Deep Reinforcement Learning with Tianshou

Runcheng Liu 2018010316

1. Basic questions

1.1. Description the observation space of CartPole-v0

We can know the dimension of observation space of CartPole-v0 is 4 by print(env.observation_space). The 4 dimensions of CartPole-v0 are Cart Position, Cart Velocity, Pole Angle, Pole Angular Velocity.

1.2. Description the action space of CartPole-v0

We can know the dimension of action space of CartPole-v0 is 2 by print(env.action_space). The two actions are Push cart to the left or Push cart to the right.

1.3. Understanding of Policy Gradient algorithm

The objective of a Reinforement Learning agent is to maximize the "expected" reward when following a policy π. Like any machine learning setup, we define s set of parameters θ to parametrize this policy which is $\pi(s, a, \theta)$.

Then, in the policy gradient approach, the policy parameters are updated appoximately proportional to the gradient:

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta} \tag{1}$$

where α is a positive-definite step size.

There are several advantages compared to value-function approach. First, the optimal policy is often stochastic, selecting different actions with specific probabilities. But the value-function approach is oriented toward finding deterministic policies. Second, an arbitrarily small change in the estimated value of an action can cause it to be, or not to be selected. While, in the policy gradient approach, here small changes in θ ca cause only small changes in the policy and in the state-visitation distribution. Thus, θ can usually be assured to converge to a locally optimal policy in the performance measure ρ.

And with function approximation, as it says in the paper, two way of formulating the agent's objective are useful. One is the average reward formulation, in which policies are ranked according to their long-term reward per step, $\rho(\pi)$:

$$\rho(\pi) = \lim_{n\to\infty} \frac{1}{n} E\left\{r_1 + r_2 + \cdots + r_n \mid \pi\right\} = \sum_s d^\pi(s) \sum_a \pi(s, a) \mathcal{R}_s^a \tag{2}$$

where$d^\pi(s) = \lim_{t\to\infty} \Pr\{s_t = s \mid s_0, \pi\}$ is the stationary distrubution of states under π, which we assume exists and is independent of s_0 for all polices. In the average reward formulation, the value of a state-action pair given a policy is defined as

$$Q^\pi(s,a) = \sum_{t=1}^{\infty} E\{r_t - \rho(\pi) \mid s_0 = s, a_0 = a, \pi\}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{3}$$

The second formulation is that in which there is a designated start state s_0, in this case, is defines as

$$\rho(\pi) = E\left\{\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_0, \pi\right\} \quad \text{and} \quad Q^\pi(s,a) = E\left\{\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \mid s_t = s, a_t = a, \pi\right\} \tag{4}$$

where$\gamma \in [0,1]$ is a discount rate. In this case, $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr\{s_t = s \mid s_0, \pi\}$.

And further in the paper, it mentions the Policy Gradient theorem which is, for any MDP, in either the average-reward or start-state formulations,

$$\frac{\partial \rho}{\partial \theta} = \sum_{\boldsymbol{s}} \boldsymbol{d}^\pi(s) \sum_{\boldsymbol{a}} \frac{\partial \pi(\boldsymbol{s}, \boldsymbol{a})}{\partial \boldsymbol{\theta}} Q^\pi(\boldsymbol{s}, \boldsymbol{a}) \tag{5}$$

The key feature of both expressions for the gradient is that there are no terms of the form $\frac{\partial d^\pi(s)}{\partial \theta}$,: the effect of policy changes on the distribution of states does not appear. This is convenient for approximating the gradient by sampling. Thus, we would be able to obtain an unbiased estimate of $\frac{\partial \rho}{\partial \theta}$.

1.4. Method(on-policy or off-policy) used in this experiment and the difference of on-policy and off-policy

This experiment is using the vanilla version of actor-critic method which is on-policy. We can see this from the beginning of homework.py file. It has imported PGPolicy from tianshou.policy. From document of tianshou, we can know that PGPolicy is a implementation of Vanilla Policy Gradient. Besides, We can know that this experiment is using on-policy from the import of onpolicy_trainer from tianshou.trainer.

For better explanation of the difference of on-policy and off-policy, I take Q-learning and SARSA as examples. Q-learning is off-policy is that it updates its Q-value using the Q-value of the next state s' and the greedy action a'. In other words, it estimates the return for state-action pairs assuming a greedy policy were followed despite the fact that it's not following a greedy policy. While, SARSA is on-policy is that is updates its Q-value using the Q-value of the next state s' and the current policy's action a'. But the distinction disappears if the current policy is a greedy policy. However, such an agent would not be a good agent since it never explores.

2. Bonus questions