

Reinforcement Learning

Notes

Jingye Wang

✉ wangjy5@shanghaitech.edu.cn

Spring 2020

Contents

1	Introduction	3
2	Review of Basic Probability	5
2.1	Interpretation of Probability	5
2.2	Transformations	5
2.3	Limit Theorem	5
2.4	Sampling & Monte Carlo Methods	6
2.5	Basic Inequalities	8
2.6	Concentration Inequalities	10
2.7	Conditional Expectation	12
3	Bandit Algorithms	14
3.1	Bandit Models	14
3.2	Stochastic Bandits	14
3.3	Greedy Algorithms	15
3.4	UCB Algorithms	16
3.5	Thompson Sampling Algorithms	17
3.6	Gradient Bandit Algorithms	18
4	Markov Chains	20
4.1	Markov Model	20
4.2	Basic Computations	20
4.3	Classifications	21

CONTENTS	2
4.4 Stationary Distribution	22
4.5 Reversibility	22
4.6 Markov Chain Monte Carlo	23
5 Markov Decision Process	25
5.1 Markov Reward Process	25
5.2 Markov Decision Process	26
5.3 Dynamic Programming	28
6 Model-Free Prediction	33
6.1 Monte-Carlo Policy Evaluation	33
6.2 Temporal-Difference Learning	35
7 Model-Free Control	38
7.1 On Policy Monte-Carlo Control	38
7.2 On Policy Temporal-Difference Control: Sarsa	40
7.3 Off-Policy Temporal-Difference Control: Q-Learning	41
8 Value Function Approximation	42
8.1 Introduction on Function Approximation	42
8.2 Incremental Method	42
8.3 Batch Methods	44
8.4 Deep Q-Learning	45
9 Policy Optimization	46
9.1 Policy Optimization	47
9.2 Monte-Carlo Policy Gradient	48
9.3 Actor-Critic Policy Gradient	51
9.4 Extension of Policy Gradient	52

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,

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

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 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 $v(s) \leftarrow return(s)/N(s)$.

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

```

1: initialize  $v(s) \in \mathbb{R}$  arbitrarily for all  $s \in \mathcal{S}$ ;
2: initialize  $return(s) \leftarrow$  an empty list for all  $s \in \mathcal{S}$ ;
3: input the policy  $\pi$  to be evaluated;
4: 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.
5:   Generate a complete episode  $\tau = (s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T)$ ;
6:    $G \leftarrow 0$ ;
7:   for  $t = T - 1, T - 2, \dots, 0$  do:
8:      $G \leftarrow \gamma G + r_{t+1}$ ;
9:     if  $s_t$  appears in  $(s_0, s_1, \dots, s_{t-1})$  then:
10:      Append  $G$  to  $return(s_t)$ ;
11:       $v(s_t) \leftarrow \text{average}(return(s_t))$ ;
12:     end if
13:   end for
14: end for

```

Algorithm 11 Every-Visit Monte-Carlo Policy Evaluation

```

1: input the policy  $\pi$  to be evaluated;
2: initialize  $v(s) \in \mathbb{R}$  arbitrarily for all  $s \in \mathcal{S}$ ;
3: initialize  $return(s) \leftarrow$  an empty list for all  $s \in \mathcal{S}$ ;
4: for true do:
5:   Generate a complete episode  $\tau = (s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T)$ ;
6:   for  $t = T - 1, T - 2, \dots, 0$  do:
7:      $G \leftarrow \gamma G + r_{t+1}$ ;
8:     Append  $G$  to  $return(s_t)$ ;
9:      $v(s_t) \leftarrow \text{average}(return(s_t))$ ;
10:  end for
11: end for

```

By law of large numbers, both of two methods can achieve $v(s) \rightarrow v^\pi(s)$ as $N(s) \rightarrow \infty$. 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

```

1: initialize  $v(s) \in \mathbb{R}$  arbitrarily for all  $s \in \mathcal{S}$  except that  $v(\text{terminal}) = 0$ ;
2: input the policy  $\pi$  to be evaluated; the step size  $\alpha \in (0, 1]$ ;
3: for true do:
4:   Generate initial state  $s$ ;
5:   while  $s$  is not terminal do:
6:      $a \leftarrow \pi(s)$ ;
7:      $r, s' \leftarrow \text{environment}(s, a)$ ;
8:      $v(s) \leftarrow v(s) + \alpha(r + \gamma v(s') - v(s))$ ;
9:      $s \leftarrow s'$ ;
10:  end while
11: 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	✓	
Monte-Carlo		✓
Temporal-Difference	✓	✓

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

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

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

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

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