Reinforcement Learning

Rui Wu

I. Monte Carlo method and TD learning

This section considers several methods for learning the optimal policy. We only consider episode environments.

Let $Q(s,a)$ be the state-action value function and $\pi(a|s)$ be the ϵ-greedy policy derived from $Q(s,a)$. According to the policy improvement theorem, we start from an initial policy and keep improving it towards the optimal policy by iteratively estimating $Q(s,a)$ and updating $\pi(a|s)$. For each episode, we obtain the samples as

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots, r_T, s_T(, a_T).$$

Then we estimate the sample value for each (s_t, a_t) and update $Q(s,a)$ with these sample values.

A. Batch Monte Carlo

Monte Carlo method estimates G_t as

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-1} r_T.$$

For batch update, we first compute the average reward for each state-action pair as

$$G(s,a) = \frac{\sum_t G_t \delta_{s,a}(s_t, a_t)}{\sum_t \delta_{s,a}(s_t, a_t)},$$

and then update $Q(s,a)$ as

$$Q(s,a) \leftarrow Q(s,a) + \alpha(G(s,a) - Q(s,a)).$$

B. Online Monte Carlo

Online Monte Carlo looks similar to TD learning and is easier to implement. It updates $Q(s,a)$ as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(G_t - Q(s_t, a_t)),$$

where G_t is defined as in batch Monte Carlo. Even though this method is called "online", it has to wait until an episode ends to actually compute the G_t's.

C. Sarsa

It is similar to online Monte Carlo except that it computes G_t as

$$G_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}).$$

D. Q-learning

It is similar to online Monte Carlo except that it computes G_t as

$$G_t = r_{t+1} + \gamma \max_a Q(s_{t+1}, a).$$

II. Eligibility trace

This section focuses on appliying eligibility trace to $Q(s,a)$ than $V(s)$ and describes the Sarsa(λ) algorithm in detail.

A. Forward view of Sarsa(λ)

We first generalizes the Sarsa reward G_t to the n-step reward

$$G_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{(n-1)} r_{t+n} + \gamma^n Q(s_{t+n}, a_{t+n}),$$

and define

$$G_t^{\lambda} = (1-\lambda) G_t^{(1)} + (1-\lambda)\lambda G_t^{(2)} + \cdots + \lambda^{T-t} G_t^{(T-t)}.$$

If we assume an episode has infinite length and uss the assumption that the reward for a termination state remains the same, we can rewrite G_t^{λ} as

$$G_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}.$$

Then we update $Q(s,a)$ as

$$Q(s,a) \leftarrow Q(s,a) + \alpha(G_t^{\lambda} - Q(s,a)).$$

This algorithm is between Monte Carlo and Sarsa. It incorporates a parameter λ to balance how much into future we want to consider for reward estimation. However, to implement this forward view algorithm, we have to wait until an episode ends to start updating $Q(s,a)$, which is similar to online Monte Carlo.

B. Backward view of Sarsa(λ)

The backward view addresses the online update issue by introducing eligibility trace. The eligibility trace updates as

$$Z(s,a) \leftarrow \lambda\gamma Z(s,a) + \delta_{s,a}(s_t, a_t),$$

and the algorithm updates $Q(s,a)$ as

$$Q(s,a) \leftarrow Q(s,a) + \alpha Z(s,a)(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)).$$

Note that this algorithm updates all states for each time t. This is very expensive compared to Sarsa or Monte Carlo. Is it really worth it?

C. Connection between forward and backward views

The forward and backward views are equivalent in an episode environments under first visit and batch update, i.e., $Q(s, a)$ is updated in batch after an episode ends. To illustrate this equivalency, we walk through the following example for policy evaluation.

Example 1. *Consider an episode as follows*

$$s_1, a_1, r_2, s_2, a_2, r_3, s_1, a_3, r_4, s_4.$$

Forward view. The estimated value for the first visit of state s_1 is

$$\begin{aligned} G^\lambda(s_1) =& (1-\lambda)(r_2 + \gamma V(s_2)) \\ & (1-\lambda)\lambda(r_2 + \gamma r_3 + \gamma^2 V(s_1)) \\ & \lambda^2(r_2 + \gamma r_3 + \gamma^2 r_4 + \gamma^3 V(s_4)). \end{aligned}$$

The value difference is

$$\Delta V_F(s_1) = G^\lambda(s_1) - V(s_1).$$

Backward view. The eligibility trace for s_1 updates as

$$Z(s_1) = 1 \to \lambda\gamma \to 1 + \lambda^2\gamma^2.$$

The value difference is

$$\begin{aligned} \Delta V_B(s_1) =& 1 \cdot (r_2 + \gamma V(s_2) - V(s_1)) \\ & \lambda\gamma \cdot (r_3 + \gamma V(s_1) - V(s_2)) \\ & (1 + \lambda^2\gamma^2)(r_3 + \gamma V(s_4) - V(s_1)). \end{aligned}$$

It is not difficult to show that $\Delta V_F(s_1) = \Delta V_B(s_1)$. However, this equality is not obvious from the first glance.

D. Other eligibility trace algorithms

There is no MC(λ) as Monte Carlo method does not do bootstrap. It is possible to do $Q(\lambda)$ but it seems more involved.

III. Policy gradient and imitation learning

A. Policy gradient

B. Imitation learning

C. Policy gradient as imitation learning

D. DAgger