**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

# Рубежный контроль №2

# «Методы обучения с подкреплением»

ИСПОЛНИТЕЛЬ: Филатова А. Е.
ФИО

группа ИУ5-23М _____
подпись

"_"_____2023 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО

_____
подпись

"_"_____2023 г.

Москва - 2023

## 1. 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.

## 2. Код программы

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


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 в 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


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=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 Q-Learning
        ''' 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

            # 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('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 run_q_learning():
    env = gym.make("CliffWalking-v0")

    epsilons = [0.3, 0.4, 0.5]
    learning_rates = [0.05, 0.1, 0.2]
    gammas = [0.95, 0.98, 0.99]
    num_episodes = 20000

    best_reward = float('-inf')
    best_hyperparams = None
    best_agent = None

    for eps in epsilons:
        for lr in learning_rates:
            for gamma in gammas:
                            agent = QLearning_Agent(env, eps=eps, lr=lr,
gamma=gamma, num_episodes=num_episodes)
                agent.learn() total_reward =
                sum(agent.episodes_reward)
                        print(f'Гиперпараметры: epsilon={eps}, learning
rate={lr}, gamma={gamma}, num episodes={num_episodes}')
                print(f'Суммарная награда: {total_reward}\n')

                if total_reward > best_reward:
                    best_reward  =  total_reward  best_hyperparams  =
                    (eps, lr, gamma, num_episodes) best_agent = agent

        print(f'Лучшие гиперпараметры: epsilon={best_hyperparams[0]},
```

```
learning rate={best_hyperparams[1]}, gamma={best_hyperparams[2]}, num
episodes={best_hyperparams[3]}')        print(f'Суммарная        награда:
{best_reward}\n')

    best_agent.print_q() best_agent.draw_episodes_reward()
    play_agent(best_agent)


def main():
    run_q_learning()


if name == ' main ': main()
```

## 3. Результаты подбора гиперпараметров

### 3.1. Гиперпараметры: epsilon = 0.3, learning rate = 0.05, gamma = 0.95, num episodes = 20000

Суммарная награда: -734106

### 3.2. Гиперпараметры: epsilon = 0.3, learning rate = 0.05, gamma = 0.98, num episodes = 20000

Суммарная награда: -719783


Гиперпараметры: epsilon = learning gamma = 0.99, num episodes = 20000 Суммарная награда:


3.3. 0.3, rate = 0.05,

-724730


3.4. Гиперпараметры: epsilon = 0.3, learning rate = 0.1, gamma = 0.95, num episodes = 20000 Суммарная награда: -738206


3.5. Гиперпараметры: epsilon = 0.3, learning rate = 0.1, gamma = 0.98, num episodes = 20000 Суммарная награда: -723993


3.6. Гиперпараметры: epsilon = 0.3, learning rate = 0.1, gamma = 0.99, num episodes = 20000 Суммарная награда: -747135


3.7. Гиперпараметры: epsilon = 0.3, learning rate = 0.2, gamma = 0.95, num episodes = 20000 Суммарная награда: -690261

Гиперпараметры: epsilon = 0.3, learning gamma = 0.99, num episodes = 20000 Суммарная награда:


3.8. Гиперпараметры: epsilon = 0.3, learning rate = 0.2, gamma = 0.98, num episodes = 20000

Суммарная награда: -695174

3.9. rate = 0.2,


-697318

3.10. Гиперпараметры: epsilon = 0.4, learning rate = 0.05, gamma = 0.95, num episodes = 20000 Суммарная


награда: -1187325

3.11. Гиперпараметры: epsilon = 0.4, learning rate = 0.05, gamma = 0.98, num episodes = 20000 Суммарная


награда: -1187912

3.12. Гиперпараметры: epsilon = 0.4, learning rate = 0.05, gamma = 0.99, num episodes = 20000 Суммарная


награда: -1187358

3.13. Гиперпараметры: epsilon = 0.4, learning rate = 0.1, gamma =

Гиперпараметры: epsilon = 0.4, learning gamma = 0.99, num episodes = 20000

Суммарная награда:


0.95, num episodes = 20000 Суммарная награда: -1067977

3.14. Гиперпараметры: epsilon = 0.4, learning rate = 0.1, gamma =


0.98, num episodes = 20000

Суммарная награда: -1073649

3.15. rate = 0.1,


-1085515

3.16. Гиперпараметры: epsilon = 0.4, learning rate = 0.2, gamma = 0.95, num episodes = 20000 Суммарная


награда: -1085515

3.17. Гиперпараметры: epsilon = 0.4, learning rate = 0.2, gamma = 0.98, num episodes = 20000 Суммарная


награда: -1001796

3.18. Гиперпараметры: epsilon = 0.4, learning rate = 0.2, gamma =

Гиперпараметры: epsilon = 0.5, learning gamma = 0.99, num episodes = 20000 Суммарная награда:

0.99, num episodes = 20000 Суммарная награда: -992852

3.19. Гиперпараметры: epsilon = 0.5, learning rate = 0.05, gamma = 0.95, num episodes = 20000 Суммарная награда: -1753563

3.20. Гиперпараметры: epsilon = 0.5, learning rate = 0.05, gamma = 0.98, num episodes = 20000

Суммарная награда: -1753563

3.21. rate = 0.05,

-1767028

3.22. Гиперпараметры: epsilon = 0.5, learning rate = 0.1, gamma = 0.95, num episodes = 20000

Суммарная награда: -1553120

3.23. Гиперпараметры: epsilon = 0.5, learning rate = 0.1, gamma = 0.98, num episodes = 20000

Суммарная награда: -1593516

Гиперпараметры: epsilon = 0.6, learning gamma = 0.99, num episodes = 20000

Суммарная награда:


3.24. Гиперпараметры: epsilon = 0.5, learning rate = 0.1, gamma = 0.99, num episodes = 20000 Суммарная


награда: -1573441

3.25. Гиперпараметры: epsilon = 0.5, learning rate = 0.2, gamma = 0.95, num episodes = 20000 Суммарная


награда: -1419773

3.26. Гиперпараметры: epsilon = 0.5, learning rate = 0.2, gamma = 0.98, num episodes = 20000


Суммарная награда: -1418196


3.27. 0.5, rate = 0.2,

-1418465


## 4. Вывод

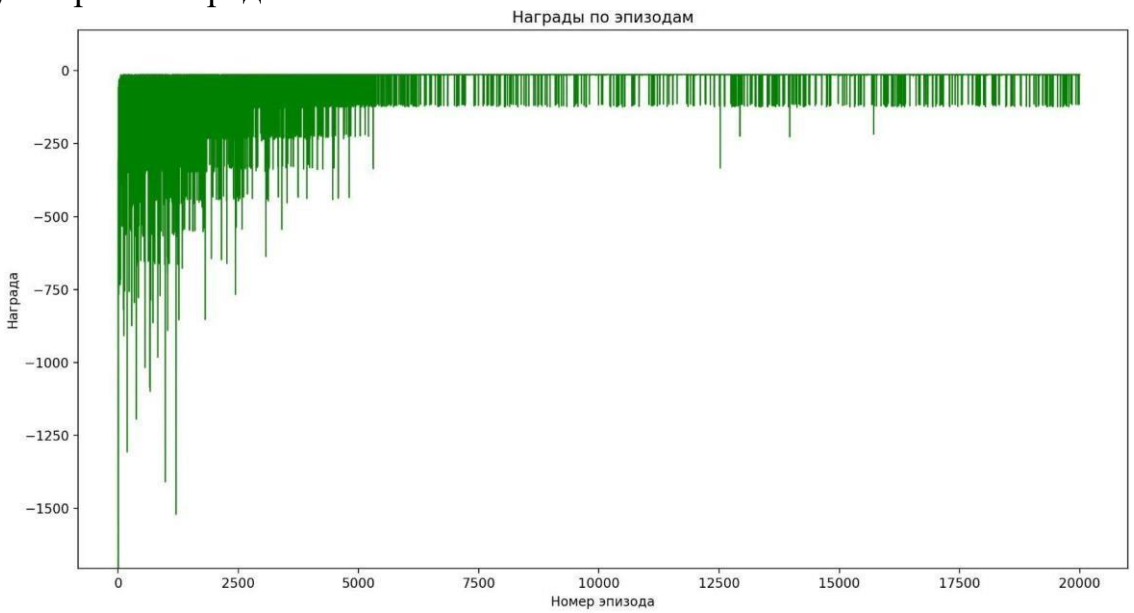
Лучшие гиперпараметры: epsilon = 0.3, learning rate = 0.2, gamma = 0.95, num episodes = 20000 Суммарная награда: -690261

```
Лучшие гиперпараметры: epsilon=0.3, learning rate=0.2, gamma=0.95, num episodes=20000
Суммарная награда: -690261

Вывод Q-матрицы для алгоритма  Q-обучение
[[ -10.15122819  -10.16080799  -10.15528908  -10.2115043 ]
 [  -9.78189542   -9.72532187   -9.72634666   -9.92690154]
 [  -9.52241221   -9.19199606   -9.19209058   -9.70172541]
 [  -8.78334646   -8.62394292   -8.62393363   -9.44289466]
 [  -8.55195742   -8.02524897   -8.0252482    -9.04865249]
 [  -7.96616829   -7.3950095    -7.39500978   -8.38539217]
 [  -7.28558023   -6.73159123   -6.7315912    -8.00852816]
 [  -6.57247631   -6.03325406   -6.03325406   -7.29529639]
 [  -5.99462608   -5.29816219   -5.29816219   -6.69052357]
 [  -5.24931545   -4.52438125   -4.52438125   -5.9616897 ]
 [  -4.52246015   -3.709875     -3.709875     -5.10578582]
 [  -3.64362124   -3.51325354   -2.8525       -4.40984427]
 [ -10.55568587   -9.73315833   -9.73315833  -10.24602962]
 [ -10.22844006   -9.19279825   -9.19279825  -10.24626327]
 [  -9.7310139    -8.62399815   -8.62399815   -9.73314855]
 [  -9.19262235   -8.02526122   -8.02526122   -9.19279805]
 [  -8.62385584   -7.39501181   -7.39501181   -8.62399799]
 [  -8.02524409   -6.73159137   -6.73159137   -8.02526113]
 [  -7.39501061   -6.03325408   -6.03325408   -7.39501121]
 [  -6.7315913    -5.29816219   -5.29816219   -6.73159137]
 [  -6.03325406   -4.52438125   -4.52438125   -6.03325408]
 [  -5.29816215   -3.709875     -3.709875     -5.29816211]
 [  -4.52438118   -2.8525       -2.8525       -4.5243811 ]
 [  -3.709875     -2.8525       -1.95         -3.70987483]
 [ -10.24650042   -9.19279825  -10.24650042   -9.73315833]
 [  -9.73315833   -8.62399815 -109.24650042   -9.73315833]
 [  -9.19279825   -8.02526122 -109.24650042   -9.19279825]
 [  -8.62399815   -7.39501181 -109.24650042   -8.62399815]
 [  -8.02526122   -6.73159137 -109.24650042   -8.02526122]
 [  -7.39501181   -6.03325408 -109.24650042   -7.39501181]
 [  -6.73159137   -5.29816219 -109.24650042   -6.73159137]
 [  -6.03325408   -4.52438125 -109.24650042   -6.03325408]
 [  -5.29816219   -3.709875   -109.24650042   -5.29816219]
 [  -4.52438125   -2.8525     -109.24650042   -4.52438125]
 [  -3.709875     -1.95       -109.24650042   -3.709875  ]
 [  -2.8525       -1.95         -1.           -2.8525    ]
 [  -9.73315833 -109.24650042  -10.24650042  -10.24650042]
 [   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.        ]]
```

Гиперпараметры: epsilon = learning gamma = 0.99, num episodes = 20000

Суммарная награда:




Для исходных гиперпараметров epsilon = 0.4, learning rate = 0.1, gamma = 0.98, num episodes = 20000 суммарная награда равнялась -1073649, следовательно подбор гиперпараметров помог улучшить результаты обучения и суммарную награду. Подбор гиперпараметров является важным шагом в обучении моделей и может значительно улучшить качество работы модели.