Report for MC and TD

516021910212 Liuyuxi

April 2019

1 Introduction

I have done experiments with Monte-Carlo(MC) Learning and Temporal-Difference(TD) Learning to learn directly from episodes of experience without knowledge of MDP model. TD method can learn after every step, while MC method requires a full episode to update value evaluation. After implement MC and TD, I test them in my grid world to compare with the result in DP methods. You should implement both first-visit and every-visit MC method and TD(0) and to evaluate state value in small grid world. Both first-visit and every-visit MC method and TD(0) are tested. And the result is similar to those with DP methods.

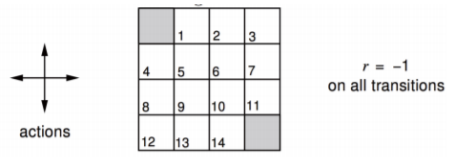

Figure 1: The gridworld

The agent is in an 4×4 gridworld, trying to get to the top left or bottom right corner as shown in 1. Each gird in the gridworld represents a certain state. S_t is the state at grid t. And the state space can be denoted as $S = \{s_t | t \in 0, ..., 15\}$. S_0 and S_{15} are terminal states, where $S_1$to $S_14$are non-terminal states. And the actions leading out of the grid leave state unchanged. Each movement get a reward of -1 until the terminal state is reached.

2 Algorithms

2.1 Setup

This is a model in the Reinforcement Learning literature. In this particular case:

- State space S: GridWorld has 4×4 distinct states, The start state is S_0 and the end state is S_{15}
- Actions A: The agent can choose from up to 4 actions to move around.
- Attenuation coefficient γ :1
- Testing strategy : Both MC and TD is used to evaluate the given strategy. In this project, I used random policy, which means that the agent has four directions to choose. Hence the testing action space is $A = \{N, E, S, W\}$.
- Methods: Monte-Carlo Policy Evaluation and Temporal-Difference Learning

Monte Carlo Intensive Learning refers to learning the state value directly from the complete episode without knowing the MDP state transition and instant reward. TD also learns from episode and does not need to understand the model itself; but it can learn the incomplete Episode, guessing the results of episode through its own bootstrapping, while continuously updating this guess.

2.2 First-Visit Monte-Carlo Policy Evaluation

The first time-step t that state s is visited in an episode.

- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Average return $V(s) = S(s)/N(s)$
- By law of large numbers $V(s) \rightarrow v_\pi(s) as N(s) \rightarrow \infty$

2.3 Every-Visit Monte-Carlo Policy Evaluation

Every time-step t that state s is visited in an episode.

- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Average return $V(s) = S(s)/N(s)$
- By law of large numbers $V(s) \rightarrow v_\pi(s) as N(s) \rightarrow \infty$

2.4 Temporal-Difference Learning

- TD methods learn directly from episodes of experience
- TD is model-free: no knowledge of MDP transitions / rewards
- TD learns from incomplete episodes, by bootstrapping
- TD updates a guess towards a guess
- $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

3 Code

3.1 First-Visit Monte-Carlo Policy Evaluation

Randomly produce episode which begins at random state and end at terminal state.

```
#produce a random episode which begins at random state
def line(actions):
    start=random.choice(states)
    current = start
    sample = []
    move = []
    act = random.choice(actions)
    sample.append(current)
    move.append(act)
    while isNotTerminateState(current):
        act = random.choice(actions)
        move.append(act)
        current = nextState(current, act)
        sample.append(current)
    return sample, move
```

Use First Visited MC to test 100000 episodes and calculate the result.

```
def FVMC(sample):
    occur = [0 for x in range(16)]
    lenth = len(sample)
    for index, s in enumerate(sample):
        if isTerminateState(s):
            continue
        if occur[s] == 0:  # first time appears
            times[s] = times[s] + 1
            occur[s] = 1
            gt = lenth - index - 1
            value[s] += gt

def main():
     for i in range(1000000):
        example, moves = line(actions)

        FVMC(example)
        # print(example)
    print(times)
    print(value)

    # printValue(value)
    result = [0 for x in range(16)]
    for i in range(1, 15):
        result[i] = -value[i] / times[i]
    printValue(result)

if __name__ == '__main__':
    main()
```

Print the result

```
def printValue(v):
    for i in range(16):
        print('{0:>6.2f}'.format(v[i]), end=" ")
        if (i + 1) % 4 == 0:
            print("")
    print()
```

3.2 Every-Visit Monte-Carlo Policy Evaluation

Every time-step t that state s is visited in an episode

```
def EVMC(sample):
    #occur = [0 for x in range(16)]
    lenth = len(sample)
    for index, s in enumerate(sample):
        if isTerminateState(s):
            continue
        times[s]+=1
        gt=lenth-index-1
        value[s]+=gt
def main():
    for i in range(1000000):
        example, moves = line(actions)
        EVMC(example)
        # print(example)
    print(times)
    print(value)
    result = [0 for x in range(16)]
    for i in range(1, 15):
        result[i] = -value[i] / times[i]
    printValue(result)
```

3.3 Temporal-Difference Learning

$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

First set the learning rate α.

```
alpha=0.01
```

```
def TD(sample):

    for index,s in enumerate(sample):
        if index==0:
            continue
        last=sample[index-1]#last state
        value[last] = value[last] + alpha * (-1 + value[s] -
                                             value[last])

def main():
    for i in range(1000000):
        example, moves = line(actions)
```

```
12
13          TD(example)
14          # print(example)
15      printValue(value)
```

4 Result

4.1 First-Visit Monte-Carlo Policy Evaluation

I have test the First-Visit Monte-Carlo for 100000 times, 1000000 times and compare the difference with DP models. The result is shown in Table 1.

Table 1: First-Visit Monte-Carlo Policy Evaluation

DP models			
0.00	-14.00	-20.00	-21.99
-14.00	-18.00	-20.00	-20.00
-20.00	-20.00	-18.00	-14.00
-21.99	-20.00	-14.00	0.00
100000 episodes			
0.00	-13.91	-19.93	-21.97
-14.00	-18.02	-19.95	-20.01
-20.07	-20.04	-18.08	-14.05
-22.10	-20.05	-14.07	0.00
1000000 episodes			
0.00	-14.03	-19.99	-21.99
-14.01	-17.99	-19.99	-20.01
-20.00	-19.96	-17.97	-13.99
-22.02	-19.98	-13.98	0.00

4.2 Every-Visit Monte-Carlo Policy Evaluation

The Every-Visit Monte-Carlo has been test for 100000 times, 1000000 times and I compare the results with DP models. The result is shown in Table 2.

4.3 Temporal-Difference Learning

The result of Temporal-Difference Learning is shown in Table 3. Seems that TD works worse than MC.

Table 2: Every-Visit Monte-Carlo Policy Evaluation

DP models			
0.00	-14.00	-20.00	-21.99
-14.00	-18.00	-20.00	-20.00
-20.00	-20.00	-18.00	-14.00
-21.99	-20.00	-14.00	0.00
100000 episodes			
0.00	-13.91	-19.89	-21.89
-13.97	-17.83	-19.85	-19.91
-19.84	-19.84	-17.80	-13.85
-21.78	-19.88	-13.76	0.00
1000000 episodes			
0.00	-13.97	-20.00	-21.97
-13.97	-17.99	-19.98	-19.96
-19.96	-19.99	-17.95	-13.97
-21.95	-19.99	-13.94	0.00

Table 3: Temporal-Difference Learning

DP models			
0.00	-14.00	-20.00	-21.99
-14.00	-18.00	-20.00	-20.00
-20.00	-20.00	-18.00	-14.00
-21.99	-20.00	-14.00	0.00
100000 episodes			
0.00	-14.00	-20.06	-21.81
-14.64	-17.98	-20.03	-19.68
-19.98	-19.91	-17.75	-13.44
-22.03	-20.26	-13.06	0.00
1000000 episodes			
0.00	-14.48	-20.59	-22.64
-13.38	-18.40	-20.46	-20.09
-20.28	-20.42	-18.33	-13.41
-22.43	-20.48	-14.65	0.00

5 Conclusion

Thanks to this project that I have deeper understanding of reinforcement learning. I also meet with some obstacles during the period. As for the result I get, the MC period is very close to the true possibility after testing for many times. And the more the testing times, the more accurate the result. However, the TD seems not perform as well as MC methods. I think it depends on the learning rate α of TD. This project has evoked my deep thinking and it demonstrates the power of MC and TD evaluation. Thanks for the guidance of the teacher and our teaching assistants.