Reinforcement Learning

Notes

Jingye Wang

wangjy5@shanghaitech.edu.cn

Spring 2020

Contents

6 Model-Free Prediction

To begin, we firstly give definition of the term *episode* then give the prediction algorithm based MC and TD methods.

Episodes: *An episode τ is a sequence of states and actions in the environment,*

$$\tau = (s_0, a_0, s_1, a_1, ...).$$

We call an episode is complete if it ends with the terminal state.

In model free method, we only utilize the episode itself without further exploitation.

6.1 Monte-Carlo Policy Evaluation

Monte-Carlo (MC) policy evaluation methods learn directly from the *complete* episodes, thus it needs no knowledge of MDP transitions or rewards. The limitation of MC policy method methods is that it requires MDPs are episodic.

Monte-Carlo Policy Evaluation: *Given some complete episodes under the policy π, we can approximate the value of s by the average returns observed after visiting to s.*

Depending on when average returns for state s in an episode, there are two different implementations.

First-Visit Monte-Carlo Policy Evaluation: *Only at the first time-step t that state s is visited in an episodes, we do the following procedure:*

- Increment counter $N(s) \leftarrow N(s) + 1$;
- Increment the total return $return(s) \leftarrow return(s) + G_t$;
- Update the value by the mean return $v(s) \leftarrow return(s)/N(s)$.

Every-Visit Monte-Carlo Policy Evaluation: *Every time-step t that state s is visited in an episode, we do the following procedure:*

- Increment counter $N(s) \leftarrow N(s) + 1$;
- Increment the total return $return(s) \leftarrow return(s) + G_t$;
- Update the value by the mean return $v(s) \leftarrow return(s)/N(s)$.

Algorithm 10 First-Visit Monte-Carlo Policy Evaluation

initialize $v(s) \in \mathbb{R}$ arbitrarily for all $s \in \mathcal{S}$;
initialize $return(s) \leftarrow$ an empty list for all $s \in \mathcal{S}$;
input the policy π to be evaluated;
for true **do**:
    # variants for this alg. can be start with $s_0 \in \mathcal{S}, a_0 \in \mathcal{A}(s_0)$ randomly, $\varepsilon-$greedy, *etc.*
    Generate a complete episode $\tau = (s_0, a_0, r_1, ..., s_{T-1}, a_{T-1}, r_T)$;
    $G \leftarrow 0$;
    for $t = T-1, T-2, ...0$ **do**:
        $G \leftarrow \gamma G + r_{t+1}$;
        if s_t appears in $(s_0, s_1, ..., s_{t-1})$ **then**:
            Append G to $return(s_t)$;
            $v(s_t) \leftarrow$average($return(s_t)$);
        end if
    end for
end for

Algorithm 11 Every-Visit Monte-Carlo Policy Evaluation

input the policy π to be evaluated;
initialize $v(s) \in \mathbb{R}$ arbitrarily for all $s \in \mathcal{S}$;
initialize $return(s) \leftarrow$ an empty list for all $s \in \mathcal{S}$;
for true **do**:
    Generate a complete episode $\tau = (s_0, a_0, r_1, ..., s_{T-1}, a_{T-1}, r_T)$;
    for $t = T-1, T-2, ..., 0$ **do**:
        $G \leftarrow \gamma G + r_{t+1}$;
        Append G to $return(s_t)$;
        $v(s_t) \leftarrow$average($return(s_t)$);
    end for
end for

By *law of large numbers*, both of two methods can achieve $v(s) \to v^{\pi}(s)$ as $N(s) \to \infty$. In practice, we can use a trick *incremental mean* to simplify the calculation.

Differences between DP and MC for policy evaluation:

- DP computes v_k by bootstrapping the rest of the expected return calculated with v_{k-1};
- DP iterates on Bellman expectation backup:

$$v_k(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_{k-1}(s') \right).$$

- MC updates the empirical mean return with a sampled episode:

$$v(s_t) \leftarrow v(s_t) + \alpha(G_{k,t} - v(s_t)).$$

Advantages of MC over DP:

- MC can work when the environment is unknown;
- Working with sampled episodes has a huge advantage. Even with the complete knowledge of the environment's dynamics, the complexity still could be a challenge;
- Cost of estimating a single state's value is independent of the total number of states. So one can sample episodes starting from the states of interest then average returns.

6.2 Temporal-Difference Learning

Unlike MC methods, temporal-difference (TD) does not require the episodes are complete. TD methods can learn from incomplete episodes by bootstrapping.

Temporal-Difference Learning: *Given some incomplete episodes under the policy π, we update value $v(s_t)$ toward estimated return $r_{t+1} + \gamma v(s_{t+1})$:*

$$v(s_t) \leftarrow v(s_t) + \alpha(r_{t+1} + \gamma v(s_{t+1}) - v(s_t)),$$

where $r_{t+1} + \gamma v(s_{t+1})$ is called the TD target, α is the step-size, and we call

$$\delta_t = r_{t+1} + \gamma v(s_{t+1}) - v(s_t)$$

the TD error.

Algorithm 12 TD(0) Evaluation

```
initialize v(s) ∈ ℝ arbitrarily for all s ∈ S except that v(terminal) = 0;
input the policy π to be evaluated; the step size α ∈ (0, 1];
for true do:
    Generate initial state s;
    while s is not terminal do:
        a ← π(s);
        r, s′ ← environment(s, a);
        v(s) ← v(s) + α(r + γv(s′) − v(s));
        s ← s′;
    end while
end for
```

Differences between MC and TD for policy evaluation:

- TD can learn online after every step;
- MC must wait until end of episode before return is known;
- TD can learn from incomplete sequences;
- MC can only learn from complete sequences;
- TD works in continuing (non-terminating) environments;
- MC only works in episodic (terminating) environments;
- TD exploits Markov property, and it is efficient in Markov environments;
- MC does not exploit Markov property, thus it is relatively effective in non-Markov environments.

Bootstrapping: involves old values, it is something like in-place update.

Sampling: samples to get an expectation.

A summary of *bootstrapping* and *sampling* for DP, MC, and TD is shown in Table 2.

Algorithm	Bootstraps	Sampling
Dynamic Programming	$\surd$	
Monte-Carlo		$\surd$
Temporal-Difference	$\surd$	$\surd$

Table 2: *Bootstrapping* and *sampling* for DP, MC, and TD.

n-step TD methods: *Unlike the simplest TD method, in n-step TD methods, the updating rule of value $v(s_t)$ is*

$$v(s_t) \leftarrow v(s_t) + \alpha(G_t^{(n)} - v(S_t)),$$

where $G_t^{(n)}$ is the n-step return

$$G_t^{(n)} = r_{t+1} + \gamma r_{t+2} + ... + \gamma^n v(s_{t+n}).$$

Notice that with additional definition, we can generalize TD to MC when $n \to \infty$.

TD(λ) methods: *To make use of the information from all time-steps, we can use weight $(1-\lambda)\lambda^{n-1}$ to average n-step returns over different n as*

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

thus the updating rule for the value becomes

$$v(s_t) \leftarrow v(s_t) + \alpha(G_t^\lambda - v(s_t)).$$