23 ●
☑ q2_pg_lb_rtg_nb_CartPole-v0_01-06-2025_19-31-35 ●
Actor_Loss
100
80
60
40
0
10
20
30
40
50
60
70
80
90
100

Figure 3: Learning curve of actor loss over training steps.

References

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018. URL `http://incompleteideas.net/book/RLbook2020.pdf`.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12 (NeurIPS 1999)*, pp. 1057–1063. MIT Press, 1999.