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

ФАКУЛЬТЕТ	**«Информатики и систем управления»**
КАФЕДРА	Системы обработки информации и управления

Дисциплина «Методы машинного обучения»

РУБЕЖНЫЙ КОНТРОЛЬ №2

Студент	Сахарова Е. К. ИУ5-21М
Преподаватель	Гапанюк Ю. Е.

Задание

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.

Выполнение

Для выполнения реализуем алгоритм Q-обучение на примере такси.

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

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

all_reward=[]
parameter=[]

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 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

# ********************************** Q-обучение
**************************************

class QLearning_Agent(BasicAgent):
    '''
    Реализация алгоритма Q-Learning
    '''
    # Наименование алгоритма
    ALGO_NAME = 'Q-обучение'


    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=100):
        # Вызов конструктора верхнего уровня
        super().__init__(env, eps)
        # 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 print_q(self):
        all_reward.append(np.sum(self.Q))
        print('Суммарная награда:',np.sum(self.Q), f"lr = {self.lr:.3f} gamma =
{self.gamma:.3f} eps = {self.eps:.3f}")

    def learn(self):
        '''
        Обучение на основе алгоритма Q-Learning
        '''
        self.episodes_reward = []
        # Цикл по эпизодам
        for ep in 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
```

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

                # Выбор действия
                # В SARSA следующее действие выбиралось после шага в среде
                action = self.make_action(state)

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

                # Правило обновления Q для SARSA (для сравнения)
                # self.Q[state][action] = self.Q[state][action] + self.lr * \
                #     (rew + self.gamma * self.Q[next_state][next_action] -
self.Q[state][action])

                # Правило обновления для Q-обучения
                self.Q[state][action] = self.Q[state][action] + self.lr * \
                    (rew + self.gamma * np.max(self.Q[next_state]) -
self.Q[state][action])

                # Следующее состояние считаем текущим
                state = next_state
                # Суммарная награда за эпизод
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)

def play_agent(agent):
    '''
    Проигрывание сессии для обученного агента
    '''
    env2 = gym.make('Taxi-v3')
    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_q_learning():
    env = gym.make('Taxi-v3')
    lr_list = np.linspace(0.0005, 0.005, num=10)
    gamma_list = np.linspace(0.9, 1, num=10)
    eps_list = np.linspace(0.05, 0.9, num=18)
    for l in tqdm(lr_list):
        for g in gamma_list:
            for e in eps_list:
```

```
                agent = QLearning_Agent(env, lr=l, gamma=g, eps=e)
                agent.learn()
                agent.print_q()
                parameter.append([l,g,e])

def main():
    run_q_learning()


if __name__ == '__main__':

    st = time.time()
    main()
    print(all_reward)
    print('Максимальная награда:',np.max(all_reward),'Значения
гиперпараметров(lr, gamma, eps):',parameter[np.argmax(np.max(all_reward))])
    all_time = time.time() - st
    print(f"Закончено за {all_time:.3f} сек")
    parameter = np.asarray(parameter)
    print(parameter.shape)
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')
    ax.scatter(parameter[:,0], parameter[:,1], parameter[:,2], c=all_reward,
cmap='viridis')
    ax.set_xlabel('lr')
    ax.set_ylabel('gamma')
    ax.set_zlabel('eps')

    plt.show()
```

Результат

```
Суммарная награда: -228.96034222661564 lr = 0.005 gamma = 1.000 eps = 0.400
Суммарная награда: -239.05364228309617 lr = 0.005 gamma = 1.000 eps = 0.450
Суммарная награда: -253.09809862521902 lr = 0.005 gamma = 1.000 eps = 0.500
Суммарная награда: -262.9567144348629 lr = 0.005 gamma = 1.000 eps = 0.550
Суммарная награда: -276.86067028312914 lr = 0.005 gamma = 1.000 eps = 0.600
Суммарная награда: -291.39466513427624 lr = 0.005 gamma = 1.000 eps = 0.650
Суммарная награда: -298.31852662801555 lr = 0.005 gamma = 1.000 eps = 0.700
Суммарная награда: -303.6991909048701 lr = 0.005 gamma = 1.000 eps = 0.750
Суммарная награда: -326.7691587461088 lr = 0.005 gamma = 1.000 eps = 0.800
Суммарная награда: -338.7621324012641 lr = 0.005 gamma = 1.000 eps = 0.850
Суммарная награда: -351.817314964316 lr = 0.005 gamma = 1.000 eps = 0.900
100%| 10/10 [32:30<00:00, 195.03s/it]
[-15.356927953938648, -16.186181447039125, -16.709467161823333, -18.320897424063517, -19.203609079467892, -20.720873632264933, -21.33661051356132, -23.19532
4746536148, -24.94849386826697, -25.18533957951094, -26.156204421871646, -27.810635402676084, -29.515975878355768, -29.934158712802294, -31.882105598371066,
 -33.997199958269476, -34.27616432397967, -35.85813326088808, -15.254046833435211, -16.24671003847819, -17.108731787018215, -18.087064065950642, -19.6252722
```

```
9.78416357472327, -290.20248933660065, -304.15469322426657, -314.261338173133, -326.0492969608045, -338.6859444050058, -358.0325200479904, -154.125779454641
22, -159.00191667101012, -171.18783406138527, -180.50866307384442, -191.95275542440868, -201.60056084642804, -212.26257986952868, -224.13534161302462, -237.
9546859560292, -251.4235775737367, -262.08844226566566, -281.1857196237859, -284.8649033737944, -300.9027307606787 6, -316.20966494844373, -333.1247394583338
, -343.3917260031196 3, -350.5189325872095, -152.12252802895463, -163.59618952037818, -167.35899825403334, -176.18613797380573, -194.6368978894255, -205.0389
076863992, -208.4628028667072, -228.9736025731726, -237.91638034022316, -248.10411338657883, -262.8667462435417, -271.3448871542664, -295.75236014516327, -3
07.71614893464664, -317.8131755291738, -326.45972360882297, -345.60629773108343, -353.38292073552185, -151.72527499165162, -163.6825539102258, -169.54032777
310528, -180.21668598729866, -188.8240441239902, -201.05416069104774, -215.0563656760213, -227.81893253135135, -234.66985994896595, -257.6632544417406, -265
.26980226371455, -276.87635100034043, -295.206358115646, -292.8718544892223, -316.39245817470015, -322.86654884209673, -334.7921144737271, -356.558267484433
94, -150.8141793824053, -162.68376028791636, -170.23221912154276, -181.08557033344835, -191.93714831045264, -203.5501914212738, -215.30252514617627, -223.87
7022787702, -234.88342909532736, -253.39168756543108, -262.44340849679537, -272.72014246980604, -291.79443929545386, -304.6494076193246, -312.9460662853966,
 -328.12603211307294, -342.5376249062772, -352.48570346063843, -152.72545506138937, -162.82647612180855, -168.6553242317743, -185.1071734522309, -189.891381
0806489, -204.6856440242997, -209.68642249029978, -228.91984193231164, -239.85444047499703, -249.36257662283268, -266.94386739408776, -272.27115612785883, -
286.74704170342164, -299.1870385245221, -309.48908679185195, -324.7762427511152, -335.6272288731094, -364.78495150456047, -153.91614186821377, -159.87047970
423438, -171.57729209382327, -182.5277337444908, -195.6412372492881, -204.10169964304885, -217.84135364704832, -221.37483442659016, -240.49462521785767, -25
1.94624696547618, -266.76430915672313, -281.5585401356149, -285.9833509960298, -297.75504376369787, -309.62792198791067, -327.1051901122547, -343.6313267369
5293, -357.04066429929844, -155.80906343201 8, -163.96784040168745, -172.40032166431553, -180.7136954865469, -188.88592942497655, -204.62182367978158, -220.0459831
1060133, -224.6890980562266, -236.95767148330165, -250.23276491631503, -262.0607901205312, -270.46217778618734, -290.7325164409486, -302.43325800770765, -318.7488
27574286, -328.17402164635706, -335.2684219984369, -361.5048602348735, -150.94531242701288, -162.30124231286993, -173.3603303273361, -179.9900999325164, -192.9168
1019582637, -203.37629617845693, -211.491232646829, -224.55614602293457, -238.0917396174853, -256.18054960140927, -264.0519518539769, -275.7326425354201, -287.925
69893476997, -303.0172137820381, -314.0635593570171, -330.43710807016316, -346.8953394777405, -355.52550943330607, -153.2460441667378, -157.7991575346727, -172.35
711891352992, -183.58159898630362, -190.6255064292306, -205.89257889449533, -219.87957840893895, -228.96034222661564, -239.05364228309617, -253.09809862521902, -2
62.9567144348629, -276.86067028312914, -291.39466513427624, -298.31852662801555, -303.6991909048701, -326.7691587461088, -338.7621324012641, -351.817314964316]
Максимальная награда: -14.936735066696 49 Значения гиперпараметров(lr, gamma, eps): [0.0005, 0.9, 0.05]
Закончено за 1950.366 сек
(1800, 3)
```

Как можно заметить, максимальная награда: -14.936735066696 49 со значениями гиперпараметров (lr, gamma, eps): [0.0005, 0.9, 0.05]. Количество комбинаций – 1800.