

## 第四次作业 第七章

题目 1. 练习 7.2 在  $n$  步方法中, 价值函数需要每步都更新, 所以利用 TD 误差值和代替下述公式

$$V(S_t) \leftarrow V(S_t) + \alpha[G_{t:t+n} - V(S_t)], \quad 0 \leq t < T$$

中的错误项的算法将会与之前不同. 这种算法是一个更好的还是更差的算法? 请设计一个小实验并编程验证这个问题.

解答. 类似 MC 算法, 如果不考虑每步都进行更新, 则也可表示为 TD 误差值和; 在  $n$  步 TD 方法中, 如果不考虑每步都更新, 则可写为  $n$  个 TD 误差和

$$\begin{aligned}
 G_{t:t+n} - V(S_t) &= R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \\
 &\quad + \gamma(R_{t+2} + \cdots + \gamma^{n-2}R_{t+n} + \gamma^{n-1}V(S_{t+n}) - V(S_{t+1})) \\
 &= \delta_t + \gamma(R_{t+2} + \gamma V(S_{t+2}) - V(S_{t+1})) \\
 &\quad + \gamma^2(R_{t+3} + \cdots + \gamma^{n-3}R_{t+n} + \gamma^{n-2}V(S_{t+n}) - V(S_{t+2})) \\
 &= \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{n-2}\delta_{t+n-2} \\
 &\quad + \gamma^{n-1}(R_{t+n} + \gamma V(S_{t+n}) - V(S_{t+n-1})) + \gamma^n(V(S_{t+n}) - V(S_{t+n})) \\
 &= \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{n-1}\delta_{t+n-1} = \sum_{k=t}^{t+n-1} \gamma^{k-t}\delta_k
 \end{aligned}$$

所以该题的问题就是比较  $n$  步 TD 算法的实时更新和非实时更新 (每次一幕结束之后, 更新状态价值函数) 的好坏.

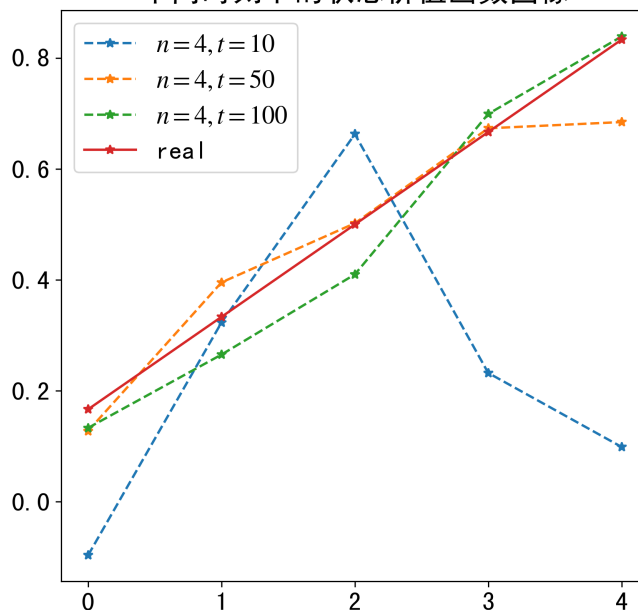
我使用的例子是书上例 6.2 随机游走:

A~E 的真实价值分别为  $\{\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\}$ , 使用实时更新 4 步 TD 算法状态价值函数 (Real time  $n$ -TD) 图像如右图所示.

我绘制了实时 TD 算法 (Real time) 和非实时 TD 算法 (non-Real time) 的均方误差和幕数的变换关系图, 容易看出, 实时 TD 算法的均方误差一直比非实时 TD 算法要小, 说明实时 TD 算法更优, 因为在实时更新算法中智能体学习过程中, 仍在不断更新状态价值函数, 从而可以更快地收敛到真实状态价值函数.

但二者差别并不是很大, 因为策略是固定的, 如果在策略迭代中, 实时算法收敛速度应该更加迅速.

不同时刻下的状态价值函数图像



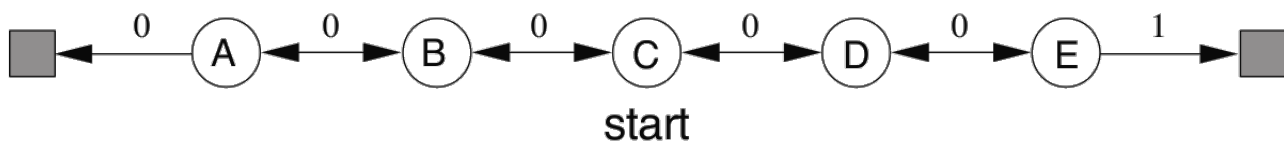
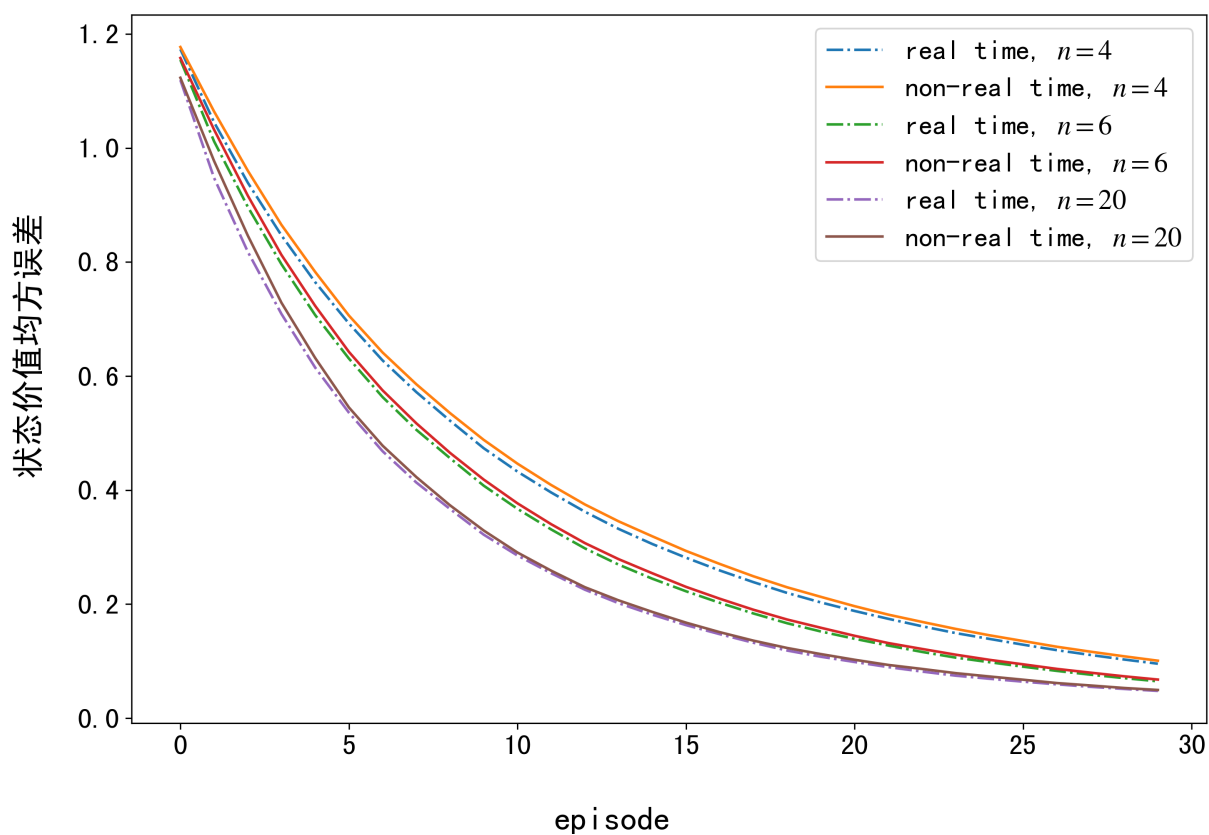


图 1: 随机游走

比较实时 (real time) 和非实时 (non-real time) 的  $n$ -TD 算法  
 状态价值均方误差变换效果  
 $\alpha = 0.05, \gamma = 1, episode = 30, epoch = 1000$



## 完整代码：

```
1 # -*- coding: utf-8 -*-
2 '''
3 @File      : main.py
4 @Time      : 2023/03/21 19:43:45
5 @Author    : wty-yy
6 @Version   : 1.0
7 @Blog      : https://wty-yy.space/
8 @Desc      : 《强化学习》中文书第 141 页练习 7.2, 比较 n-TD 算法的两种实现方法
9             1. 实时更新与书上写法一致。
10            2. 进行完一回合之后更新一次。
11            两种算法均对状态价值函数进行估计, 选的例子是第 123 页的“随机游走”,
12            随机游走例子中每个节点的真实状态价值函数都是已知的,
13            通过绘制“步数-价值函数的均方误差”的图像来比较两种算法。
14 '''
15
16 import numpy as np
17 import matplotlib.pyplot as plt
18 from tqdm import tqdm
19 from template.reinforcement import Environment, Agent
20 from pathlib import Path
21 PATH_FIGURES = Path(__file__).parent
22
23 class RandomWalk(Environment):
24     avail_points = 5
25     def __init__(self, avail_points=5) -> None:
26         super().__init__()
27         self.avail_points = avail_points
28         self.rewards = [0 for _ in range(self.avail_points + 1)]
29         self.rewards.append(1)
30         self.true_state_value = [0, *[(i+1)/(1+self.avail_points) for i in
31             ↪ range(self.avail_points)], 0]
32
33     def reset(self):
34         self.state = self.avail_points // 2 + 1
35         return self.state
36
37     def step(self, action):
38         self.state += action
39         reward = self.rewards[self.state]
40         terminal = 0 if self.state != 0 and self.state != self.avail_points + 1
41             ↪ else 1
42         return reward, self.state, terminal
43
44 class n_TD_Agent(Agent):
45     def __init__(self, n_TD=6, alpha=0.05, epsilon=0, gamma=1, seed=42,
46         ↪ episode=30, epoch=1000) -> None:
47         super().__init__(alpha, epsilon, gamma, seed, episode, epoch)
48         self.n_TD = n_TD
49
50     def start(self):
51         for real_time in [True, False]:
```

```

49     # for real_time in [True]:
50         self.history = np.zeros(self.episode)
51         for _ in tqdm(range(self.epoch)):
52             self.history += (self.train(real_time) - self.history) / (1 + _)
53         self.plot_MSE(label=f"{' ' if real_time else 'non-'}real time,
54             ↪ $n={self.n_TD}$", ls='-. ' if real_time else '-')
55
56 def train(self, real_time=True, verbose=False, **kwargs):
57     def choose_action():
58         return 1 if np.random.rand(1)[0] > 0.5 else -1
59
60     history = []
61     n = self.n_TD
62     state_value = np.random.normal(size=RandomWalk.avail_points + 2)
63     state_value[0] = state_value[-1] = 0
64
65     for _ in range(self.episode):
66         env = RandomWalk()
67         state = env.reset()
68         t, T = 0, np.inf
69         # rewards, states = ([0 for _ in range(n + 1)] for _ in range(2))
70         # states[0] = state
71         rewards, states = [0], []
72         states.append(state)
73
74     def update_state_value():
75         if tau >= 0:
76             target = 0
77             for i in range(tau + 1, min(tau + n, T) + 1):
78                 # target += np.power(self.gamma, i - tau - 1) * rewards[i]
79                 ↪ % (n+1)]
80                 target += np.power(self.gamma, i - tau - 1) * rewards[i]
81             if tau + n < T:
82                 # target += np.power(self.gamma, n) *
83                 ↪ state_value[states[(tau+n) % (n+1)]]
84                 target += np.power(self.gamma, n) *
85                 ↪ state_value[states[tau+n]]
86             # delta = target - state_value[states[tau % (n+1)]]
87             delta = target - state_value[states[tau]]
88             # state_value[states[tau % (n+1)]] += self.alpha * delta
89             state_value[states[tau]] += self.alpha * delta
90
91     while True:
92         if t < T:
93             action = choose_action()
94             reward, state, terminal = env.step(action)
95             rewards.append(reward)
96             states.append(state)
97             # rewards[(t+1) % (n+1)] = reward
98             # states[(t+1) % (n+1)] = state
99             if terminal:
100                 T = t + 1
101                 tau = t - n + 1

```

```

98         if real_time:
99             update_state_value()
100         if tau == T - 1:
101             break
102         t += 1
103     if not real_time:
104         for tau in range(T):
105             update_state_value()
106
107     mse = np.sum(np.power(state_value - env.true_state_value, 2)) /
108     ↪ (env.avail_points)
109     history.append(mse)
110     if verbose and (_+1) in kargs['verbose_time']:
111         plt.plot(state_value[1:-1], '--*', label=f"$n={n}, t={_+1}$")
112 if verbose:
113     plt.plot(env.true_state_value[1:-1], '-*', label="real")
114
115     return np.array(history)
116
117 def plot_MSE(self, label, ls):
118     plt.plot(self.history, label=label, ls=ls)
119
120 if __name__ == '__main__':
121     plt.title(" 不同时刻下的状态价值函数图像")
122     agent = n_TD_Agent(n_TD=4, alpha=0.05, episode=100)
123     agent.train(real_time=True, verbose=True, verbose_time=[10, 50, 100])
124     plt.legend()
125     plt.tight_layout()
126     plt.savefig(PATH_FIGURES.joinpath("state value function in diff time.png"),
127     ↪ dpi=300)
128     plt.show()
129
130 if __name__ == "__main__":
131     fig = plt.figure(figsize=(10, 8))
132     for subid, n in enumerate([4, 6, 20]):
133         agent = n_TD_Agent(n_TD=n)
134         agent.start()
135         plt.legend()
136     fig.suptitle(" 比较实时 (real time) 和非实时 (non-real time) 的 n-TD 算法\n状态
137     ↪ 价值均方误差变换效果\n")
138
139     ↪ +f"$\\alpha={agent.alpha}, \\gamma={agent.gamma}, episode={agent.episode}"
140     fig.supylabel(" 状态价值均方误差")
141     fig.supxlabel("episode")
142
143     plt.tight_layout()
144     plt.savefig(PATH_FIGURES.joinpath("compare (non)real time n-TD.png"),
145     ↪ dpi=300)
146     plt.show()

```