Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Методы машинного обучения»

Отчет по домашнему заданию

Выполнила:
студент группы ИУ5-22М
Павловская А.А.
23.05.2023

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2023 г.

Цель работы: ознакомление с методом обучения с подкреплением на основе временных различий Double SARSA.

Задание:

Реализовать алгоритм Double SARSA для среды обучения с подкрепления CliffWalking из библиотеки Gym

Ход работы

В среде CliffWalking агент может находиться в 48 состояниях и осуществлять 4 действия. Одно состояние начальное, одно конечное, а также есть «обрыв» из 10 клеток внизу карты (рис.1).

1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48

Рис.1. Карта состояний

Действия: 1 – вверх, 2 – вправо, 3 – вниз, 4 – влево.

Вознаграждение равно –1 для всех переходов, кроме ведущих в «обрыв». За вход в эту область начисляется вознаграждение –100, после чего агент мгновенно переносится в начальную позицию [2].

В алгоритме Double SARSA, как и в Double Q-learning, используется эпсилон-жадная стратегия, определяющая действие на основе среднего по сумме значений двух таблиц Q^1 и Q^2.

Обновление происходит по правилу [1]:

$$Q^1_{t+1}(s,a) \leftarrow Q^1_t(s,a) + \alpha[r + \gamma Q^2_t(s',a') - Q^1_t(s,a)]$$

Алгоритм Double SARSA отличается от SARSA наличием дополнительной таблицы, а от Double Q-learning тем, что в конце Q^1 и Q^2 не обновляются с вероятностью 0.5, а обновляется лишь одна таблица, а затем с вероятностью 0.5 они меняются местами.

Алгоритм 1. Double SARSA

1. Инициализировать $Q^1(s,a)$ и $Q^2(s,a)$ произвольным образом
2. **Повторять** для каждого эпизода:
3. Инициализировать s
4. Выбрать a в состоянии s, следуя произвольной стратегии
5. **Повторять** для каждого шага эпизода:
6. Предпринять действие, наблюдать r, s’
7. Выбрать a’ из состояния s’, следуя ε-жадной стратегии относительно среднего $Q^1 + Q^2$
8. $$Q^1(s,a) \leftarrow Q^1(s,a) + \alpha[r + \gamma Q^2(s',a') - Q^1(s,a)]$$

9. $s \leftarrow s'; a \leftarrow a';$

10. **С вероятностью** 0.5:

11. Поменять местами Q^1 и Q^2

Текст программы

```
import numpy as np
import matplotlib.pyplot as plt
import gym
from tqdm import tqdm

# ********************************** БАЗОВЫЙ АГЕНТ
**************************************

class BasicAgent:
    '''
    Базовый агент, от которого наследуются стратегии обучения
    '''

    # Наименование алгоритма
    ALGO_NAME = '---'

    def __init__(self, env, eps=0.1):
        # Среда
        self.env = env
        # Размерности Q-матрицы
        self.nA = env.action_space.n
        self.nS = env.observation_space.n
        #и сама матрица
        self.Q = np.zeros((self.nS, self.nA))
        # Значения коэффициентов
        # Порог выбора случайного действия
        self.eps=eps
        # Награды по эпизодам
        self.episodes_reward = []


    def print_q(self):
        print('Вывод Q-матрицы для алгоритма ', self.ALGO_NAME)
        print(self.Q)


    def get_state(self, state):
        '''
        Возвращает правильное начальное состояние
        '''
        if type(state) is tuple:
            # Если состояние вернулось с виде кортежа, то вернуть только номер
состояния
            return state[0]
```

```
        else:
            return state


    def greedy(self, state):
        '''
        <<Жадное>> текущее действие
        Возвращает действие, соответствующее максимальному Q-значению
        для состояния state
        '''
        return np.argmax(self.Q[state])


    def make_action(self, state):
        '''
        Выбор действия агентом
        '''
        if np.random.uniform(0,1) < self.eps:

            # Если вероятность меньше eps
            # то выбирается случайное действие
            return self.env.action_space.sample()
        else:
            # иначе действие, соответствующее максимальному Q-значению
            return self.greedy(state)


    def draw_episodes_reward(self):
        # Построение графика наград по эпизодам
        fig, ax = plt.subplots(figsize = (15,10))
        y = self.episodes_reward
        x = list(range(1, len(y)+1))
        plt.plot(x, y, '-', linewidth=1, color='green')
        plt.title('Награды по эпизодам')
        plt.xlabel('Номер эпизода')
        plt.ylabel('Награда')
        plt.show()


    def learn():
        '''
        Реализация алгоритма обучения
        '''
        pass

# ********************************** Double SARSA
**************************************

class DoubleSARSA_Agent(BasicAgent):
    '''
    Реализация алгоритма Double SARSA
```

```
    '''
    # Наименование алгоритма
    ALGO_NAME = 'Double SARSA'


    def __init__(self, env, eps=0.015, lr=0.1, gamma=0.97, num_episodes=20000):
#self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000
        # Вызов конструктора верхнего уровня
        super().__init__(env, eps)
        # Вторая матрица
        self.Q2 = np.zeros((self.nS, self.nA))
        # Learning rate
        self.lr=lr
        # Коэффициент дисконтирования
        self.gamma = gamma
        # Количество эпизодов
        self.num_episodes=num_episodes
        # Постепенное уменьшение eps
        self.eps_decay=0.00005
        self.eps_threshold=0.01


    def greedy(self, state):
        '''
        <<Жадное>> текущее действие
        Возвращает действие, соответствующее максимальному Q-значению
        для состояния state
        '''
        temp_q = self.Q[state] + self.Q2[state]
        return np.argmax(temp_q)


    def print_q(self):
        print('Вывод Q-матриц для алгоритма ', self.ALGO_NAME)
        print('Q1')
        print(self.Q)
        print('Q2')
        print(self.Q2)


    def learn(self):
        '''
        Обучение на основе алгоритма Double SARSA
        '''
        self.episodes_reward = []
        # Цикл по эпизодам
        for ep in tqdm(list(range(self.num_episodes))):
            # Начальное состояние среды
            state = self.get_state(self.env.reset())
            # Флаг штатного завершения эпизода
            done = False
```

```
            # Флаг нештатного завершения эпизода
            truncated = False
            # Суммарная награда по эпизоду
            tot_rew = 0

            # По мере заполнения Q-матрицы уменьшаем вероятность случайного
выбора действия
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay
            # Выбор действия

            action = self.make_action(state)

            # Проигрывание одного эпизода до финального состояния
            while not (done or truncated):

                # Выполняем шаг в среде
                next_state, rew, done, truncated, _ = self.env.step(action)

                # Выполняем следующее действие
                next_action = self.make_action(next_state)

                self.Q[state][action] = self.Q[state][action] + self.lr * \
                (rew + self.gamma * self.Q2[next_state][next_action] -
self.Q[state][action])

                # Следующее состояние считаем текущим
                state = next_state
                action = next_action
                if np.random.rand() < 0.5:
                    swap_list = self.Q
                    self.Q = self.Q2
                    self.Q2 = swap_list
                # Суммарная награда за эпизод
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)
    def sum_rewards(self):
        # Суммарная награда
        sum_rewards = sum(self.episodes_reward)
        print('Суммарная награда Double SARSA: ', sum_rewards)

def play_agent(agent):
    '''
    Проигрывание сессии для обученного агента
    '''
    env2 = gym.make('CliffWalking-v0', render_mode='human') #FrozenLake-v1
    state = env2.reset()[0]
    done = False
    while not done:
        action = agent.greedy(state)
```

```
            next_state, reward, terminated, truncated, _ = env2.step(action)
            env2.render()
            state = next_state
            if terminated or truncated:
                done = True

def run_double_sarsa():
    env = gym.make('CliffWalking-v0')
    agent = DoubleSARSA_Agent(env)
    agent.learn()
    agent.sum_rewards()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)



def main():
    run_double_sarsa()


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

Результаты выполнения программы

Обучение проводилось со следующими гиперпараметрами: eps=0.015, lr=0.1, gamma=0.97, num_episodes=20000. Награды по эпизодам представлены на рис.1. Суммарная награда -333627.

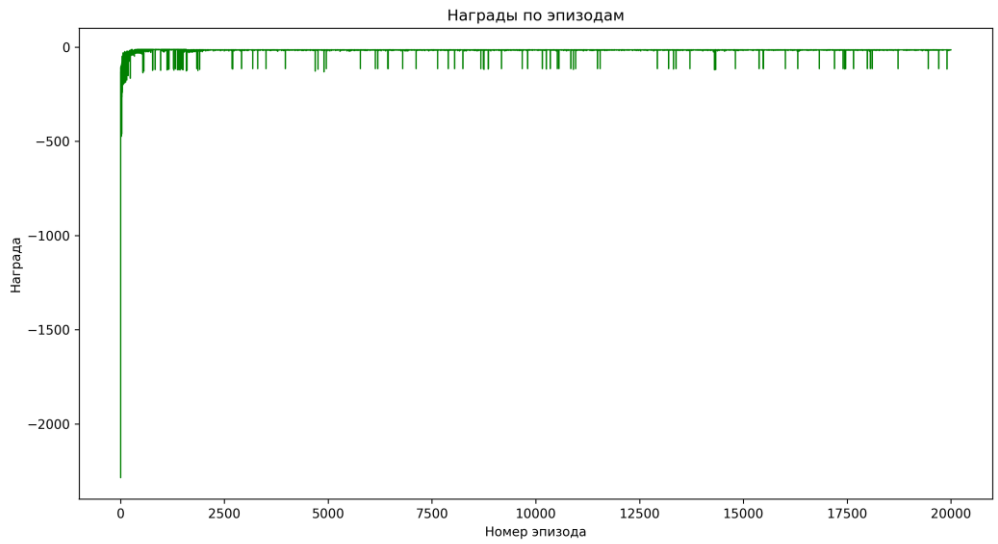

Рис.1. Награды по эпизодам

Q-матрицы для алгоритма Double SARSA представлены на рис.2 – рис.3.

```
Вывод Q-матриц для алгоритма  Double SARSA
Q1
[[ -11.080563    -11.1381556   -11.05454038  -11.11041249]
 [ -10.47925392  -10.47303154  -10.64778514  -10.28036108]
 [  -9.95960502  -10.01369474  -10.00166319  -10.40600962]
 [  -9.46017341   -9.33493124   -9.29599732   -9.222075  ]
 [  -8.66843163   -8.67558988   -8.73119932   -8.81884671]
 [  -8.0507983    -7.94314158   -8.01122188   -8.03456083]
 [  -7.25189287   -7.18646638   -7.23522527   -7.52513129]
 [  -6.39291067   -6.4007632    -6.37242727   -6.39648028]
 [  -5.72714348   -5.5806194    -5.57370527   -5.84415855]
 [  -4.88478849   -4.7321794    -4.78412757   -4.78375171]
 [  -4.04469185   -3.84313136   -3.83868257   -4.04988292]
 [  -3.2558845    -3.08177772   -2.94222421   -3.08306041]
 [ -11.52352827  -10.96718729  -12.21571314  -11.58529952]
 [ -11.02328942  -10.23906784  -11.46116045  -11.1337513 ]
 [ -10.47181758   -9.52623657  -10.80006759  -10.91843436]
 [  -9.84971543   -8.81715302  -10.20984914  -10.14858394]
 [  -9.28753044   -8.01497063   -8.70087502   -9.52272554]
 [  -8.47847094   -7.23689253   -8.13150605   -8.7754626 ]
 [  -7.83615369   -6.40330597   -7.58790841   -7.96359037]
 [  -7.0807738    -5.57996225   -6.90912291   -7.05497317]
 [  -6.08337408   -4.70887933   -5.35020335   -6.24810428]
 [  -5.1767316    -3.823573     -4.64947883   -5.47112069]
 [  -4.49918143   -2.91090003   -2.91240345   -4.12286757]
 [  -3.78768089   -2.8438351    -1.97         -3.8103546 ]
 [ -11.60510063  -11.69867551  -12.92267719  -12.34220934]
 [ -10.99244834  -13.27354085  -56.4972935   -11.31808978]
 [ -10.40577568  -10.25785105  -29.79668826  -10.39097666]
 [  -9.55428046   -8.32217054  -10.36718708   -9.76840674]
 [  -8.32342187   -8.18982129  -36.6882572    -8.31505523]
 [  -7.7826398    -7.73934877  -19.3874094    -7.81201746]
 [  -7.07119194   -7.03589959  -29.98935449   -6.92214375]
 [  -6.33352046   -5.15377218  -10.08334664   -6.31086741]
 [  -5.5715785    -4.12854192  -10.55995861   -5.59587035]
 [  -4.68666798   -2.91090761  -11.06479844   -4.65884721]
 [  -2.6907713    -1.98120476  -37.08031157   -3.30007157]
 [  -2.87623003   -1.85877447   -1.           -2.73965621]
 [ -12.33717635 -110.41566698  -12.88609681  -12.80314886]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]]
```

Рис.2. Матрица Q1

```
Q2
[[ -11.08586903  -11.0187356   -11.14236039  -11.11051837]
 [ -10.70289976  -10.62908604  -10.46303112  -10.84127892]
 [ -10.10155398   -9.9538629    -9.96169557   -9.60945485]
 [  -9.27384878   -9.37020332   -9.40103606   -9.52853684]
 [  -8.71519936   -8.6741153    -8.62242224   -8.5556677 ]
 [  -7.87014519   -7.9641878    -7.90259358   -7.99138534]
 [  -7.14390103   -7.19562782   -7.14451599   -6.95680005]
 [  -6.4709336    -6.3897553    -6.41576745   -6.44430163]
 [  -5.53512624   -5.57402064   -5.5855124    -5.36419096]
 [  -4.62329965   -4.72631      -4.71213949   -4.80604929]
 [  -3.8850537    -3.85661287   -3.8475419    -3.78756401]
 [  -2.82034339   -3.1089602    -2.9325495    -3.20182789]
 [ -11.57765262  -10.92921705  -12.25243694  -11.60152393]
 [ -10.93920533  -10.24774153  -12.61630665  -11.53691485]
 [ -10.32578674   -9.53816933  -10.85207677  -11.01607786]
 [  -9.65844603   -8.78313115  -10.12349265  -10.23599838]
 [  -9.19613449   -8.054812     -8.70440541   -9.52274081]
 [  -8.49558505   -7.22124816   -8.29493558   -8.70293537]
 [  -7.85228374   -6.42426042   -7.72855516   -8.05395695]
 [  -6.91447624   -5.56830521   -6.40064595   -7.12232534]
 [  -6.24831343   -4.7117238    -6.13849229   -6.199037  ]
 [  -5.40076311   -3.82357485   -4.13965141   -5.37722821]
 [  -4.52676965   -2.9109       -3.32840886   -4.62469757]
 [  -3.63074626   -2.73355204   -1.97         -3.80259484]
 [ -11.66898222  -14.87818947  -12.92617068  -12.21039056]
 [ -11.02129774  -10.96765214  -20.41955606  -11.13474743]
 [ -10.26796194  -10.27859575  -36.62288387  -10.39022051]
 [  -9.55408516  -10.88267187  -60.89680258   -9.65761756]
 [  -8.14942466   -8.29769486  -21.07054352   -8.3409971 ]
 [  -7.73011679   -7.73959607  -21.0573616    -7.77656998]
 [  -7.17589919   -6.64666656  -28.57844014   -7.3428351 ]
 [  -6.37517613   -5.64057452  -29.37401332   -6.50345661]
 [  -5.70300463   -3.92624885  -28.14777497   -5.75110091]
 [  -4.60536207   -3.0152848   -44.163656     -4.7573137 ]
 [  -2.30227688   -1.97000003  -37.1949361    -2.3720749 ]
 [  -2.84549939   -1.90216026   -1.           -2.88136095]
 [ -12.26284655 -106.97945634  -12.9130412   -12.81776506]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]]
```

Рис.3. Матрица Q2

По матрицам видно, что для состояний 38 – 48 награды 0 (эти состояния соответствуют «обрыву» и конечному состоянию. Максимальное вознаграждение из 36-го

состояния (клетка над конечным состоянием) соответствует исходному -1 за действие «вниз».

Путь обученного агента представлен на рис.4. При значении eps=0.4 путь проходит по максимально «безопасной» траектории, то есть через максимально удаленные клетки от «обрыва». Однако в данном примере eps=0.015, и стратегия дальше приближается к оптимальной при дальнейшем уменьшении этого параметра.

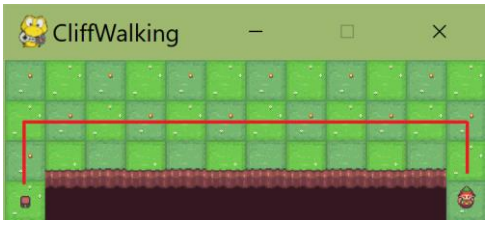

Рис.4. Путь обученного агента

Вывод: в ходе выполнения домашнего задания я ознакомилась с методом обучения с подкреплением на основе временных различий Double SARSA.

Список использованных источников

1. Ganger, M., Du- ryea, E. and Hu, W. (2016) Double Sarsa and Double Expected Sarsa with Shallow and Deep Learning. Journal of Data Ana- lysis and Information Processing, 4, 159-176. http://dx.doi.org/10.4236/jdaip.2016.44014

2. Саттон Р. С., Барто Э. Дж. Обучение с подкреплением: Введение. 2-е изд. / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 552 с.