



國立陽明交通大學

NATIONAL YANG MING CHIAO TUNG UNIVERSITY

[CSIC30046] Reinforcement Learning

Homework 1

Po-Chuan, Chen

Student ID: 311511052

present90308.ee11@nycu.edu.tw

NATIONAL YANG MING CHIAO TUNG UNIVERSITY

March 30, 2023

Contents

1 Problem 1 (Baseline for Variance Reduction)	2
2 Problem 2 (Policy Gradient)	8
3 Problem 3 (Monte Carlo Policy Evaluation)	9
4 Problem 4 (Policy Gradient Algorithms With Function Approximation)	11

1 Problem 1 (Baseline for Variance Reduction)

(a)

Problem 1 (Baseline for Variance Reduction)

Only 1 non-terminal starting state s and 3 actions a, b, c in the MDP of interest.

Reward : $r(s, a) = 100$, $r(s, b) = 98$, $r(s, c) = 95$

$$\text{Softmax policy} : \pi(\cdot | s) = \frac{\exp(\theta_a)}{(\exp(\theta_a) + \exp(\theta_b) + \exp(\theta_c))}$$

$$\theta_a = 0, \theta_b = \ln 5, \theta_c = \ln 4$$

$s_0 = s$, a_0 is either a, b, c , and s_1 is terminal state

(a) What are the mean vector of \hat{V} and the covariance matrix of $\hat{V} = (\mathbb{E}[\hat{V}] - \mathbb{E}[\hat{V}]) (\hat{V} - \mathbb{E}[\hat{V}])^\top$?

$$\bar{\nabla}_T := R(T) \sum_{t=0}^{\infty} \nabla_t \log \pi_\theta(a_t | s_t)$$

$$\pi(a_0 = a | s_0) = \frac{\exp(0)}{\exp(0) + \exp(\ln 4) + \exp(\ln 5)} = \frac{1}{1+4+5} = \frac{1}{10}$$

$$\pi(a_0 = b | s_0) = \frac{5}{10} \quad \pi(a_0 = c | s_0) = \frac{4}{10}$$

$$\frac{\partial \log \pi_\theta(s, a)}{\partial \theta(s, a)} = 1 - \pi_\theta(s, a) = 1 - 0.1 = 0.9$$

$$\frac{\partial \log \pi_\theta(s, a)}{\partial \theta(s, b)} = -\pi_\theta(s, b) = -0.5 \quad \frac{\partial \log \pi_\theta(s, a)}{\partial \theta(s, c)} = -0.4$$

$$\Rightarrow \nabla_\theta \log \pi_\theta(a | s) = \begin{bmatrix} 0.9 \\ -0.5 \\ -0.4 \end{bmatrix}, \quad \nabla_\theta \log \pi_\theta(b | s) = \begin{bmatrix} -0.1 \\ 0.5 \\ -0.4 \end{bmatrix}$$

$$\nabla_\theta \log \pi_\theta(c | s) = \begin{bmatrix} -0.1 \\ -0.5 \\ 0.6 \end{bmatrix}$$

Mean vector

$$\begin{aligned}
 \mathbb{E}[\hat{\nabla}V] &= \mathbb{E}_{\tau \sim p_{\mathcal{M}}^{\pi_0}} \left[\sum_{t=0}^{T-1} r_t^t Q^{\pi_0}(s_t, a_t) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right] \\
 &= \pi_{\theta}(a|s) \cdot [r_a \cdot \nabla_{\theta} \log \pi_{\theta}(a|s)] \\
 &\quad + \pi_{\theta}(b|s) \cdot [r_b \cdot \nabla_{\theta} \log \pi_{\theta}(b|s)] \\
 &\quad + \pi_{\theta}(c|s) \cdot [r_c \cdot \nabla_{\theta} \log \pi_{\theta}(c|s)] \\
 &= 0.1 \times 100 \times \begin{bmatrix} 0.9 \\ -0.5 \\ -0.4 \end{bmatrix} + 0.5 \times 98 \times \begin{bmatrix} -0.1 \\ 0.5 \\ -0.4 \end{bmatrix} \\
 &\quad + 0.4 \times 95 \times \begin{bmatrix} -0.1 \\ -0.5 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 9 - 4.9 - 3.8 \\ -5 + 24.5 - 19 \\ -4 - 19.6 + 22.8 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.5 \\ -0.8 \end{bmatrix}
 \end{aligned}$$

Covariance matrix

$$\begin{aligned}
 \therefore \mathbb{E}[\hat{\nabla}V] &= \begin{bmatrix} 0.3 \\ 0.5 \\ -0.8 \end{bmatrix} \quad \hat{\nabla}V_a = \begin{bmatrix} 40 \\ -50 \\ -40 \end{bmatrix} \quad \hat{\nabla}V_b = \begin{bmatrix} -9.8 \\ 49 \\ -39.2 \end{bmatrix} \quad \hat{\nabla}V_c = \begin{bmatrix} -9.5 \\ -47.5 \\ 57 \end{bmatrix} \\
 \hat{\nabla}V_a - \mathbb{E}[\hat{\nabla}V] &= \begin{bmatrix} 89.7 \\ -50.5 \\ -39.2 \end{bmatrix} \quad \hat{\nabla}V_b - \mathbb{E}[\hat{\nabla}V] = \begin{bmatrix} -10.1 \\ 48.5 \\ -38.4 \end{bmatrix} \\
 \hat{\nabla}V_c - \mathbb{E}[\hat{\nabla}V] &= \begin{bmatrix} -9.8 \\ -48 \\ 57.8 \end{bmatrix} \\
 \mathbb{E}[(\hat{\nabla}V - \mathbb{E}[\hat{\nabla}V])(\hat{\nabla}V - \mathbb{E}[\hat{\nabla}V])^T] & \\
 \Rightarrow 0.1 \times \begin{bmatrix} 89.7 \\ -50.5 \\ -39.2 \end{bmatrix} \begin{bmatrix} 89.7 & -50.5 & -39.2 \end{bmatrix}^T &\times 0.5 \begin{bmatrix} -10.1 \\ 48.5 \\ -38.4 \end{bmatrix} \begin{bmatrix} -10.1 & 48.5 & -38.4 \end{bmatrix}^T \\
 + 0.4 \times \begin{bmatrix} -9.8 \\ -48 \\ 57.8 \end{bmatrix} \begin{bmatrix} -9.8 & -48 & 57.8 \end{bmatrix}^T &
 \end{aligned}$$

$$\begin{aligned}
 & \Rightarrow \begin{bmatrix} 804.609 & -452.985 & -351.624 \\ -452.985 & 255.025 & 197.96 \\ -351.624 & 197.96 & 153.664 \end{bmatrix} + \begin{bmatrix} 51.005 & -244.925 & 193.92 \\ -244.925 & 1176.125 & -931.2 \\ 193.92 & -931.2 & 737.28 \end{bmatrix} \\
 & + \begin{bmatrix} 38.416 & 188.16 & -226.576 \\ 188.16 & 921.6 & -1109.76 \\ -226.576 & -1109.76 & 1336.336 \end{bmatrix} = \begin{bmatrix} 894.03 & -509.75 & -384.28 \\ -509.75 & 2352.75 & -1843 \\ -384.28 & -1843 & 2227.28 \end{bmatrix} \quad \# \\
 \end{aligned}$$

(b)

Mean vector

(b) Value function $V^{\pi_0}(s)$ as the baseline and denote by $\tilde{\nabla}V$ the corresponding estimated policy gradient.

Mean vector and the covariance matrix of $\tilde{\nabla}V$?

$$\begin{aligned}
 \text{Mean vector } V^{\pi_0}(s) &= \sum_{a \in A} \pi(a|s) Q^{\pi_0}(s, a) = 0.1 \times 100 + 0.5 \times 98 + 0.4 \times 95 \\
 &= 97
 \end{aligned}$$

$$Q^{\pi_0}(s, a) - \beta(s) = 3$$

$$Q^{\pi_0}(s, b) - \beta(s) = 1$$

$$Q^{\pi_0}(s, c) - \beta(s) = -2$$

$$\begin{aligned}
 \mathbb{E}[\tilde{\nabla}V] &= 0.1 \times 3 \times \begin{bmatrix} 0.9 \\ -0.5 \\ -0.4 \end{bmatrix} + 0.5 \times 1 \times \begin{bmatrix} -0.1 \\ 0.5 \\ -0.4 \end{bmatrix} \\
 &\quad + 0.4 \times (-2) \times \begin{bmatrix} -0.1 \\ -0.5 \\ 0.6 \end{bmatrix} \\
 &= \begin{bmatrix} 0.27 - 0.05 + 0.08 \\ -0.15 + 0.25 + 0.4 \\ -0.12 - 0.2 - 0.48 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.5 \\ -0.8 \end{bmatrix}
 \end{aligned}$$

Covariance matrix

Since we got the mean vector

The covariance matrix

$$\mathbb{E}[(\tilde{V} - \mathbb{E}[\tilde{V}])(\tilde{V} - \mathbb{E}[\tilde{V}])^T]$$

$$\begin{aligned}\tilde{V}_A &= 3 \times \begin{bmatrix} 0.9 \\ -0.5 \\ -0.4 \end{bmatrix} & \tilde{V}_B &= 1 \times \begin{bmatrix} -0.1 \\ 0.5 \\ -0.4 \end{bmatrix} & \tilde{V}_C &= (-2) \times \begin{bmatrix} -0.1 \\ -0.5 \\ 0.6 \end{bmatrix} \\ &= \begin{bmatrix} 2.7 \\ -1.5 \\ -1.2 \end{bmatrix} & &= \begin{bmatrix} -0.1 \\ 0.5 \\ -0.4 \end{bmatrix} & &= \begin{bmatrix} 0.2 \\ 1 \\ -1.2 \end{bmatrix}\end{aligned}$$

$$\tilde{V}_A - \mathbb{E}[\tilde{V}] = \begin{bmatrix} 2.7 & -0.3 \\ -1.5 & 0.5 \\ -1.2 & 0.8 \end{bmatrix} = \begin{bmatrix} -2.4 \\ -2 \\ 2.4 \end{bmatrix} \quad \tilde{V}_B - \mathbb{E}[\tilde{V}]$$

$$\tilde{V}_B - \mathbb{E}[\tilde{V}] = \begin{bmatrix} -0.1 & -0.3 \\ 0.5 & -0.5 \\ -0.4 & 0.8 \end{bmatrix} = \begin{bmatrix} -0.4 \\ 0 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.2 & -0.3 \\ 1 & -0.5 \\ -1.2 & 0.8 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.5 \\ -0.4 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 5.76 & -4.8 & -0.96 \\ -4.8 & 4 & 0.8 \\ -0.96 & 0.8 & 0.16 \end{bmatrix} \times 0.1$$

$$+ \begin{bmatrix} 0.16 & 0 & -0.16 \\ 0 & 0 & 0 \\ -0.16 & 0 & 0.16 \end{bmatrix} \times 0.5$$

$$+ \begin{bmatrix} 0.01 & -0.05 & 0.04 \\ -0.05 & 0.25 & -0.2 \\ 0.04 & -0.12 & 0.16 \end{bmatrix} \times 0.4 = \begin{bmatrix} 0.66 & -0.15 & -0.16 \\ -0.15 & 0.15 & 0 \\ -0.16 & 0 & 0.16 \end{bmatrix}$$

(c)

(c) $B(s)$ denote a baseline function and ∇V_B be the corresponding estimated policy gradient. Find the optimal $B(s)$ that can get the minimum trace of the corresponding covariance matrix of ∇V_B .

Assume that $B = B(s)$

$$\mathbb{E} \left[\nabla V_B \right] = 0.1 \times (100 - B) \times \begin{bmatrix} 0.9 \\ -0.5 \\ -0.4 \end{bmatrix} + 0.5 \times (98 - B) \times \begin{bmatrix} -0.1 \\ 0.5 \\ -0.4 \end{bmatrix} \\ + 0.4 \times (95 - B) \times \begin{bmatrix} -0.1 \\ -0.5 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.3 \\ -0.5 \\ 0.8 \end{bmatrix}$$

The trace of $\mathbb{E} \left[(\nabla V_B - \mathbb{E} [\nabla V_B]) (\nabla V_B - \mathbb{E} [\nabla V_B])^T \right]$

$$\Rightarrow 0.1 \times [(89.7 - 0.9B)^2 + (-50.5 + 0.5B)^2 + (-39.2 + 0.4B)^2] \\ + 0.5 \times [(-10.1 + 0.1B)^2 + (48.5 - 0.5B)^2 + (-38.4 + 0.4B)^2] \\ + 0.4 \times [(-9.8 + 0.1B)^2 + (-48 + 0.5B)^2 + (57.8 - 0.6B)^2]$$

$$\begin{aligned}
 \frac{dT}{dB} &= 0.1 \times 2 \times \left[(-0.9) \times (89.7 - 0.9B) + (0.5) \times (-50.5 + 0.5B) \right. \\
 &\quad \left. + (0.4) (-39.2 + 0.4B) \right] \\
 &\quad + 0.5 \times 2 \times \left[(0.1) \times (-10.1 + 0.1B) + (-0.5) (48.5 - 0.5B) \right. \\
 &\quad \left. + (0.4) (-38.4 + 0.4B) \right] \\
 &\quad + 0.4 \times 2 \times \left[(0.1) \times (-4.8 + 0.1B) + (0.5) (-48 + 0.5B) \right. \\
 &\quad \left. + (-0.6) (57.8 - 0.6B) \right] \\
 &= 2 \times \left[0.1 \times (-121.66 + 1.22B) + 0.5 \times (-40.62 + 0.42B) \right. \\
 &\quad \left. + 0.4 \times (-59.66 + 0.16B) \right] \\
 &= 2 \times [-56.34 + 0.58B]
 \end{aligned}$$

If the $\frac{dT}{dB} = 0$, we can find the best B

$$B = \frac{56.34}{0.58} = 97.137931034 \approx 97.14$$

2 Problem 2 (Policy Gradient)

Problem 2 (Policy Gradient)

For any function $f: S \times A \rightarrow \mathbb{R}$

$$\mathbb{E}_{\tau \sim P_M^{\pi_\theta}} \left[\sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \right] = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_M^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta(\cdot | s)} [f(s, a)]$$

First, we can start from $\frac{1}{1-\gamma} \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(a|s) [f(s, a)]$

since the def. $d^{\pi_\theta}(s)$, it equals to

$$\frac{1}{1-\gamma} \sum_s \left(\sum_{s_0} M(s_0) \cdot (1-\gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t=s | s_0, \pi_\theta) \right) \cdot \sum_a \pi_\theta(a|s) P(s, a)$$

$$= \sum_{s_0} M(s_0) \sum_s \sum_a \pi_\theta(a|s) \sum_{t=0}^{\infty} \gamma^t P(s_t=s | s_0, \pi_\theta) \cdot f(s, a)$$

$$= \sum_{s_0} M(s_0) \left[\sum_s \sum_a \sum_{t=0}^{\infty} \gamma^t P(s_t=s, a_t=a | s_0, \pi_\theta) \right] \cdot f(s, a)$$

$$= \sum_{s_0} M(s_0) \sum_{\tau} \sum_{\substack{s=s \\ s_t=s, a_t=a}} \gamma^t f(s_t, a_t)$$

$$= \sum_{\tau} P(\tau | s_0, \pi_\theta) \cdot \sum_{t=0}^{\infty} \gamma^t \sum_{\substack{s=s \\ s_t=s, a_t=a}} f(s_t, a_t)$$

$$= \sum_{\tau} P(\tau) \cdot \sum_{t=0}^{\infty} \gamma^t \cdot f(s_t, a_t) = \mathbb{E}_{\tau \sim P_M^{\pi_\theta}} \left[\sum_{t=0}^{\infty} \gamma^t \cdot f(s_t, a_t) \right]$$

all τ

3 Problem 3 (Monte Carlo Policy Evaluation)

Property 1

Problem 3 (Monte Carlo Policy Evaluation)

R_S, P_S (S) $\xrightarrow{P_T}$ T
A simple 2-state MDP

Property 1: Show that the true value function at state S satisfies that $V(S) = \frac{P_S}{P_T} R_S + R_T$

First, we know that

Possible scenarios	$P_S = \text{Prob. of visiting } S \text{ again before reaching } T$
	$P_T = \text{Prob. of visiting } T \text{ before visiting } S \text{ again}$

$R_S = \text{Expected reward of } S \rightarrow S \text{ transition}$

$R_T = \text{Expected reward of } S \rightarrow T \text{ transition}$

$S \rightarrow S$

$\{S\}_1 \rightarrow \dots \rightarrow \{S\}_T$

$(S) \rightarrow O \dots O \rightarrow (S) \dots (S) \rightarrow O \dots O \rightarrow (T)$

From Bellman's equation

$$V(S) = P_S (R_S + V(S)) + P_T (R_T + V_T)$$

$$\Rightarrow V(S) = P_S R_S + P_S V(S) + P_T R_T$$

since $P_S + P_T = 1$

$$\Rightarrow (1 - P_S) V(S) = P_S R_S + P_T R_T = P_T V(S) = P_S R_S + P_T R_T$$

divide P_T

$$\Rightarrow V(S) = \frac{P_S}{P_T} R_S + R_T$$

Property 2

Property 2: Suppose we construct an every-visit MC estimate based on only 1 trajectory τ .

$$\text{Then, please show that } \mathbb{E}_{\tau} [\hat{V}_{MC}(s; \tau)] = \frac{p_s}{2p_{\tau}} R_s + R_{\tau}$$

With 1 trajectory, the bias of the every-visit MC algorithm

$$\mathbb{E} [\hat{V}_{MC}(s; \tau)] = \sum_{k=0}^{\infty} p(k) \mathbb{E}_{\tau} (\tau | k)$$

$$= \sum_{k=0}^{\infty} p_{\tau} p_s^k \left(\frac{R_s + 2R_s + \dots + kR_s + (k+1)R_{\tau}}{k+1} \right)$$

$$= \sum_{k=0}^{\infty} p_{\tau} p_s^k \left(\frac{k}{\sum} R_s + R_{\tau} \right)$$

$$= \frac{p_s}{2p_{\tau}} R_s + R_{\tau} *$$

$$\Rightarrow \text{Bias for a trajectory equal to } V(s) - \mathbb{E} [\hat{V}(s)] = \frac{p_s}{2p_{\tau}} R_s$$

4 Problem 4 (Policy Gradient Algorithms With Function Approximation)

(a)

Hyperparameters

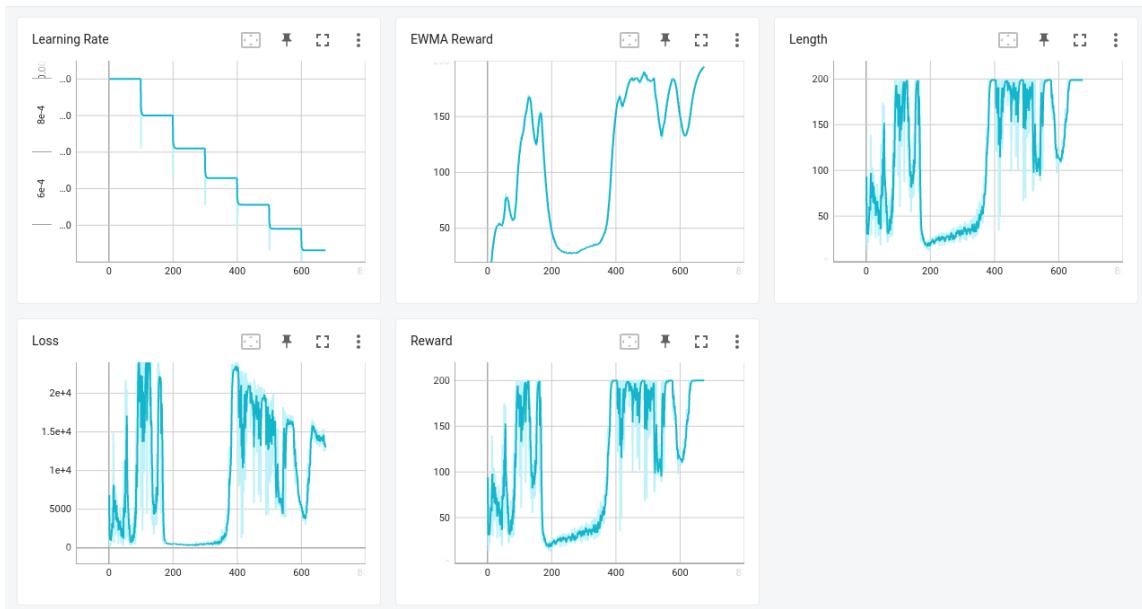
- Learning rate: 0.001
- Gamma: 0.999
- Scheduler step size: 100
- Scheduler gamma: 0.9
- Optimizer: Adam

```
1 self.shared_first = nn.Linear(self.observation_dim, self.hidden_size)
2 # initialize the shared layer
3 nn.init.kaiming_normal_(self.shared_first.weight)
4 # random weight initialization
5 self.shared_second = nn.Linear(self.hidden_size, self.hidden_size)
6 nn.init.kaiming_normal_(self.shared_second.weight)
7 self.action_head = nn.Linear(self.hidden_size, self.action_dim)
8 # initialize the action layer
9 nn.init.kaiming_normal_(self.action_head.weight)
10 self.value_head = nn.Linear(self.hidden_size, 1)
11 # initialize the value layer, since we only need one value, we set the
12     output dimension to 1
13 nn.init.kaiming_normal_(self.value_head.weight)
```

Listing 1: weight initialize

```
1 def forward(self, state):
2     """
3         Forward pass of both policy and value networks
4         - The input is the state, and the outputs are the corresponding
5             action probability distribution and the state value
6         TODO:
7             1. Implement the forward pass for both the action and the
8                 state value
9     """
10    #### YOUR CODE HERE (3~5 lines) #####
11    input = F.relu(self.shared_first(state))
12    input = F.relu(self.shared_second(input))
13    action_prob = F.softmax(self.action_head(input), dim=-1)
14    # softmax over the last dimension
15    state_value = self.value_head(input)
16    # no activation function for the value layer
17    ##### END OF YOUR CODE #####
18
19    return action_prob, state_value
```

Listing 2: Network architecture

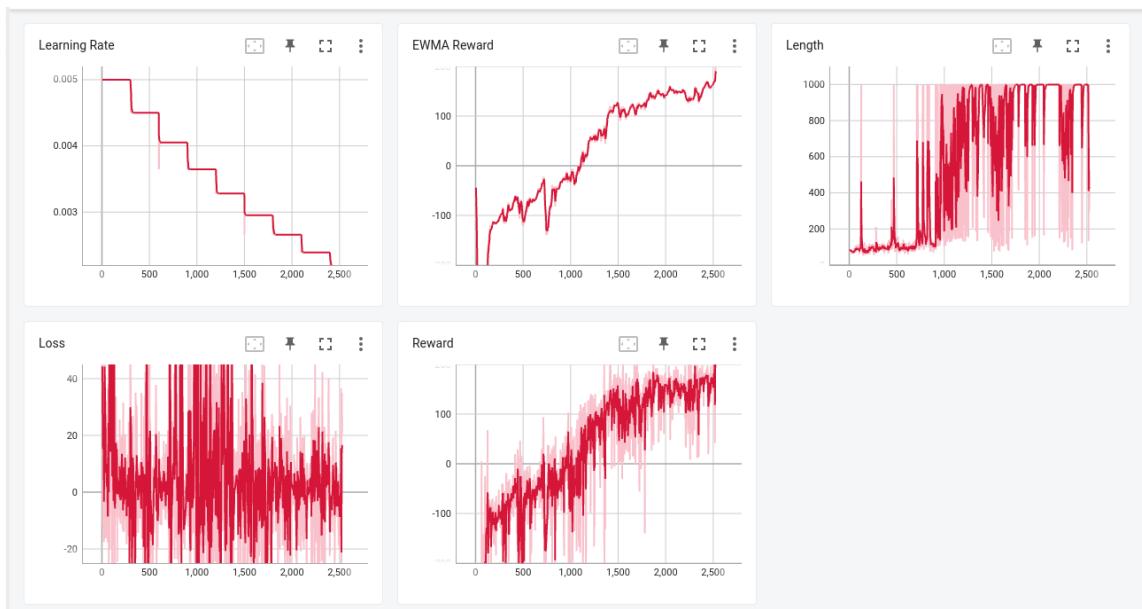


(b)

Hyperparameters

- Learning rate: 0.005
- Gamma: 0.99
- Scheduler step size: 300
- Scheduler gamma: 0.9
- Optimizer: Adam

Same network architecture as (a). But the accumulating reward will add divided by 200.



(c)

Hyperparameters

- Learning rate: 0.005
- Gamma: 0.999
- Scheduler step size: 300
- Scheduler gamma: 0.9
- Optimizer: Adam

Same network architecture as (a).

```
1 class GAE:
2     def __init__(self, gamma, lambda_, num_steps):
3         self.gamma = gamma
4         self.lambda_ = lambda_
5         self.num_steps = num_steps
6         # set num_steps = None to adapt full batch
7
8     def __call__(self, rewards, values, done):
9
10        #Implement Generalized Advantage Estimation (GAE) for your value
11        #prediction
12        #TODO (1): Pass correct corresponding inputs (rewards, values, and
13        #done) into the function arguments
14        #TODO (2): Calculate the Generalized Advantage Estimation and
15        #return the obtained value
16
17
18        ##### YOUR CODE HERE (8-15 lines) #####
19        # Initialize the lists and variables
20        advantages = []
21        advantage = 0
22        next_value = 0
23
24        for reward, value, done in zip(reversed(rewards), reversed(values),
25                                         reversed(done)):
26            td_error = reward + self.gamma * next_value * (1 - done) -
27            value
28            # calculate the temporal difference error
29            advantage = td_error + self.gamma * self.lambda_ *
30            advantage * (1 - done)
31            # calculate the advantage
32            next_value = value # update the next value
33            advantages.insert(0, advantage)
34            # insert the advantage to the front of the list
35
36            advantages = torch.tensor(advantages)
37            # convert the list to a tensor
38            advantages = (advantages - advantages.mean()) / (advantages.std()
39            () + 1e-5) # normalize the advantages
40
41
42        return advantages
```

Listing 3: Generalized Advantage Estimation (GAE)

```

1 def calculate_loss(self, gamma=0.999, advantages=None):
2     """
3         Calculate the loss (= policy loss + value loss) to perform
4         backprop later
5         TODO:
6             1. Calculate rewards-to-go required by REINFORCE with the
7                 help of self.rewards
8             2. Calculate the policy loss using the policy gradient
9             3. Calculate the value loss using either MSE loss or smooth
10            L1 loss
11    """
12
13    # Initialize the lists and variables
14
15    R = 0
16    saved_actions = self.saved_actions
17    policy_losses = []
18    value_losses = []
19
20    vals = []
21
22    for a, v in saved_actions:
23        vals.append(v[0])
24    ##### YOUR CODE HERE (8-15 lines) #####
25    for (log_prob, val), advantage in zip(saved_actions, advantages(
26        self.rewards, vals, self.done)):
27        policy_losses.append(-log_prob * advantage.detach())
28        value_losses.append(advantage.pow(2))
29
30    loss = torch.stack(policy_losses).sum() + torch.stack(value_losses)
31    .sum()
32    ##### END OF YOUR CODE #####
33
34    return loss

```

Listing 4: Loss for GAE

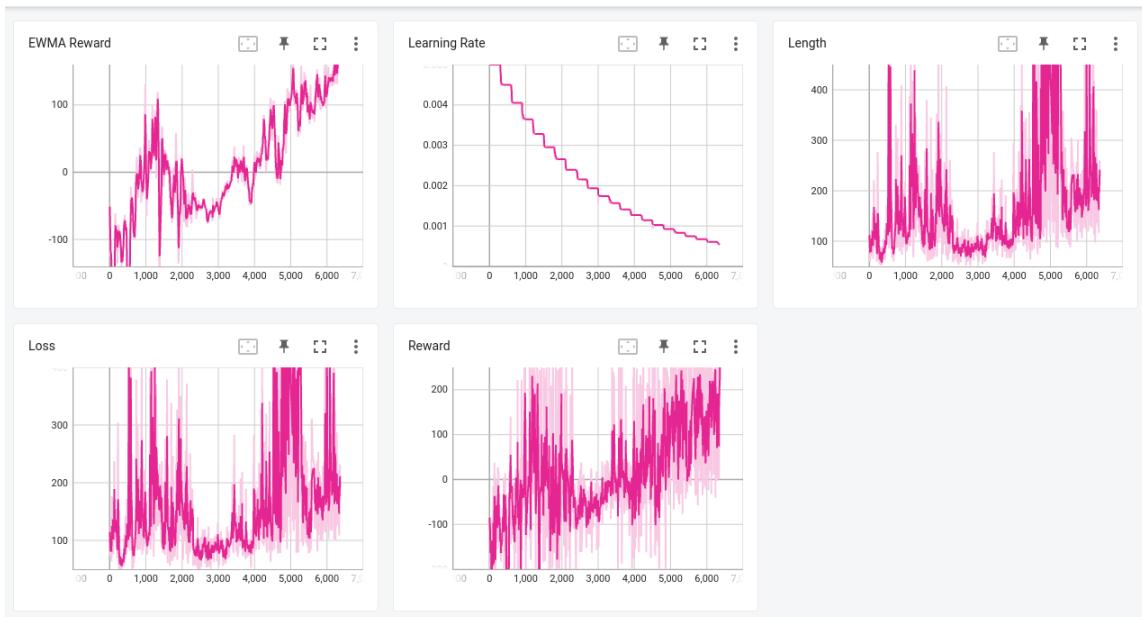
I used the algorithm about REINFORCE with GAE to implement the code

Initialize θ_0 and step size η

Sampling trajectory $\tau \sim P_\mu^{\pi_\theta}$ and make the update as

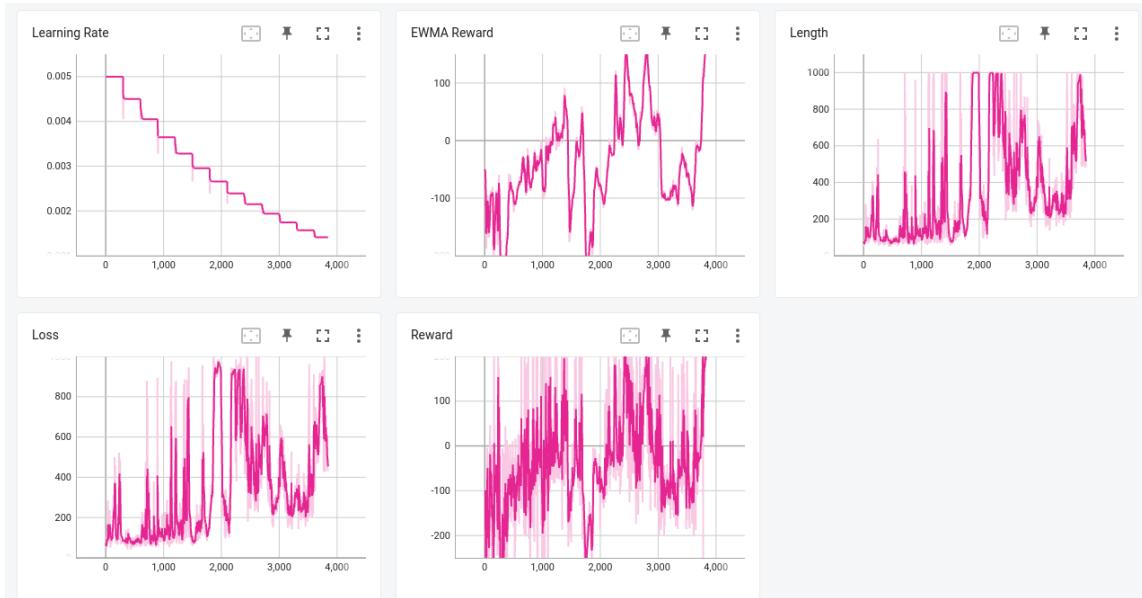
$$\theta_{k+1} = \theta_k + \eta \left(\sum_{t=0}^{\infty} \gamma^t \hat{A}_t^{GAE(\gamma, \lambda)} \nabla_\theta \log \pi_\theta(a_t | s_t) \right)$$

where $\hat{A}_t^{GAE(\gamma, \lambda)}$ is constructed from $V(s)$ with TD return



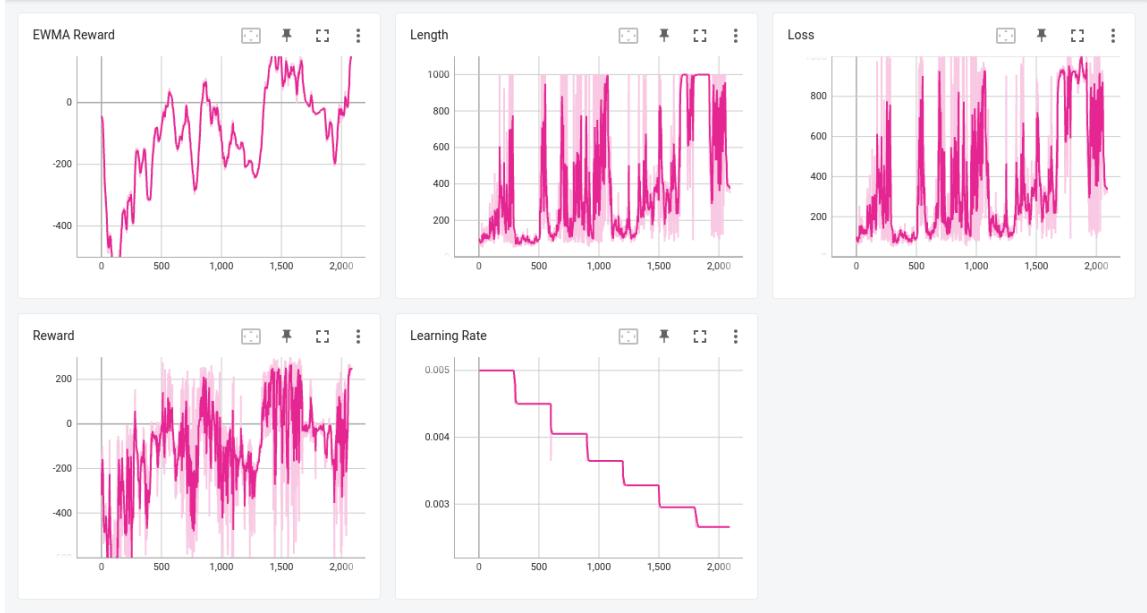
$$\lambda = 0.99$$

The model learns the task after around 6000 episodes when $\lambda = 0.99$.



$$\lambda = 0.98$$

The model learns the task after around 4000 episodes when $\lambda = 0.98$.



$$\lambda = 0.97$$

The model learns the task after around 2000 episodes when $\lambda = 0.97$.

First, we can look at the formula about GAE estimator:

$$\hat{A}^{GAE(\gamma, \lambda)} = (1 - \lambda)(\hat{A}_t^{(1)} + \lambda(\hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots)) = \sum_{\ell=0}^{\infty} (\gamma \lambda)^{\ell} \delta_{t+\ell}$$

As we can see, when we using larger λ , the convergence range will get larger.

When the $\lambda = 1$, we will use MC return, and when $\lambda = 0$ we will use TD return.