日期	科目	班级	姓名	学号
2023 年 3 月 5 日	强化学习	强基数学 002	吴天阳	2204210460

第二章作业

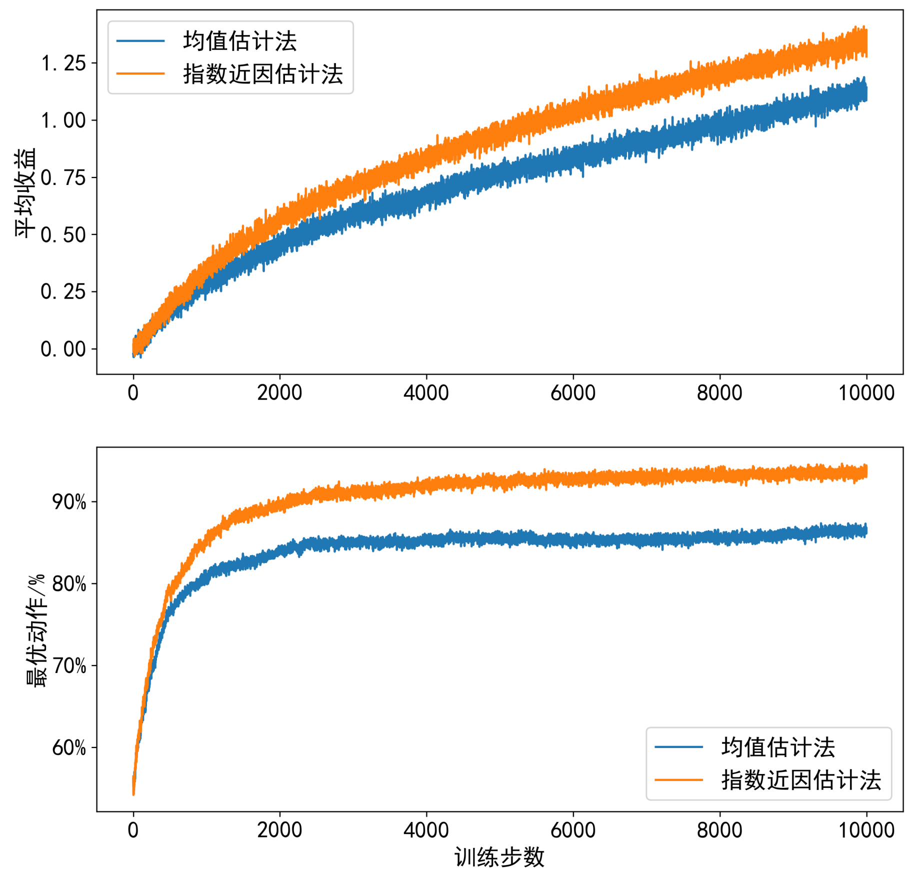
题目 1. 练习 2.2 考虑一个 $k=4$ 的多臂赌博机问题，记做 $1,2,3,4$. 使用 ε-贪心动作选择，基于均值的动作价值估计，初始估计 $Q_1(a)=0, \forall a=\{1,2,3,4\}$，假设动作及收益的序列为

$$A=\{1,2,2,2,3\},\ R=\{-1,1,-2,2,0\}$$

其中某些时刻中，可能发生了进行了探索（ε-时刻），即随机选择一个动作. 请问上述时间序列中，有哪些时刻肯定进行了探索？那些时刻可能进行了探索？

解答. 上述序列对应的每个时刻的动作价值 $Q_t(a)$ 变换如下表所示，从该表容易得出，第 $4,5$ 步的选择一定是进行了探索，而所有时刻都可能进行探索，因为在探索中，每个动作被选择到的概率是相同的.

t	1	2	3	4	5	6
$Q_t(1)$	0	-1	-1	-1	-1	-1
$Q_t(2)$	0	0	1	-0.5	1/3	1/3
$Q_t(3)$	0	0	0	0	0	0
$Q_t(4)$	0	0	0	0	0	0
A_t	1	2	2	2	3	



题目 2. 练习 2.5 设计实验来证实使用均值估计方法取解决非平稳问题的困难，使用一个 10 臂赌博机，其中所有的 $q_*(a)$ 初始时均相等，然后进行随机游走，每一步所有的 $q_*(a)$ 都加上一个服从 $N(0, 0.01^2)$ 的增量，分别使用均值估计方法和指数近因加权估计方法且步长 $\alpha = 0.1$ 进行决策，采用 ε-贪心进行动作选择，且总步数为 $T = 10000$.

解答. 如上图所示进行了 2000 次不同的多臂赌博机实验的平均结果，从中非常容易得出，指数近因估计在处理多臂赌博机问题上比均值估计要好.

```python
from tqdm import tqdm
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

config = {  # matplotlib 绘图配置
    "figure.figsize": (6, 6),  # 图像大小
    "font.size": 16, # 字号大小
    "font.sans-serif": ['SimHei'],   # 用黑体显示中文
    'axes.unicode_minus': False # 显示负号
}
plt.rcParams.update(config)

class Bandit:  # 赌博机类
    def __init__(self, n, init_means=0) -> None:
        # init_means: 赌博机初始均值
        self.n = n  # 赌博机臂个数
        self.means = np.full(self.n, init_means, dtype='float')  # 均值初始值

    def move_state(self):
        return np.random.normal(0, 0.01, self.n)  # 随机游走变换函数

    def step(self, action):  # 执行一个 action，并返回其收益
        assert(0 <= action < self.n)
        self.means += self.move_state()
        mean_now = self.means[action]
        rate = np.sum(self.means <= mean_now) / self.n
        return np.random.normal(self.means[action], size=1)[0], rate

def train(strategy, alpha=0.1):
    # 两种策略 average 和 recency
    bandit = Bandit(n)
    N = np.zeros(n)
    Q = np.zeros(n)
    history = [[], []]  # 存储每个时刻的收益和最优动作占比
    for t in range(T):
        if np.random.random(1)[0] < epsilon:  # 随机选取一个动作
            action = np.random.randint(0, n)
        else:  # 贪心选择最大价值的动作
            action = np.argmax(Q)
        reward, rate = bandit.step(action)
        delta = reward - Q[action]
        N[action] += 1
```

```
        Q[action] +=  delta / N[action] if strategy == 'average' else delta * alpha
        history[0].append(reward)
        history[1].append(rate)
    return np.array(history)

def plot_reward_rate(strategy, m=2000):  # m 为实验次数
    history = np.zeros((2, T))
    for _ in tqdm(range(m)):
        history += train(strategy=strategy)
    history /= m
    ax[0].plot(range(T), history[0], label='均值估计法' if strategy=='average' else '指数近因估计法')
    ax[1].plot(range(T), history[1], label='均值估计法' if strategy=='average' else '指数近因估计法')


if __name__ == '__main__':
    n, T = 10, 10000
    epsilon = 0.1
    np.random.seed(42)
    fig, ax = plt.subplots(2, 1, figsize=(10, 10))

    plot_reward_rate('average', m=2000)
    plot_reward_rate('recency', m=2000)

    ax[0].set_ylabel('平均收益')
    ax[1].set_ylabel('最优动作/%')
    ax[1].set_xlabel('训练步数')
    ax[1].yaxis.set_major_formatter(mpl.ticker.PercentFormatter(xmax=1, decimals=0))
    ax[0].legend()
    ax[1].legend()
    plt.savefig('33 页练习 2.5.png', dpi=300)
    plt.show()
```