[CSIC30046] Reinforcement Learning

Homework 2

Po-Chuan, Chen

Student ID: 311511052

`present90308.ee11@nycu.edu.tw`

NATIONAL YANG MING CHIAO TUNG UNIVERSITY

May 2, 2023

Contents

1 Problem 1 (Surrogate Function in TRPO)

(i)

Problem 1 Surrogate Function in TRPO

The surrogate function $L_{\pi_{\theta_1}}(\pi_\theta)$ is defined as

$$L_{\pi_{\theta_1}}(\pi_\theta) := \eta(\pi_{\theta_1}) + \sum_{s \in S} d_\mu^{\pi_{\theta_1}}(s) \sum_{a \in A} \pi_\theta(a|s) A^{\pi_{\theta_1}}(s,a)$$

Show that $L_{\pi_{\theta_1}}(\pi_\theta)$ satisfies the two properties

(i) $L_{\pi_{\theta_1}}(\pi_{\theta_1}) = \eta(\pi_{\theta_1})$

$$L_{\pi_{old}}(\pi_{new}) = \eta(\pi_{old}) + \sum_s d_\mu^{\pi_{old}}(s) \sum_a \pi_{new}(a|s) A^{\pi_{old}}(s,a)$$

$\Rightarrow$ $L_{\pi_{old}}(\pi_{new})$ satisfy two properties :

$\pi_{old} \equiv \pi_{\theta_1}$, $\pi_{new} \equiv \pi_\theta$

such that $L_{\pi_{\theta_1}}(\pi_{\theta_1}) = \eta(\pi_{\theta_1}) \equiv L_{\pi_{old}}(\pi_{old}) \equiv \eta(\pi_{\theta_{old}})$

Since the expected return of another policy $\tilde{\pi}$ in terms of the advantage over π, accumulated over timesteps : $\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \ldots \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right]$

And, we can rewrite this equation with a sum over states instead of timesteps :

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

where ρ_π be the discounted visitation frequencies also can be a distribution of the policy $\tilde{\pi}$.

This equation implies that any policy update $\pi \to \tilde{\pi}$ that has a non negative expected advantage at every state s, i.e. $\sum_a \tilde{\pi}(a|s) A_\pi(s,a) \geq 0$.

We will get constant in the case that expected advantage is zero everywhere.

$$\Rightarrow L_{\pi_{\theta_1}}(\pi_{\theta_1}) = \eta(\pi_{\theta_1}) + \sum_s d_\mu^{\pi_{\theta_1}}(s) \underbrace{\left(\sum_a \pi_{\theta_1}(s|a) A^{\pi_{\theta_1}}(s,a)\right)}_{0}$$

$$\therefore L_{\pi_{\theta_1}}(\pi_{\theta_1}) = \eta(\pi_{\theta_1}) \quad \#$$

It implies that a sufficiently small step $\pi_{\theta_1} \to \tilde{\pi}$ that improves $L_{\pi_{\theta_1}}$ will also improve η, but doesn't give any guidance on how big of step to take.

(ii)

(ii) $\nabla_\theta L_{\pi_{\theta_1}}(\pi_\theta)\big|_{\theta=\theta_1} = \nabla_\theta \eta(\pi_\theta)\big|_{\theta=\theta_1}$

$$\nabla_\theta L_{\pi_{\theta_1}}(\theta) = \nabla_\theta \eta(\pi_{\theta_1}) + \sum_s d_\mu^{\pi_{\theta_1}}(s) \sum_a \left(\nabla_\theta \pi_\theta(a|s)\right) A^{\pi_{\theta_1}}(s,a)$$

Since $\nabla_\theta \eta(\pi_{\theta_1}) = 0$

$$\Rightarrow \nabla_\theta L_{\pi_{\theta_1}}(\theta)\big|_{\theta=\theta_1} = \sum_s d_\mu^{\pi_{\theta_1}}(s) \sum_a \left(\nabla_\theta \pi_\theta(a|s)\big|_{\theta=\theta_1}\right) A^{\pi_{\theta_1}}(s,a) \quad \text{— ①}$$

since $\eta(\pi_\theta) = \underbrace{\eta(\pi_{\theta_1})}_{=0} + \sum_s d_\mu^{\pi_\theta}(s) \sum_a \pi_\theta(a|s) A^{\pi_{\theta_1}}(s,a)$

that $\nabla_\theta \eta(\pi_\theta) = \sum_s d_\mu^{\pi_\theta}(s) \sum_a \left(\nabla_\theta \pi_\theta(a|s)\right) A^{\pi_{\theta_1}}(s,a)$

$\Rightarrow \nabla_\theta \eta(\pi_\theta)\big|_{\theta=\theta_1}$

$= \sum_s d_\mu^{\pi_{\theta_1}}(s) \sum_a \left(\nabla_\theta \pi_\theta(a|s)\big|_{\theta=\theta_1}\right) A^{\pi_{\theta_1}}(s,a)$

With ① and ② $\nabla_\theta L_{\pi_{\theta_1}}(\pi_\theta)\big|_{\theta=\theta_1} = \nabla_\theta \eta(\pi_\theta)\big|_{\theta=\theta_1}$ #

2 Problem 2 (Solving TRPO Under Approximation Using Duality)

(a)

Problem 2 Solving TRPO Under Approximation Using Duality

(a) $D(\lambda) = \frac{-1}{2\lambda}\left(\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)^T H^{-1}\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)\right) - \lambda\delta$

As we know, $D(\lambda) := \min_{\theta \in R^d} \mathcal{L}(\theta,\lambda)$

, we can get θ^* from $\nabla_\theta \mathcal{L}(\theta,\lambda) = 0$

$\Rightarrow 0 = -\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right) + \lambda H(\theta-\theta_k)$

$\Rightarrow \theta^* - \theta_k = \frac{1}{\lambda} H^{-1}\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)$ —— ①

Change ① into (4)

$\Rightarrow D(\lambda) = -\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)^T \frac{H^{-1}}{\lambda}\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)$
$+ \frac{\lambda}{2}\left[\frac{H^{-1}}{\lambda}\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)\right]^T H \left[\frac{H^{-1}}{\lambda}\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)\right] - \lambda\delta$

$$= \frac{1}{2\lambda}\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)^T (\mathcal{H}^{-1})^T \left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)$$

$$\Rightarrow D(\lambda) = \frac{-1}{2\lambda}\left[\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)^T \mathcal{H}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)\right] - \lambda\delta \quad \#$$

We need to first solve $\frac{\partial D(\lambda)}{\partial \lambda} = 0$ to get λ^*

$$\frac{\partial D(\lambda)}{\partial \lambda} = \frac{1}{2\lambda^2}\left[\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)^T \mathcal{H}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)\right] - \delta = 0$$

$$\Rightarrow \lambda^* = \left(\frac{1}{2\delta}\left[\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)^T \mathcal{H}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)\right]\right)^{\frac{1}{2}}$$

$\#$

(b)

(b) Show that $\theta^* = \theta_k + \alpha \mathcal{H}^{-1} \nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}$ to get α

$$\because \theta^* - \theta_k = \frac{1}{\lambda}\mathcal{H}^{-1}\left(\nabla_\theta (L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)$$

$$\Rightarrow \theta^* = \theta_k + \frac{1}{\lambda^*}\mathcal{H}^{-1}\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)$$

$$= \theta_k + \alpha \mathcal{H}^{-1}\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)$$

Above all, $\alpha = \frac{1}{\lambda^*}$

$$= \sqrt{2\delta} \times \left[\left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)^T \mathcal{H}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)\big|_{\theta=\theta_k}\right)\right]^{-\frac{1}{2}}$$

$\#$

3 Problem 3 (Deep Deterministic Policy Gradient for Continuous Control)

(a)

```python
class Actor(nn.Module):
    def __init__(self, hidden_size, num_inputs, action_space):
        super(Actor, self).__init__()
        self.action_space = action_space
        num_outputs = action_space.shape[0]

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Construct your own actor network

        self.fc1 = nn.Linear(num_inputs, 400, device=device)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(400, 300, device=device)
        self.relu2 = nn.ReLU()
        self.fc3 = nn.Linear(300, num_outputs, device=device)
        self.tanh = nn.Tanh()

        ########## END OF YOUR CODE ##########

    def forward(self, inputs):

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Define the forward pass your actor network

        x = self.fc1(inputs)
        x = self.relu1(x)
        x = self.fc2(x)
        x = self.relu2(x)
        x = self.fc3(x)
        x = self.tanh(x)
        return x

        ########## END OF YOUR CODE ##########
```

Listing 1: Actor Network

```python
class Critic(nn.Module):
    def __init__(self, hidden_size, num_inputs, action_space):
        super(Critic, self).__init__()
        self.action_space = action_space
        num_outputs = action_space.shape[0]

        ######### YOUR CODE HERE (5~10 lines) ##########
        # Construct your own critic network

        self.state_layer = nn.Linear(num_inputs, 400, device=device)
        self.relu1 = nn.ReLU()

        self.shared_layer1 = nn.Linear(num_outputs + 400, 300, device=
    device)
```

```
        self.relu2 = nn.ReLU()
        self.shared_layer2 = nn.Linear(300, 1, device=device)

        ########## END OF YOUR CODE ##########

    def forward(self, inputs, actions):

        ########## YOUR CODE HERE (5~10 lines) ##########
        # Define the forward pass your critic network

        out = self.state_layer(inputs)
        out = self.relu1(out)
        out = torch.cat([out, actions], dim=1)
        out = self.shared_layer1(out)
        out = self.relu2(out)
        out = self.shared_layer2(out)
        return out

        ########## END OF YOUR CODE ##########
```

Listing 2: Critic Network

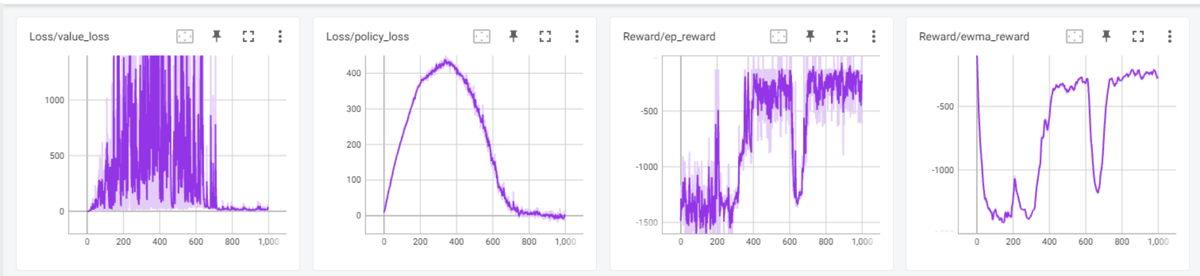
With the `update_parameters` function, here are steps:

1. Compute Q-value for the next state
2. Compute the target in the current state
3. Predict the Q-value in the current state
4. Compute the critic loss and try to minimize it
5. Predict action in the current state
6. Compute the actor loss and try to minimize it

For the parameters:

```
    num_episodes = 200
    gamma = 0.995
    tau = 0.002
    hidden_size = 128
    noise_scale = 0.3
    replay_size = 100000
    batch_size = 128
    updates_per_step = 1
    print_freq = 1
```

Listing 3: Pendulum-v1's parameters



For the task `Pendulum-v1`, after about 200 epochs, we can total train the agent based on the task's goal. And the parameter remain the same except the hidden layer. We change the hidden layer into 3 layers.

We save the model into a format called

- Actor model: `ddpg_actor_Pendulum-v1_05022023_155126_.pth`
- Critic model: `ddpg_critic_Pendulum-v1_05022023_155126_.pth`

(b)

For the parameters:

```
num_episodes = 1000
gamma = 0.995
tau = 0.002
hidden_size = 128
noise_scale = 0.3
replay_size = 100000
batch_size = 128
updates_per_step = 1
print_freq = 1
```

Listing 4: LunarLanderContinuous-v2's parameters

For the task `LunarLanderContinuous-v2`, after about 1000 epochs, we can total train the agent based on the task's goal. And the parameter remain the same except the hidden layer. We change the hidden layer into 3 layers.

We save the model into a format called

- Actor model: `ddpg_actor_LunarLanderContinuous-v2_05022023_185800_.pth`
- Critic model: `ddpg_critic_LunarLanderContinuous-v2_05022023_185800_.pth`