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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)

ОТЧЕТ

Рубежный контроль №2

по курсу «Методы машинного обучения»

ИСПОЛНИТЕЛЬ: Группа ИУ5-25М
Ваксина И.Р.
ФИО
_____
подпись

"__"_____2023 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО
_____
подпись

"__"_____2023 г.

Москва - 2023

Задание:

Для алгоритма временных различий SARSA, реализованного Вами в соответствующей лабораторная работе, осуществите подбор гиперпараметров. Критерием оптимизации должна являться суммарная награда.

Выполнение:

```
import numpy as np
import matplotlib.pyplot as plt
import gymnasium as gym
from tqdm import tqdm
import matplotlib
matplotlib.use('TkAgg')

class BasicAgent:
    ALGO_NAME = '---'

    def __init__(self, env, eps=0.1):
        self.env = env
        self.nA = env.action_space.n
        self.nS = env.observation_space.n
        self.Q = np.zeros((self.nS, self.nA))
        self.eps=eps
        self.episodes_reward = []

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

    def get_state(self, state):

        if type(state) is tuple:
            return state[0]
        else:
            return state

    def greedy(self, state):
        return np.argmax(self.Q[state])

    def make_action(self, state):
        if np.random.uniform(0,1) < self.eps:
```

```
            return self.env.action_space.sample()
        else:
            return self.greedy(state)

    def draw_episodes_reward(self):
        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():
        pass

class SARSA_Agent(BasicAgent):
    ALGO_NAME = 'SARSA'

    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
        super().__init__(env, eps)
        self.lr=lr
        self.gamma = gamma
        self.num_episodes=num_episodes
        self.eps_decay=0.00005
        self.eps_threshold=0.01

    def learn(self):
        self.episodes_reward = []
        for ep in tqdm(list(range(self.num_episodes))):
            state = self.get_state(self.env.reset())
            done = False
            truncated = False
            tot_rew = 0

            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            action = self.make_action(state)

            while not (done or truncated):
                next_state, rew, done, truncated, _ = self.env.step(action)
```

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

            state = next_state
            action = next_action
            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('CliffWalking-v0', render_mode='human')
    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 plot_rewards(x, y):
    # Построение графика наград по эпизодам
    fig, ax = plt.subplots(figsize = (15,10))
    plt.plot(x, y, '-', linewidth=1, color='green')
    plt.title('Награды')
    plt.xlabel('Параметр')
    plt.ylabel('Награда')
    plt.show()

def bruteforce_sarsa():
    env = gym.make('CliffWalking-v0')
    rewards_eps = []
    rewards_lr = []
    rewards_gamma = []
    x = np.arange(0.1, 1, 0.1)
    for i in x:
        agent = SARSA_Agent(env,eps=i)
        agent.learn()
        agent.print_q()
        rewards_eps.append(np.asarray(agent.episodes_reward).sum())
```

```
    plot_rewards(x, rewards_eps)
    best_eps = x[rewards_eps.index(max(rewards_eps))]
    print(f"Best eps: {best_eps}")
    x = np.arange(0, 1, 0.03)
    for i in x:
        agent = SARSA_Agent(env, eps = best_eps, lr = i)
        agent.learn()
        agent.print_q()
        rewards_lr.append(np.asarray(agent.episodes_reward).sum())
    best_lr = x[rewards_lr.index(max(rewards_lr))]
    print(f"Best lr: {best_lr}")
    plot_rewards(x, rewards_lr)
    x = np.arange(0, 1, 0.03)
    for i in x:
        agent = SARSA_Agent(env, eps = best_eps, lr = best_lr, gamma = i)
        agent.learn()
        agent.print_q()
        rewards_gamma.append(np.asarray(agent.episodes_reward).sum())
    best_gamma = x[rewards_gamma.index(max(rewards_gamma))]
    print(f"Best gamma: {best_gamma}")
    plot_rewards(x, rewards_gamma)
    print(rewards_eps)
    print(rewards_lr)
    print(rewards_gamma)
    print(f"Best params: eps={best_eps}, lr={best_lr}, gamma={best_gamma}")

def run_sarsa():
    env = gym.make('CliffWalking-v0')
    agent = SARSA_Agent(env, eps=0.1, lr=0.39, gamma=0.93)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)


def main():
    bruteforce_sarsa()
    # run_sarsa()
    # run_q_learning()
    # run_double_q_learning()

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