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)

Курс «Методы машинного обучения»

Отчет по рубежному контролю №2

Выполнила:
студентка группы ИУ5-24М
Мащенко Е. И.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

2023

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 работы

Листинг:

Рубежный контроль №2

Выполнила: Мащенко Е.И., ИУ5-24М

Алгоритм SARSA

Для проведения работы была выбрана среда обучения с подкреплением CliffWalking из библиотеки Gym. Проведем подбор гиперпараметров для алгоритма SARSA. Критерий оптимизации – суммарная награда. В класс SARSA_Agent добавлен метод sum_rewards, подсчитывающий итоговую суммарную награду.

B [1]:
```
! pip install gymnasium
```

```
Requirement already satisfied: gymnasium in c:\users\user\anaconda3\lib\site-packages (0.28.1)
Requirement already satisfied: cloudpickle>=1.2.0 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (1.6.0)
Requirement already satisfied: jax-jumpy>=1.0.0 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (1.0.0)
Requirement already satisfied: typing-extensions>=4.3.0 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (4.6.
3)
Requirement already satisfied: importlib-metadata>=4.8.0; python_version < "3.10" in c:\users\user\anaconda3\lib\site-packag
es (from gymnasium) (6.6.0)
Requirement already satisfied: numpy>=1.21.0 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (1.24.3)
Requirement already satisfied: farama-notifications>=0.0.1 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (0.
0.4)
Requirement already satisfied: zipp>=0.5 in c:\users\user\anaconda3\lib\site-packages (from importlib-metadata>=4.8.0; pytho
n_version < "3.10"->gymnasium) (3.4.0)
```

B [2]:
```
import gymnasium as gym
import numpy as np
from pprint import pprint
import matplotlib.pyplot as plt
from tqdm import tqdm
```

In [3]:

```
# ********************************** 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
```

In [4]:

```
# ********************************** SARSA ****************************************

class SARSA_Agent(BasicAgent):
    '''
    Реализация алгоритма SARSA
    '''
    # Наименование алгоритма
    ALGO_NAME = 'SARSA'


    def __init__(self, env, eps=0.015, lr=0.1, gamma=0.97, num_episodes=20000):
        # 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 learn(self):
        '''
        Обучение на основе алгоритма 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)

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

                # Следующее состояние считаем текущим
                state = next_state
                action = next_action
                # 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 SARSA: ', sum_rewards)
```

В [5]:
```
!pip install pygame
```

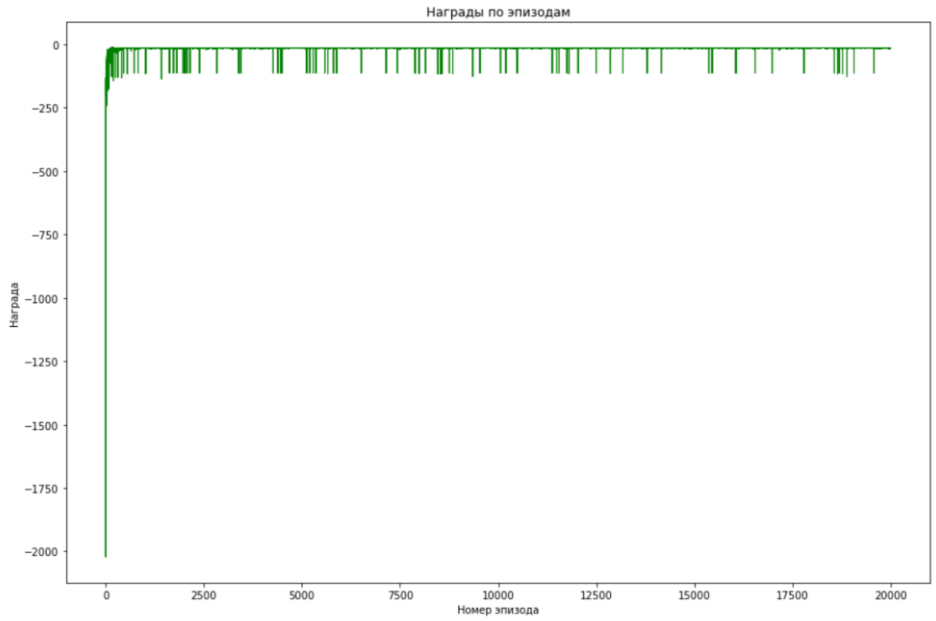
```
Requirement already satisfied: pygame in c:\users\user\anaconda3\lib\site-packages (2.4.0)
```

В [6]:
```
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
```

В [7]:
```
env = gym.make('CliffWalking-v0')
agent = SARSA_Agent(env)
agent.learn()
agent.print_q()
agent.draw_episodes_reward()
agent.sum_rewards()
# play_agent(agent)
```

```
100%|██████████████████████████████████████████████████████████| 20000/20000 [00:08<00:00, 2238.39it/s]
```

```
Вывод Q-матрицы для алгоритма  SARSA
[[ -11.45890587  -11.39245172  -11.39377187  -11.50468673]
 [ -10.89681885  -10.83580844  -10.84237668  -10.93864324]
 [ -10.22523906  -10.20005169  -10.19983023  -10.23798437]
 [  -9.54558935   -9.49992106   -9.49954321   -9.70032823]
 [  -8.77471888   -8.76895524   -8.77424204   -8.87733262]
 [  -8.15384977   -8.01546422   -8.03136638   -8.15792674]
 [  -7.24147696   -7.23595199   -7.24074512   -7.24469757]
 [  -6.61617109   -6.43480788   -6.43869993   -6.7660588 ]
 [  -5.68927395   -5.61640292   -5.62188551   -5.73445781]
 [  -4.86917696   -4.78068798   -4.90825246   -4.89313555]
 [  -4.04316019   -3.90536008   -4.02795145   -4.12075341]
 [  -3.34324784   -3.29911547   -2.93385819   -3.15528184]
 [ -11.91221621  -10.92728946  -12.46168126  -11.65637737]
 [ -11.32114775  -10.2284693   -11.64161679  -11.78660572]
 [ -10.84796841   -9.50819649  -10.93583543  -11.10881937]
 [ -10.15738872   -8.76557432   -9.85037117  -10.40821276]
 [  -9.4614996    -8.00042359   -9.20033474   -9.71738125]
 [  -8.73939261   -7.2127831    -8.45797601   -8.98871495]
 [  -7.98188521   -6.40232418   -7.87363869   -8.19105959]
 [  -7.22113978   -5.56814017   -7.10665653   -7.42706066]
 [  -6.38362208   -4.70897585   -6.33519224   -6.57419574]
 [  -5.59550226   -3.8235841    -5.55401287   -5.74354892]
 [  -4.76213816   -2.9109       -3.17226568   -4.89172659]
 [  -3.86510534   -2.92718412   -1.97         -4.07496077]
 [ -11.60623464  -13.67829282  -13.07345187  -12.33735645]
 [ -10.98894988  -11.62500195  -44.23958802  -11.46876534]
 [ -10.30416498  -10.9597      -29.11102548  -10.34146159]
 [  -9.24736811   -9.21163482  -37.23464082   -9.2466042 ]
 [  -8.67416659   -8.61674017  -28.90282086   -8.67424697]
 [  -7.93509178   -7.9387103   -28.62057099   -7.97193829]
 [  -7.2034112    -7.20577184  -20.29415789   -7.33251354]
 [  -6.42329716   -6.42138523  -10.12788496   -6.55531627]
 [  -5.59916709   -5.5950909   -29.46174181   -5.63643723]
 [  -4.73600148   -5.56918049  -20.24106386   -4.91527203]
 [  -2.54594655   -1.97        -45.061545     -2.74491248]
 [  -3.08351178   -2.04213384   -1.           -2.90556086]
 [ -12.26696885 -111.51402964  -12.94518914  -12.93810748]
 [   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.        ]
 [   0.            0.            0.            0.        ]]
```



Суммарная награда SARSA: -319923

Результаты изменения суммарной награды от скорости обучения lr представлены в Таблице 1. График зависимости суммарной награды от lr представлен на рис.1.

Таблица 1. Результаты изменения суммарной награды от скорости обучения lr

Суммарная награда	Награда за последний эпизод	lr	eps	gamma	episodes
-529192	-17	0,025	0,4	0,98	20000
-499310	-17	0,05	0,4	0,98	20000
-484401	-17	0,1	0,4	0,98	20000
-488352	-17	0,15	0,4	0,98	20000
-488035	-19	0,2	0,4	0,98	20000
-497641	-17	0,25	0,4	0,98	20000

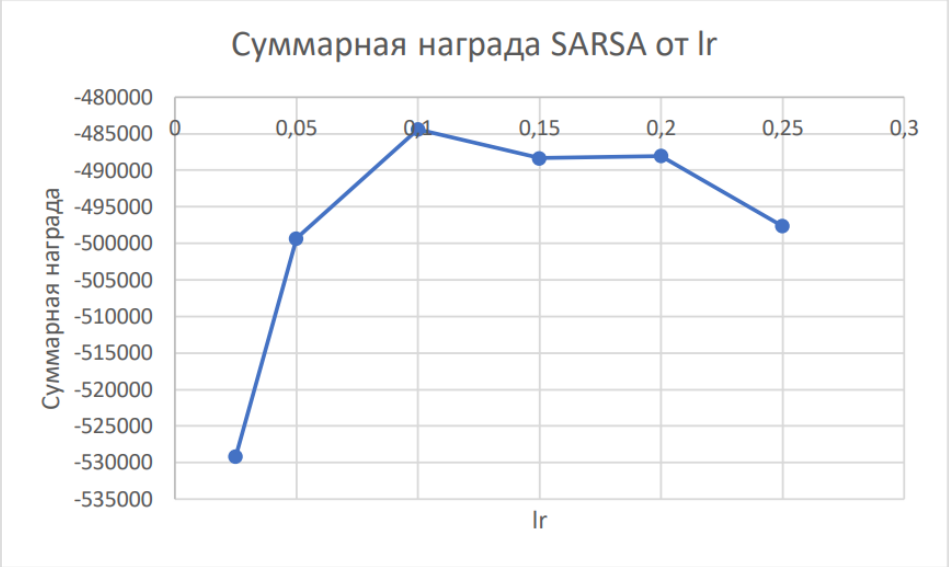


Рис.1. График зависимости суммарной награды от lr

Результаты изменения суммарной награды от параметра eps представлены в Таблице 2. График зависимости суммарной награды от параметра eps представлен на рис.2.

Таблица 2. Результаты изменения суммарной награды от параметра eps

Суммарная награда	Награда за последний эпизод	lr	eps	gamma	episodes
-321854	-15	0,1	0,015	0,98	20000
-322896	-15	0,1	0,025	0,98	20000
-322810	-15	0,1	0,05	0,98	20000
-329450	-15	0,1	0,1	0,98	20000
-351044	-15	0,1	0,2	0,98	20000
-422912	-17	0,1	0,3	0,98	20000
-484401	-17	0,1	0,4	0,98	20000
-586762	-17	0,1	0,5	0,98	20000
-806757	-17	0,1	0,6	0,98	20000

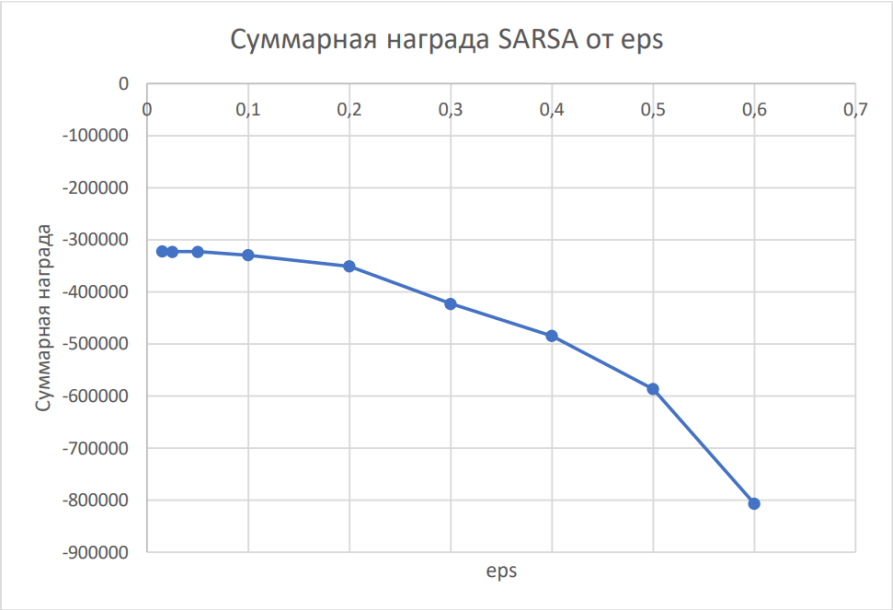


Рис.2. График зависимости суммарной награды от параметра eps

Результаты изменения суммарной награды от параметра gamma представлены в Таблице 3. График зависимости суммарной награды от параметра gamma представлен на рис.3.

Таблица 3. Результаты изменения суммарной награды от параметра gamma

Суммарная награда	Награда за последний эпизод	lr	eps	gamma	episodes
-321112	-15	0,1	0,015	0,96	20000
-319923	-15	0,1	0,015	0,97	20000
-321854	-15	0,1	0,015	0,98	20000
-321521	-15	0,1	0,015	0,99	20000
-321573	-15	0,1	0,015	0,995	20000

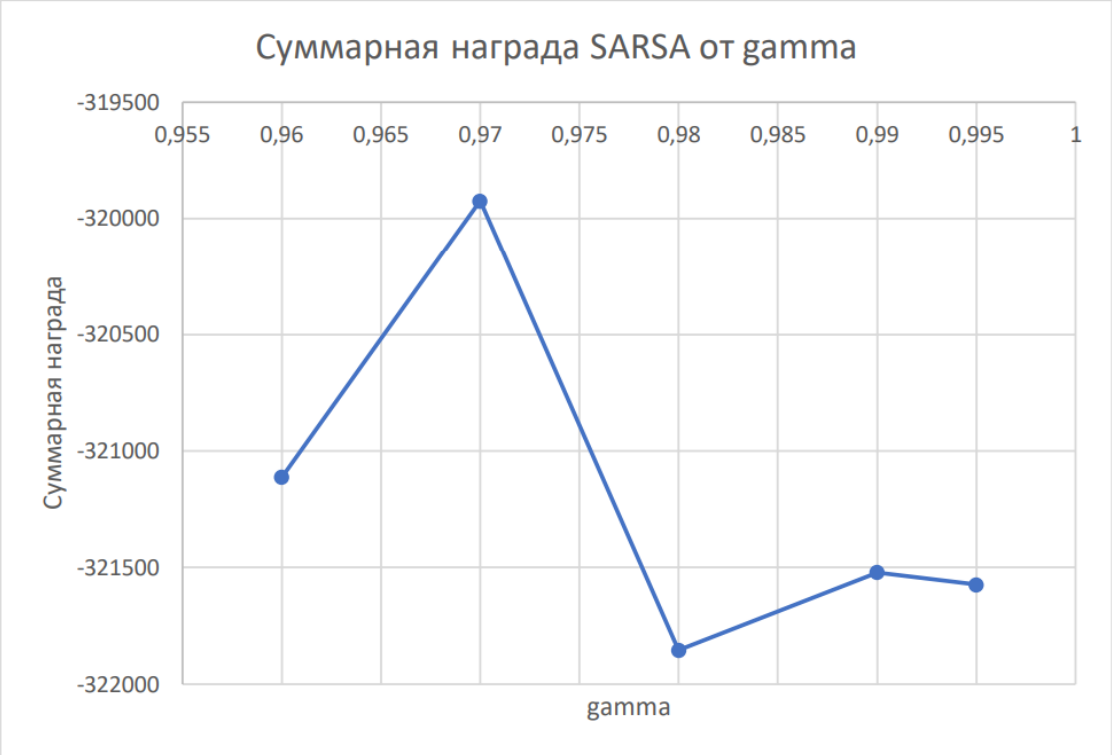


Рис.3. График зависимости суммарной награды от параметра gamma

Вывод

В результате подбора гиперпараметров лучшими значениями оказались: eps=0.015, lr=0.1, gamma=0.97, num_episodes=20000. При этом, при уменьшении eps стратегия агента приближалась к максимальной (к движению по краю обрыва).