МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

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

ОТЧЕТ

Лабораторная работа № 5
по дисциплине «Методы машинного обучения»

Тема: «Обработка признаков (часть 2).»

ИСПОЛНИТЕЛЬ: Ли Яцзинь
ФИО
группа ИУ5И-21М
подпись"

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю .Е

Москва - 2024

Задание:

На основе рассмотренного на лекции примера реализуйте следующие алгоритмы:

- SARSA
- Q-обучение
- Двойное Q-обучение

для любой среды обучения с подкреплением (кроме рассмотренной на лекции среды Toy Text / Frozen Lake) из библиотеки Gym (или аналогичной библиотеки)

SARSA

Создайте код для реализации алгоритма SARS (состояние-действие-вознаграждение-следующее состояние-действие) для решения простой задачи в лабиринте.

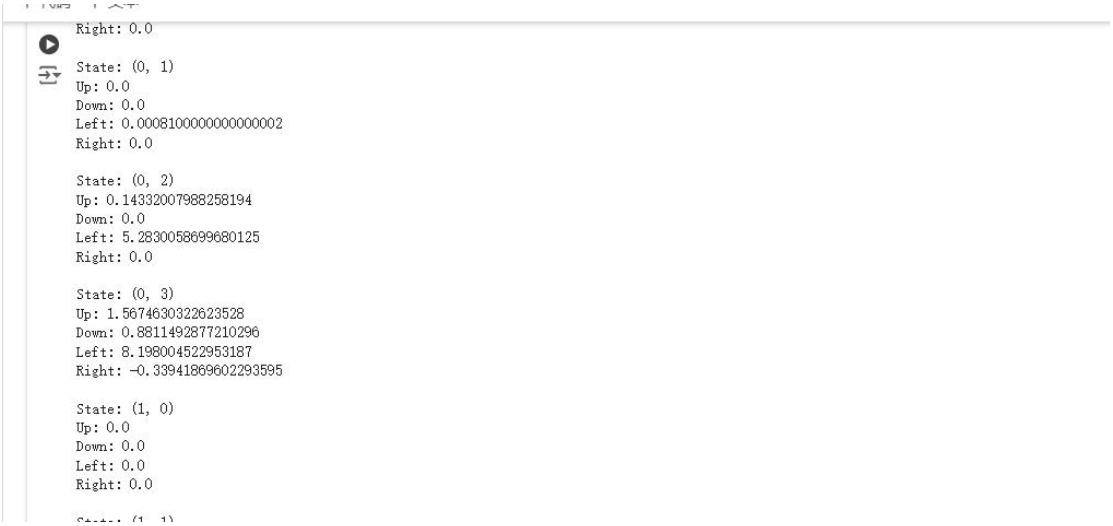
```
# SARSA算法
for episode in range(max_episodes):
    state = start_state
    action = np.random.choice(range(4)) if np.random.rand() < epsilon else np.argmax(Q[state])

    while state != goal_state:
        # next_state = (state[0] + actions[action][0], state[1] + actions[action][1])
        a = state[0] + actions[action][0]
        b = state[1] + actions[action][1]
        if a > 3:
            a-=1
        elif b > 3:
            b-=1
        elif a < -4:
            a+= 1
        elif b < -4:
            b+= 1
        next_state = (a,b)
        reward = maze[next_state]
        next_action = np.random.choice(range(4)) if np.random.rand() < epsilon else np.argmax(Q[next_state])
        Q[state][action] += alpha * (reward + gamma * Q[next_state][next_action] - Q[state][action])

        state = next_state
        action = next_action

# 输出结果
for i in range(4):
    for j in range(4):
        print("State:", (i, j))
        print("Up:", Q[i][j][0])
        print("Down:", Q[i][j][1])
        print("Left:", Q[i][j][2])
        print("Right:", Q[i][j][3])
```

Рисунок 1. Код результаты SARSA



```
Right: 0.0

State: (0, 1)
Up: 0.0
Down: 0.0
Left: 0.0008100000000000002
Right: 0.0

State: (0, 2)
Up: 0.14332007988258194
Down: 0.0
Left: 5.2830058699680125
Right: 0.0

State: (0, 3)
Up: 1.5674630322623528
Down: 0.8811492877210296
Left: 8.198004522953187
Right: -0.33941869602293595

State: (1, 0)
Up: 0.0
Down: 0.0
Left: 0.0
Right: 0.0
```

Рисунок 2. результаты SARSA

Q-Обучение

Создайте код для реализации алгоритма Q-обучения. В этом коде мы предполагаем простую среду сетки, в которой агент

учится находить оптимальную политику с помощью алгоритма Q-обучения.

```
+ 代码  + 文本

# 运行Q-learning算法进行训练
num_episodes = 1000
for episode in range(num_episodes):
    state = np.random.randint(num_states)  # 随机选择一个起始状态
    done = False
    while not done:
        action = agent.choose_action(state)  # 根据当前策略选择动作
        if action == 0:  # 上
            next_state = max(0, state - 1)
        elif action == 1:  # 下
            next_state = min(num_states - 1, state + 1)
        elif action == 2:  # 左
            next_state = max(0, state - 1)
        else:  # 右
            next_state = min(num_states - 1, state + 1)

        reward = 0
        if next_state == num_states - 1:  # 到达目标位置，获得+1的奖励
            reward = 1
            done = True

        # 更新Q值函数
        agent.learn(state, action, reward, next_state)

        state = next_state  # 更新状态

# 打印学习到的Q值函数
print("Learned Q-values:")
print(agent.Q)

Learned Q-values:
[[1.43659092 0.36577292 1.59253891 3.64937476]
 [1.54432707 4.08513067 1.51843521 2.40630573]
 [2.78177559 3.66249678 2.27370049 4.55156673]
 [3.284947   3.98496246 3.16216149 5.06385588]
 [4.51892753 1.17672809 1.12473567 1.73250984]]

✓ 1秒  完成时间：13:44
```

Рисунок 3. Код Z-оценки и результаты Q-Обучение

Двойное Q-обучение

```
# 运行Double Q-learning算法进行训练
num_episodes = 1000
for episode in range(num_episodes):
    state = np.random.randint(num_states)  # 随机选择一个起始状态
    done = False
    while not done:
        action = agent.choose_action(state)  # 根据当前策略选择动作
        if action == 0:  # 上
            next_state = max(0, state - 1)
        elif action == 1:  # 下
            next_state = min(num_states - 1, state + 1)
        elif action == 2:  # 左
            next_state = max(0, state - 1)
        else:  # 右
            next_state = min(num_states - 1, state + 1)

        reward = 0
        if next_state == num_states - 1:  # 到达目标位置，获得+1的奖励
            reward = 1
            done = True

        # 更新Q值函数
        agent.learn(state, action, reward, next_state)

        state = next_state  # 更新状态

# 打印学习到的Q值函数
print("Learned Q-values:")
print("Q1:")
```

Рисунок 4-Код Двойное Q-обучение

```
# 打印学习到的Q值函数
print("Learned Q-values:")
print("Q1:")
print(agent.Q1)
print("Q2:")
print(agent.Q2)
```

```
Learned Q-values:
Q1:
[[0.43499351 0.         0.32911716 3.0301276 ]
 [1.48585265 1.55812233 0.82279627 3.46324513]
 [1.07178318 3.91453899 1.43966066 1.96652781]
 [1.22365307 3.24724382 1.86128169 4.37919984]
 [3.79371819 0.57276659 0.61031959 0.59604726]]
Q2:
[[0.11177675 0.19437398 0.         2.95097101]
 [0.51674766 0.68537353 0.74017839 3.46560802]
 [0.9801829  3.89999628 1.67943174 0.86921794]
 [1.75241339 2.73605752 2.14560506 4.40124908]
 [3.81455617 0.9422444  0.2509109  0.        ]]
```

Рисунок 5- результаты Двойное Q-обучение

Этот код реализует алгоритм двойного Q-обучения, в котором агент использует две функции Q-значения (Q1 и Q2) для изучения политик в среде. При каждом обновлении агент выбирает, какую функцию Q-значения обновить с вероятностью 0,5. Наконец, агент выбирает действия и оптимизирует стратегию, изучая две функции Q-значения.