CS489 HW2 Report

Sheng Cen

March 2021

1 Introduction

In unknown Markov reward/decision process, we cannot know the state transition probability or immediate reward for each state directly. Therefore, the only way for us to get state value function is to do experiment with large number of sample. In this report, Monte-Carlo (first-visit and every-visit) and Temporal-Difference (TD0) learning are tested to generate stable value function for the grid world.

2 Setting of Grid World

In the 6*6 grid world we have set up (see Figure 1), each grid represents a state. Grid 1 and 35 are two terminated states, in which we expect to reach as quickly as possible. In each state, four actions (towards east, west, south and north) could be taken, and in order to reach terminated states quickly, we penalize and assign a reward of -1 (we do not know in advance) for each action. Also, actions are bounded in the 6*6 grid world, meaning that agent could not get out of the world. The policy we analyze is random policy, i.e., each direction we take in any non-terminated state is assigned to a probability of 1/4.

3 Details about Monte-Carlo and Temporal-Difference Learning

3.1 Monte-Carlo Learning

For Monte-Carlo Learning, we need to sample an episode each time, to calculate each visited state's actual return. And the expected return is estimated by the average return after many rounds of sampling. There are two kinds of learning methods, first-visit and every-visit. The former one only cares about the first-time visited states, so at most one return value is sampled after an episode, while the latter cares about every-state visited states, so more sampling returns are obtained. The pseudocode for the first-visit MC learning is shown in Figure 2.

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Figure 1: The Grid World

First-visit MC prediction, for estimating $V \approx v_\pi$

Initialize:
    $\pi \leftarrow$ policy to be evaluated
    $V \leftarrow$ an arbitrary state-value function
    $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:
    Generate an episode using π
    For each state s appearing in the episode:
        $G \leftarrow$ return following the first occurrence of s
        Append G to $Returns(s)$
        $V(s) \leftarrow$ average($Returns(s)$)

Figure 2: Pseudocode for Monte-Carlo (First-Visit) Learning

3.2 Temporal-Difference Learning

For Temporal-Difference Learning, we do not need to wait for sampled return until the end of an episode. Instead we can update our "guess" of expected value function of the current state from the "guess" of the next state. In TD0 learning, we can infer/update our value function from the next state. The new

sampling value for the current state is just the immediate reward plus the decay coefficient multiplied by the next state's value. The value function for each state is initialized randomly first, and then these values should be converged. The obtained values are just the state function for the policy we evaluate. The pseudocode for the TD0 learning is shown in Figure 3.

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

Figure 3: Pseudocode for TD0 Learning

4 Experiment

We use Python 3 to simulate the grid world, and use both MC and TD0 learning to effectively evaluate the random policy, and get corresponding value function for each state.

We set decay coefficient λ to be 0.9 for both methods, and the learning step α to be 0.01 for TD0.

Simulation of first-visit, every-visit MC, and TD0 learning are shown in Figure 4 to 6.

```
[-4.412727096361509, 0, -5.649906287546152, -7.944295653394445, -8.700812657072783, -9.083994090115471]
[-6.315692760165945, -5.855853659442578, -7.347034831446297, -8.408449192446461, -8.850851639021347, -9.061060865253397]
[-7.951219558821805, -7.990922462669413, -8.48524431652137, -8.716249725729202, -8.78868407971259, -8.761780502597212]
[-8.928522683418594, -8.898058344335873, -8.918695361151071, -8.734043529416951, -8.292894445141943, -7.795116288312172 5]
[-9.301754004022998, -9.228424322430097, -8.844114330672301, -8.476895005567325, -7.458296364240865, -5.527672525745937]
[-9.407883023147797, -9.286799355252786, -8.954966673054072, -7.835914067356105, -5.747811500791754, 0]

Process finished with exit code 0
```

Figure 4: Result for First-Visit MC Learning

```
[-3.9855826274749, 0, -5.487161465885569, -7.722328263278037, -8.753507490131572, -9.142695439202742]
[-6.689766658536561, -6.0525285631776615, -7.347400586382574, -8.33214923610883, -8.902171888268906, -9.10060366024785]
[-8.1820707588519, -8.042039498682083, -8.471159419908501, -8.732000508206333, -8.831984531588143, -8.725523067297864]
[-8.894297628244933, -8.869207367847087, -8.857132476527116, -8.753549451364485, -8.36614440298096, -7.589189584044819]
[-9.222867829506775, -9.191514921040246, -8.971210865321105, -8.476077599758467, -7.300078944509767, -5.2799612144631265]
[-9.346415123520472, -9.279882534161409, -9.001798441304881, -7.969159079958789, -5.566994854487229, 0]

Process finished with exit code 0
```

Figure 5: Result for Every-Visit MC Learning

```
[-3.2049458888722113, 0, -4.686113394622342, -6.479329295364888, -7.1850225299068615, -7.388473911393375]
[-5.225812300334853, -4.66068696508589, -5.889283876810367, -6.766050474275485, -7.196502860977488, -7.324726204202623]
[-6.423897542414034, -6.444895419827005, -6.7941136876494035, -7.048209920812266, -7.091758726462811, -6.979919632773894]
[-7.18747880521663, -7.153633258042397, -7.16548979642474, -7.052258714115551, -6.672100613027883, -6.30338050902512]
[-7.535000894702866, -7.435435165112208, -7.246312713336426, -6.8561123943995, -5.853999347054884, -4.472865776565051]
[-7.656910935535482, -7.530036544730132, -7.1635974752925815, -6.395016395053766, -4.5056129135194185, 0]

Process finished with exit code 0
```

Figure 6: Result for TD0 Learning

5 Result Analysis

We have got the value functions for all the 36 states, for three different learning methods. Despite minor disparity, the relative value for all the grids are roughly the same. We can see that the value functions are large when close to the two terminal grids, and small when far away from these two grids. We can also try to improve the random policy after getting the value functions. The policy improvement direction is shown to be same as what we learn in last assignment.