

Lecture 13 - Model-Free Policy Evaluation

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning

Course Page: [\[Click\]](#)

Monte-Carlo Policy Evaluation

An example of the Monte-Carlo method: Suppose we want to estimate how long the commute from your house to the campus will take today.

- We have access to a **commute simulator** that models our uncertainty of how bad the traffic will be, the weather, construction delays, and other variables, as well as how these variables interact with each other.
- We estimate the expected commute time by **simulating our commute many times** on the simulator and then take an **average over the simulated commute times**.

This is called a Monte-Carlo estimate of our commute time. Monte-Carlo method only works in episodic environments



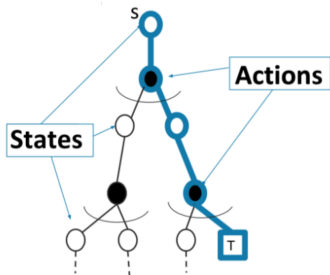
Monte-Carlo Policy Evaluation

In the context of reinforcement learning, the quantity we want to estimate is $V^\pi(s)$, which is the average of returns G_t (which equals R_t without n -step truncate or eligibility traces) under policy π starting at state s . We can thus get a Monte-Carlo estimate of $V^\pi(s)$ through three steps:

1. Execute a rollout of policy π until termination many times;
2. Record the returns G_t that we observe when starting at state s ;
3. Take an average of the values we get for G_t to estimate $V^\pi(s)$.



Monte-Carlo Policy Evaluation



The **backup diagram for Monte-Carlo policy evaluation**. The new blue line indicates that we sample an entire episode until termination starting at state s .

 = Expectation
 = **Terminal state**



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Monte-Carlo Policy Evaluation

First-visit Monte-Carlo: Take an average over just the **first time** we visit a state in each rollout.

Algorithm 1: First-visit Monte-Carlo policy evaluation

```
Input:  $h_1, \dots, h_j$   
For all states  $s$ ,  $N(s) \leftarrow 0$ ,  $S(s) \leftarrow 0$ ,  $V(s) \leftarrow 0$   
for each episode  $h_j$  do  
    for  $t = 1, \dots, L_j$  do  
        if  $s_{j,t} \neq s_{j,u}$  for  $u < t$  then  
             $N(s_{j,t}) \leftarrow N(s_{j,t}) + 1$   
             $S(s_{j,t}) \leftarrow S(s_{j,t}) + G_{j,t}$   
             $V^\pi(s_{j,t}) \leftarrow S(s_{j,t})/N(s_{j,t})$   
return  $V^\pi$ 
```



Monte-Carlo Policy Evaluation

Every-visit Monte-Carlo: Take an average over **every time** we visit the state in each rollout. If we are in a **truly Markovian-domain**, **every-visit** Monte Carlo will be more **data efficient** because we update our average return for a state every time we visit the state.

Algorithm 2: Every-visit Monte-Carlo policy evaluation

```
Input:  $h_1, \dots, h_j$   
For all states  $s$ ,  $N(s) \leftarrow 0$ ,  $S(s) \leftarrow 0$ ,  $V(s) \leftarrow 0$   
for each episode  $h_j$  do  
    for  $t = 1, \dots, L_j$  do  
         $N(s_{j,t}) \leftarrow N(s_{j,t}) + 1$   
         $S(s_{j,t}) \leftarrow S(s_{j,t}) + G_{j,t}$   
         $V^\pi(s_{j,t}) \leftarrow S(s_{j,t})/N(s_{j,t})$   
return  $V^\pi$ 
```



Monte-Carlo Policy Evaluation

In these Algorithms, we can remove vector S and replace the update for $V^\pi(s_{j,t})$ with

$$V^\pi(s_{j,t}) \leftarrow V^\pi(s_{j,t}) + \frac{1}{N(s_{j,t})} (G_{j,t} - V^\pi(s_{j,t})).$$

This is because the new average is **the average of $N(s_{j,t}) - 1$ of the old values $V^\pi(s_{j,t})$ and the new return $G_{j,t}$** , giving us

$$\frac{V^\pi(s_{j,t}) \cdot (N(s_{j,t}) - 1) + G_{j,t}}{N(s_{j,t})} = V^\pi(s_{j,t}) + \frac{1}{N(s_{j,t})} (G_{j,t} - V^\pi(s_{j,t})),$$

Replacing $1/N(s_{j,t})$ with α in this new update gives us the more general **incremental**

Monte-Carlo policy evaluation.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Monte-Carlo Policy Evaluation

Incremental First-visit Monte-Carlo policy evaluation:

Algorithm 3: Incremental first-visit Monte-Carlo policy evaluation

Input: α, h_1, \dots, h_j
For all states s , $N(s) \leftarrow 0$, $V(s) \leftarrow 0$
for each episode h_j **do**
 for $t = 1, \dots, \text{terminal}$ **do**
 if $s_{j,t} \neq s_{j,u}$ for $u < t$ **then**
 $N(s_{j,t}) \leftarrow N(s_{j,t}) + 1$
 $V^\pi(s_{j,t}) \leftarrow V^\pi(s) + \alpha(G_{j,t} - V^\pi(s))$
 return V^π



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Monte-Carlo Policy Evaluation

Incremental Every-visit Monte-Carlo policy evaluation:

Algorithm 4: Incremental every-visit Monte-Carlo policy evaluation

Input: α, h_1, \dots, h_j

For all states s , $N(s) \leftarrow 0$, $V(s) \leftarrow 0$

for each episode h_j **do**

for $t = 1, \dots, \text{terminal}$ **do**

$N(s_{j,t}) \leftarrow N(s_{j,t}) + 1$

$V^\pi(s_{j,t}) \leftarrow V^\pi(s) + \alpha(G_{j,t} - V^\pi(s))$

return V^π

Setting $\alpha = 1/N(s_{j,t})$ recovers the original Monte-Carlo policy evaluation algorithms given in the above Algorithms, while setting $\alpha > \frac{1}{N(s)}$ gives a higher weight to newer data, which can help learning in non-stationary domains.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Monte-Carlo Off-Policy Evaluation

Motivation:

- In the above, we discussed the case where we are able to obtain many realizations of G_t under the policy π that we want to evaluate.
- However, in many costly or high-risk situations, we are unable to obtain rollouts of G_t under the policy that we wish to evaluate.
- In this section, we describe Monte-Carlo off-policy policy evaluation, a method for using data from one policy to evaluate a different policy.



Monte-Carlo Off-Policy Evaluation

Importance Sampling: that estimates the expected value of a function $f(x)$ when x is drawn from the distribution q using only the data $f(x_1), \dots, f(x_n)$, where x_i are drawn from a different distribution p . In summary, given $q(x_i), p(x_i), f(x_i)$ for $1 \leq x_i \leq n$, we would like an estimate for $\mathbb{E}_{x \sim q}[f(x)]$. We can do this via the approximation:

$$\begin{aligned}\mathbb{E}_{x \sim q}[f(x)] &= \int_x q(x) f(x) dx \\ &= \int_x p(x) \left[\frac{q(x)}{p(x)} f(x) \right] dx \\ &= \mathbb{E}_{x \sim p} \left[\frac{q(x)}{p(x)} f(x) \right] \\ &\approx \sum_{i=1}^n \left[\frac{q(x_i)}{p(x_i)} f(x_i) \right].\end{aligned}$$



Monte-Carlo Off-Policy Evaluation

Importance sampling for off-policy policy evaluation: We apply importance sampling estimates to reinforcement learning. In this instance, we want to approximate the value of state s under policy π_1 , given by $V^{\pi_1}(s) = \mathbb{E}[G_t \mid s_t = s]$, using n histories h_1, \dots, h_n generated under policy π_2 . The importance sampling estimate result provides:

$$V^{\pi_1}(s) \approx \frac{1}{n} \sum_{j=1}^n \frac{\mathbb{P}(h_j \mid \pi_1, s)}{\mathbb{P}(h_j \mid \pi_2, s)} G(h_j),$$

where $G(h_j) = \sum_{t=1}^{L_j-1} \gamma^{t-1} r_{j,t}$ is the total discounted sum of rewards for history h_j .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Monte-Carlo Off-Policy Evaluation

Now, for a general policy π , we have that the probability of experiencing history h_j under policy π is

$$\mathbb{P}(h_j \mid \pi, s = s_{j,1}) = \prod_{t=1}^{L_j-1} \mathbb{P}(a_{j,t} \mid s_{j,t}) \mathbb{P}(r_{j,t} \mid s_{j,t}, a_{j,t}) \mathbb{P}(s_{j,t+1} \mid s_{j,t}, a_{j,t})$$

where L_j is the length of the j -th episode. In each transition, the components are 1) $\mathbb{P}(a_{j,t} \mid s_{j,t})$ - probability we take action $a_{j,t}$ at state $s_{j,t}$; 2) $\mathbb{P}(r_{j,t} \mid s_{j,t}, a_{j,t})$ - probability we experience reward $r_{j,t}$ after taking action $a_{j,t}$ in state $s_{j,t}$; 3) $\mathbb{P}(s_{j,t+1} \mid s_{j,t}, a_{j,t})$ - probability we transition to state $s_{j,t+1}$ after taking action $a_{j,t}$ in state $s_{j,t}$.



Monte-Carlo Off-Policy Evaluation

Combining our importance sampling estimate for $V^{\pi_1}(s)$ with our decomposition of the history probabilities, $\mathbb{P}(h_j \mid \pi, s = s_{j,1})$, we get that

$$\begin{aligned} V^{\pi_1}(s) &\approx \frac{1}{n} \sum_{j=1}^n \frac{\mathbb{P}(h_j \mid \pi_1, s)}{\mathbb{P}(h_j \mid \pi_2, s)} G(h_j) \\ &= \frac{1}{n} \sum_{j=1}^n \frac{\prod_{t=1}^{L_j-1} \pi_1(a_{j,t} \mid s_{j,t}) \mathbb{P}(r_{j,t} \mid s_{j,t}, a_{j,t}) \mathbb{P}(s_{j,t+1} \mid s_{j,t}, a_{j,t})}{\prod_{t=1}^{L_j-1} \pi_2(a_{j,t} \mid s_{j,t}) \mathbb{P}(r_{j,t} \mid s_{j,t}, a_{j,t}) \mathbb{P}(s_{j,t+1} \mid s_{j,t}, a_{j,t})} G(h_j) \\ &= \frac{1}{n} \sum_{j=1}^n G(h_j) \prod_{t=1}^{L_j-1} \frac{\pi_1(a_{j,t} \mid s_{j,t})}{\pi_2(a_{j,t} \mid s_{j,t})}. \end{aligned}$$



Temporal Difference Learning

Motivation. A recap of the policy evaluation methods:

- **Dynamic programming** leverages bootstrapping to help us get value estimates with only one backup.
- **Monte Carlo** samples many histories for many trajectories which frees us from using a model.
- **Temporal difference learning** combines bootstrapping with sampling to give us a new model-free policy evaluation algorithm.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Temporal Difference Learning

To see how to combine sampling with bootstrapping, we go back to our incremental Monte-Carlo update

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha(G_t - V^\pi(s_t)).$$

We replace G_t with a Bellman backup like $r_t + \gamma V^\pi(s_{t+1})$, where r_t is a sample of the reward at time step t and $V^\pi(s_{t+1})$ is our current estimate of the value at the next state. It gives us the temporal difference (TD) learning update

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha(r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)).$$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Temporal Difference Learning

The **TD error** is given by:

$$\delta_t = r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$

The sampled reward combined with the bootstrap estimate of the next state value, i.e., the **TD target** is given by:

$$r_t + \gamma V^\pi(s_{t+1}),$$

We can see that using this method, we update our value for $V^\pi(s_t)$ directly after witnessing the transition (s_t, a_t, r_t, s_{t+1}) .



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Temporal Difference Learning

Algorithm 5: TD Learning to evaluate policy π

Input: step size α , number of trajectories n

For all states s , $V^\pi(s) \leftarrow 0$

while $n > 0$ **do**

 Begin episode E at state s

while *episode E has not terminated* **do**

$a \leftarrow$ action at state s under policy π

 Take action a in E and observe reward r , next state s'

$V^\pi(s) \leftarrow V^\pi(s) + \alpha(r + \gamma V^\pi(s') - V^\pi(s))$

$s \leftarrow s'$

$n \leftarrow n - 1$

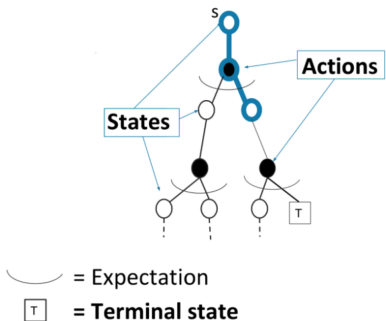
return V^π



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Temporal Difference Learning



Here, we see via the blue line that we **sample one transition starting at s** , then we estimate the **value of the next state** via our current estimate of the next state to construct a **full Bellman backup estimate**.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Temporal Difference Learning

Remark. There is actually an entire spectrum of ways we can **blend Monte Carlo and dynamic programming** using a method called $TD(\lambda)$.

- When $\lambda = 0$, we get the TD learning, hence giving us the alias $TD(0)$.
- When $\lambda = 1$, we recover the Monte-Carlo policy evaluation.
- When $0 < \lambda < 1$, we get a blend of these two methods.

For a more thorough treatment of $TD(\lambda)$, we refer the interested reader to Sections 7.1 and 12.1-12.5 of *Reinforcement learning: An introduction*.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Batch Monte-Carlo sampling and temporal difference

We consider the batch cases of Monte Carlo and TD(0).

- In the batch case, we are given a batch, or set of histories h_1, \dots, h_n , which we then feed through Monte Carlo or TD(0) many times.
- The only difference from our formulations before is that we only update the value function after each time we process the entire batch.



Batch Monte-Carlo sampling and temporal difference

Motivation Example. Suppose $\gamma = 1$ and we have eight histories generated by policy π , take action a_1 in all states:

$$h_1 = (A, a_1, +0, B, a_1, +0, \text{terminal})$$

$$h_j = (B, a_1, +1, \text{terminal}) \text{ for } j = 2, \dots, 7$$

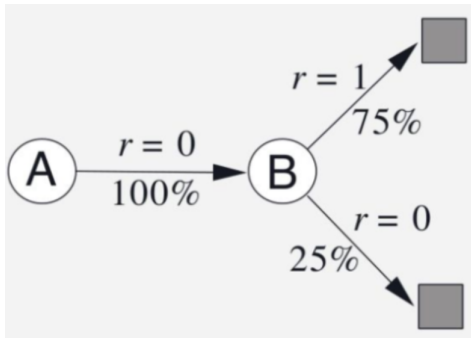
$$h_8 = (B, a_1, +0, \text{terminal}).$$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Batch Monte-Carlo sampling and temporal difference



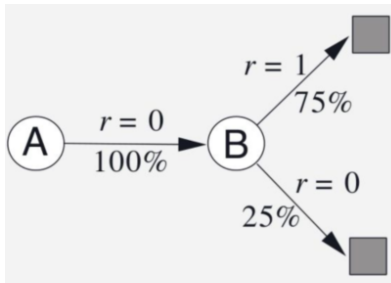
In this example, using either batch Monte Carlo or TD(0) with $\alpha = 1$, we see that $V(B) = 0.75$.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Batch Monte-Carlo sampling and temporal difference



However, if we use

- Monte Carlo, we get that $V(A) = 0$ since only the first episode visits state A and has return 0.
- TD(0) giving us $V(A) = 0.75$ because we perform the update $V(A) \leftarrow r_{1,1} + \gamma V(B)$. The estimate given by TD(0) makes more sense.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Question and Answering (Q&A)



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen