МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Рубежный контроль №__2__
по дисциплине «Методы машинного обучения»

Тема: «Методы обучения с подкреплением»

ИСПОЛНИТЕЛЬ: _____
Лу Сяои
ФИО

группа ИУ5И-22М

_____
подпись

"5" Июнь 2023 г.

ПРЕПОДАВАТЕЛЬ: _____
ФИО

_____
подпись

"__"_____2023 г.

Москва - 2023

Тема: Методы обучения с подкреплением.

Для одного из алгоритмов временных различий, реализованных Вами в соответствующей лабораторная работе:

- SARSA
- Q-обучение
- Двойное Q-обучение

осуществите подбор гиперпараметров. Критерием оптимизации должна являться суммарная награда.

Среда, которую я использовала в этом эксперименте, была средой Cliff Walking. **Я выбрала алгоритм SARSA, который имеет следующие гиперпараметры: скорость обучения α, вероятность исследования epsilon, коэффициент дисконтирования gamma.**

Исходный код:

```
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm  # tqdm 是显示循环进度条的库


class CliffWalkingEnv:
    def __init__(self, ncol, nrow):
        self.nrow = nrow
        self.ncol = ncol
        self.x = 0  # 记录当前智能体位置的横坐标
        self.y = self.nrow - 1  # 记录当前智能体位置的纵坐标
```

```
    def step(self, action):  # 外部调用这个函数来改变当前位置
        # 4 种动作, change[0]:上, change[1]:下, change[2]:左, change[3]:右。坐标系原点(0,0)
        # 定义在左上角
        change = [[0, -1], [0, 1], [-1, 0], [1, 0]]
        self.x = min(self.ncol - 1, max(0, self.x + change[action][0]))
        self.y = min(self.nrow - 1, max(0, self.y + change[action][1]))
        next_state = self.y * self.ncol + self.x
        reward = -1
        done = False
        if self.y == self.nrow - 1 and self.x > 0:  # 下一个位置在悬崖或者目标
            done = True
            if self.x != self.ncol - 1:
                reward = -100
        return next_state, reward, done
```

```
    def reset(self):  # 回归初始状态,坐标轴原点在左上角
        self.x = 0
        self.y = self.nrow - 1
        return self.y * self.ncol + self.x
class Sarsa:
    """ Sarsa 算法 """
    def __init__(self, ncol, nrow, epsilon, alpha, gamma, n_action=4):
        self.Q_table = np.zeros([nrow * ncol, n_action])  # 初始化 Q(s,a)表格
        self.n_action = n_action  # 动作个数
        self.alpha = alpha  # 学习率
        self.gamma = gamma  # 折扣因子
        self.epsilon = epsilon  # epsilon-贪婪策略中的参数

    def take_action(self, state):  # 选取下一步的操作,具体实现为 epsilon-贪婪
        if np.random.random() < self.epsilon:
            action = np.random.randint(self.n_action)
```

```
        else:
            action = np.argmax(self.Q_table[state])
        return action
```

```
    def best_action(self, state):  # 用于打印策略
        Q_max = np.max(self.Q_table[state])
        a = [0 for _ in range(self.n_action)]
        for i in range(self.n_action):  # 若两个动作的价值一样,都会记录下来
            if self.Q_table[state, i] == Q_max:
                a[i] = 1
        return a
```

```
    def update(self, s0, a0, r, s1, a1):
        td_error = r + self.gamma * self.Q_table[s1, a1] - self.Q_table[s0, a0]
        self.Q_table[s0, a0] += self.alpha * td_error
```

```
ncol = 12
nrow = 4
env = CliffWalkingEnv(ncol, nrow)
np.random.seed(0)
epsilon = 0.1
alpha = 0.1
gamma = 0.9
agent = Sarsa(ncol, nrow, epsilon, alpha, gamma)
num_episodes = 500  # 智能体在环境中运行的序列的数量

return_list = []  # 记录每一条序列的回报
for i in range(10):  # 显示 10 个进度条
    # tqdm 的进度条功能
    with tqdm(total=int(num_episodes / 10), desc='Iteration %d' % i) as pbar:
        for i_episode in range(int(num_episodes / 10)):  # 每个进度条的序列数
            episode_return = 0
            state = env.reset()
            action = agent.take_action(state)
            done = False
            while not done:
                next_state, reward, done = env.step(action)
                next_action = agent.take_action(next_state)
                episode_return += reward  # 这里回报的计算不进行折扣因子衰减
                agent.update(state, action, reward, next_state, next_action)
                state = next_state
                action = next_action
            return_list.append(episode_return)
            if (i_episode + 1) % 10 == 0:  # 每 10 条序列打印一下这 10 条序列的平均回报
                pbar.set_postfix({
                    'episode':
                    '%d' % (num_episodes / 10 * i + i_episode + 1),
                    'return':
                    '%.3f' % np.mean(return_list[-10:])
                })
            pbar.update(1)
```

```
episodes_list = list(range(len(return_list)))
plt.plot(episodes_list, return_list)
plt.xlabel('Episodes')
plt.ylabel('Returns')
plt.title('Sarsa on {}'.format('Cliff Walking'))
plt.show()
```

```
def print_agent(agent, env, action_meaning, disaster=[], end=[]):
    for i in range(env.nrow):
        for j in range(env.ncol):
            if (i * env.ncol + j) in disaster:
                print('****', end=' ')
            elif (i * env.ncol + j) in end:
                print('EEEE', end=' ')
            else:
```

```
                a = agent.best_action(i * env.ncol + j)
                pi_str = ''
                for k in range(len(action_meaning)):
                    pi_str += action_meaning[k] if a[k] > 0 else 'o'
                print(pi_str, end=' ')
        print()

action_meaning = ['^', 'v', '<', '>']
print('The final convergence of Sarsa algorithm yields a strategy of')
print_agent(agent, env, action_meaning, list(range(37, 47)), [47])
```

Результат:

```
Iteration 0: 100%|██████████| 50/50 [00:00<00:00, 358.70it/s, episode=50, return=-119.400]
Iteration 1: 100%|██████████| 50/50 [00:00<00:00, 300.49it/s, episode=100, return=-63.000]
Iteration 2: 100%|██████████| 50/50 [00:00<00:00, 1047.64it/s, episode=150, return=-51.200]
Iteration 3: 100%|██████████| 50/50 [00:00<00:00, 1053.02it/s, episode=200, return=-48.100]
Iteration 4: 100%|██████████| 50/50 [00:00<00:00, 1211.57it/s, episode=250, return=-35.700]
Iteration 5: 100%|██████████| 50/50 [00:00<00:00, 1065.81it/s, episode=300, return=-29.900]
Iteration 6: 100%|██████████| 50/50 [00:00<00:00, 1203.40it/s, episode=350, return=-28.300]
Iteration 7: 100%|██████████| 50/50 [00:00<00:00, 1248.19it/s, episode=400, return=-27.700]
Iteration 8: 100%|██████████| 50/50 [00:00<00:00, 1075.68it/s, episode=450, return=-28.500]
Iteration 9: 100%|██████████| 50/50 [00:00<00:00, 1717.75it/s, episode=500, return=-18.900]
```


```
The final convergence of Sarsa algorithm yields a strategy of
ooo> ooo> ooo> ooo> ooo> ooo> ooo> ooo> ooo> ooo> ooo> ovoo
^ooo ooo> ^ooo ooo> ooo> ooo> ooo> ooo> ooo> ^ooo ooo> ovoo
^ooo ^ooo ^ooo ooo> ooo> ^ooo ^ooo oo<o ooo> ooo> ooo> ovoo
^ooo **** **** **** **** **** **** **** **** **** **** EEEE
```

Выбор гиперпараметров:

Критерием оптимизации должна являться суммарная награда.[return_list]
скорость обучения α, вероятность исследования epsilon, коэффициент дисконтирования gamma.
epsilon、alpha、gamma.[0.1,0.1,0.9]

```
num_episodes = 500
```

Для сравнения добавлен код для вычисления максимального значения и стандартного отклонения

```
max_return = np.max(return_list)
std_return = np.std(return_list)

print("Max return: {}".format(max_return))
print("Std of return: {}".format(std_return))
```

```
Max return: -15
Std of return: 44.573294740236555
```

Эпсилон может искать с шагом 0,05 между 0,05 и 0,2,
альфа может искать с шагом 0,1 между 0,1 и 0,5,
а гамма может искать с шагом 0,1 между 0,7 и 0,99.

Epsilon

0.2	Max return: -17 Std of return: 46.23069473845272
0.15	Max return: -15 Std of return: 46.356447275433005
0.1	Max return: -15 Std of return: 44.573294740236555
0.05	Max return: -15 Std of return: 43.223925966991935

0.05

Alpha

0.1	Max return: -15 Std of return: 43.223925966991935
0.2	Max return: -15 Std of return: 34.7556210705549
0.3	Max return: -15 Std of return: 28.89919382958632
0.4	Max return: -17 Std of return: 29.654637141600634
0.5	Max return: -15 Std of return: 308.0353770007594

0.3

Gamma

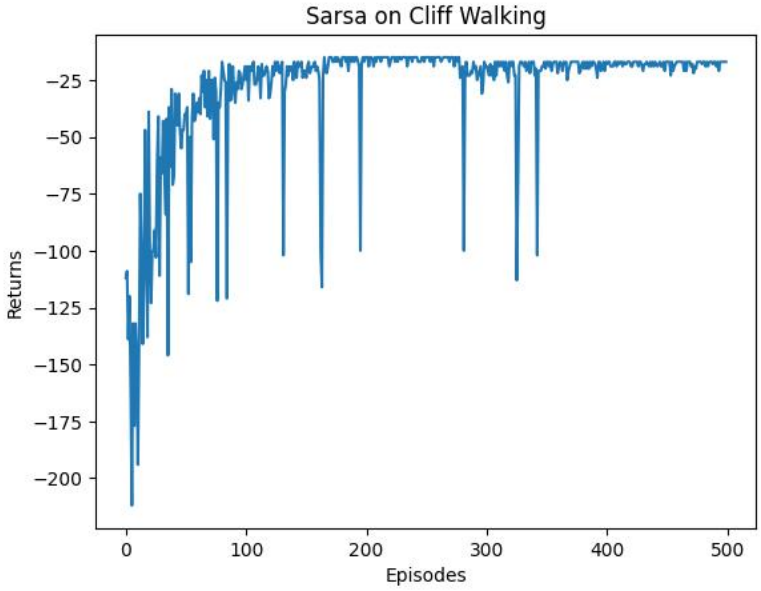
0.99	Max return: -15 Std of return: 30.267046833148424
0.95	Max return: -15 Std of return: 30.774183985932105
0.9	Max return: -15 Std of return: 28.89919382958632

0.85	Max return: -15 Std of return: 32.27787130527662
0.8	Max return: -15 Std of return: 33.56611124333589
0.7	Max return: -17 Std of return: 43.54758220613402

0.9

——Оптимальные гиперпараметры: epsilon、alpha、gamma.[0.05,0.3,0.9]

```
Iteration 0: 100%|██████████| 50/50 [00:00<00:00, 991.69it/s, episode=50, return=-45.000]
Iteration 1: 100%|██████████| 50/50 [00:00<00:00, 1751.67it/s, episode=100, return=-26.200]
Iteration 2: 100%|██████████| 50/50 [00:00<00:00, 2243.59it/s, episode=150, return=-20.000]
Iteration 3: 100%|██████████| 50/50 [00:00<00:00, 1918.80it/s, episode=200, return=-24.800]
Iteration 4: 100%|██████████| 50/50 [00:00<00:00, 2142.51it/s, episode=250, return=-15.800]
Iteration 5: 100%|██████████| 50/50 [00:00<00:00, 2241.82it/s, episode=300, return=-22.100]
Iteration 6: 100%|██████████| 50/50 [00:00<00:00, 1366.82it/s, episode=350, return=-28.300]
Iteration 7: 100%|██████████| 50/50 [00:00<00:00, 1851.10it/s, episode=400, return=-19.100]
Iteration 8: 100%|██████████| 50/50 [00:00<00:00, 2194.73it/s, episode=450, return=-18.200]
Iteration 9: 100%|██████████| 50/50 [00:00<00:00, 1843.25it/s, episode=500, return=-17.600]
```




```
The final convergence of Sarsa algorithm yields a strategy of
ooo> ooo> ooo> ooo> ooo> ooo> ooo> ooo> ooo> ooo> ooo> ovoo
^ooo ooo> ^ooo ooo> ooo> ooo> ooo> ooo> ooo> ^ooo ooo> ovoo
^ooo ^ooo ^ooo ooo> ooo> ^ooo ^ooo oo<o ooo> ooo> ooo> ovoo
^ooo **** **** **** **** **** **** **** **** **** **** EEEE
```

В моем коде, после небольшой оптимизации гиперпараметров, алгоритм Sarsa дает стратегию, которая оставляет действия, предпринимаемые интеллектом, неизменными. Алгоритм Sarsa примет стратегию удаления от обрыва, чтобы достичь цели.