МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Лабораторная работа №__5__
по дисциплине «Методы машинного обучения в автоматизированных системах»

Тема: «Обучение на основе временны’х различий»

ИСПОЛНИТЕЛЬ: ___Калюта Н.И.___
ФИО
группа ИУ5-22М _____
подпись
"30"___05___2024 г.

ПРЕПОДАВАТЕЛЬ: _____
ФИО
_____
подпись
"__"_____2024 г.

Москва - 2024

Цель лабораторной работы:

Ознакомление с базовыми методами обучения с подкреплением на основе временных различий.

Задание:

На основе рассмотренного на лекции примера реализуйте следующие алгоритмы:

- SARSA
- Q-обучение
- Двойное Q-обучение

Для любой среды обучения с подкреплением (кроме рассмотренной на лекции среды Toy Text / Frozen Lake) из библиотеки Gym (или аналогичной библиотеки).

Установка необходимых библиотек и зависимостей для работы с Atari-играми

```
#remove " > /dev/null 2>&1" to see what is going on under the hood
!apt-get update > /dev/null 2>&1
!pip install gym pyvirtualdisplay > /dev/null 2>&1
!apt-get install -y xvfb python-opengl ffmpeg > /dev/null 2>&1
!apt-get install cmake > /dev/null 2>&1
!pip install --upgrade setuptools 2>&1
!pip install ez_setup > /dev/null 2>&1

!apt-get update > /dev/null 2>&1
!apt-get install cmake > /dev/null 2>&1
!pip install --upgrade setuptools 2>&1
!pip install ez_setup > /dev/null 2>&1

!pip install swig==4.1.1
!pip install Box2D==2.3.2
!pip install box2d-kengz==2.3.3
!pip install pygame==2.2.0
!pip install ale_py==0.8.1
!pip install pyglet==1.5.11


!pip install -U colabgymrender
!pip install imageio==2.4.1
!pip install --upgrade AutoROM
!AutoROM --accept-license
!pip install gymnasium[atari]==0.28.1

!wget http://www.atarimania.com/roms/Roms.rar
!unrar x -o+ /content/Roms.rar >/dev/nul
!python -m atari_py.import_roms /content/ROMS >/dev/nul

!pip install pyvirtualdisplay > /dev/null 2>&1

!pip install --upgrade gym
```

Установка библиотек, необходимые для работы с графикой и видео

```
[ ] !apt-get install xvfb
    !apt-get install python3-opengl ffmpeg
```

Создать папку с названием "video"

```
[ ] !mkdir -p video
```

Импорт библиотек

```
[ ] import gymnasium as gym
    from gymnasium import logger as gymlogger
    from gymnasium.wrappers.record_video import RecordVideo

    gymlogger.set_level(40) #error only
    import tensorflow as tf
    import numpy as np
    import random
    import matplotlib
    import matplotlib.pyplot as plt
    %matplotlib inline
    from pprint import pprint
    from tqdm import tqdm
    import math
    import uuid
    import glob
    import io
    import base64
    from IPython.display import HTML

    from IPython import display as ipythondisplay
```

Версия библиотеки Gym

```
[ ] gym.__version__
```

```
'0.28.1'
```

Дополнительные функции

Функции для работы с видеозаписями, создает среду Gym с возможностью записи видео и выводит информацию о среде

```
def show_video(folder_name):
  mp4list = glob.glob(f'{folder_name}/*.mp4')
  if len(mp4list) > 0:
    mp4 = mp4list[0]
    video = io.open(mp4, 'r+b').read()
    encoded = base64.b64encode(video)
    ipythondisplay.display(HTML(data='''''.format(encoded.decode('ascii'))))
  else:
    print("Could not find video")


def wrap_env(env, folder_name):
  env = RecordVideo(env, folder_name, step_trigger = lambda episode_number: True)
  return env

def create_environment(name):
    folder_name = f"./video/{name}/{uuid.uuid4()}"
    env = wrap_env(gym.make(name, render_mode="rgb_array"), folder_name)
    spec = gym.spec(name)
    print(f"Action Space: {env.action_space}")
    print(f"Observation Space: {env.observation_space}")
    print(f"Max Episode Steps: {spec.max_episode_steps}")
    print(f"Nondeterministic: {spec.nondeterministic}")
    print(f"Reward Range: {env.reward_range}")
    print(f"Reward Threshold: {spec.reward_threshold}")
    return env, folder_name
```

Создает невидимый виртуальный дисплей размером 1400x900 пикселей

```
[ ] from pyvirtualdisplay import Display
    display = Display(visible=0, size=(1400, 900))
    display.start()
```

```
<pyvirtualdisplay.display.Display at 0x7ec27399b430>
```

Задание агентов

Реализация базового агента

```
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
```

Реализация алгоритма обучения с подкреплением SARSA (State-Action-Reward-State-Action) для агента

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


        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 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, info = 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)
```

Реализация алгоритма обучения с подкреплением Q-Learning

```
[ ]  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, info = self.env.step(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)
```

Реализация алгоритма обучения с подкреплением "Double Q-Learning". Алгоритм Double Q-Learning является вариантом алгоритма Q-Learning, который помогает уменьшить переоценку значений Q и улучшить стабильность обучения.

```
[ ]  class DoubleQLearning_Agent(BasicAgent):
         '''
         Реализация алгоритма Double Q-Learning
         '''
         # Наименование алгоритма
         ALGO_NAME = 'Двойное 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)
             # 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 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)

                     if np.random.rand() < 0.5:
                         # Обновление первой таблицы
                         self.Q[state][action] = self.Q[state][action] + self.lr * \
                             (rew + self.gamma * self.Q2[next_state][np.argmax(self.Q[next_state])] - self.Q[state][action])
                     else:
                         # Обновление второй таблицы
                         self.Q2[state][action] = self.Q2[state][action] + self.lr * \
                             (rew + self.gamma * self.Q[next_state][np.argmax(self.Q2[next_state])] - self.Q2[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)
```

Запуск обученного агента

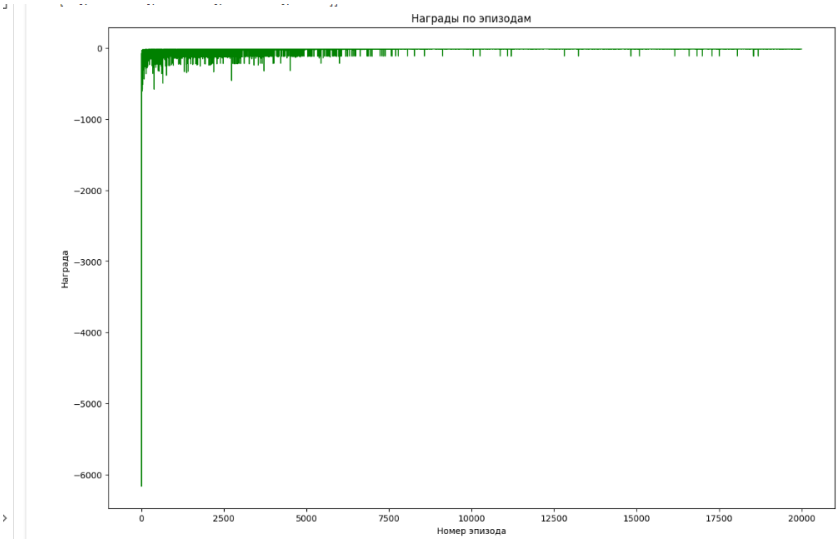
```
[ ]  def play_agent(agent):
         '''
         Проигрывание сессии для обученного агента
         '''
         env2, folder = create_environment('CliffWalking-v0')
         state = env2.reset()[0]
         done = False
         while not done:
             action = agent.greedy(state)
             next_state, reward, terminated, truncated, info = env2.step(action)
             env2.render()
             state = next_state
             if terminated or truncated:
                 done = True
         show_video(folder)
         env2.close()
```

Модель SARSA

Реализация обучения и тестирования агента, использующего алгоритм SARSA, в среде CliffWalking-v0.

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

```
100%|██████████| 20000/20000 [00:19<00:00, 1001.70it/s]
Вывод Q-матрицы для алгоритма  SARSA
[[ -13.27309584  -12.42059652  -14.18560651  -13.23009157]
 [ -12.52723728  -11.63406526  -13.32572907  -13.43462136]
 [ -11.71711824  -10.84586632  -12.59176318  -12.81776419]
 [ -10.94896783  -10.05207534  -11.8824648   -11.87104819]
 [ -10.1604863    -9.25291903  -11.43713433  -11.05410244]
 [  -9.25436708   -8.44404317  -10.41830847  -10.24320023]
 [  -8.41118052   -7.60227846   -9.32430998   -9.6566682 ]
 [  -7.60272476   -6.69517633   -8.54743291   -8.60474238]
 [  -6.65941971   -5.74717331   -7.67441894   -7.73700348]
 [  -5.76680044   -4.82325293   -5.084493     -6.85976233]
 [  -4.88395557   -3.88865444   -4.27499363   -5.97158162]
 [  -3.93843837   -3.93128378   -2.94218548   -5.06204995]
 [ -13.20567238  -13.60119012  -14.90160896  -13.99820776]
 [ -12.44985792  -16.14622045  -22.5800999   -17.63957362]
 [ -11.67955359  -18.83140269  -22.91746279  -19.60752431]
 [ -10.92702308  -15.45915445  -20.73049339  -17.99238306]
 [ -10.10058807  -13.63850672  -19.49032242  -17.19819418]
 [  -9.42033401  -12.87999838  -24.29849744  -15.06925441]
 [  -8.38829297  -10.6384615   -20.16088854  -14.80897564]
 [  -7.56813926  -12.50563276  -24.18637948  -13.6808705 ]
 [  -6.68677933   -9.44961127  -16.49235572  -14.92241975]
 [  -7.69304615   -3.96754014  -13.80114078  -10.86340593]
 [  -5.80650432   -2.96728563   -9.98120593   -7.56475364]
 [  -4.11742778   -2.98771612   -1.98024865   -3.95075492]
 [ -14.01483902  -15.74808275  -15.60238882  -14.66838045]
 [ -21.46986937  -22.69397872 -143.90550209  -14.65559504]
 [ -16.90910482  -30.73156266 -115.56391692  -24.50879773]
 [ -17.39172517  -20.20783261 -113.18440682  -24.66962653]
 [ -15.00694632  -21.59694655 -113.08488266  -23.81749956]
 [ -15.12317147  -24.11325266 -106.81872636  -19.74657498]
 [ -13.61467917  -23.37450366 -110.51021097  -15.9192675 ]
 [ -12.31428697  -24.94986117 -109.9540451   -25.61130423]
 [ -10.72175116  -22.6141395  -128.86842136  -15.62959456]
 [  -7.95827277  -17.15496656 -101.16663028  -20.77078282]
 [  -7.65548228   -2.00736742 -132.02944237  -18.30091514]
 [  -3.169736     -1.98518007   -1.           -3.00668171]
 [ -14.78555715 -114.55566028  -15.5318007   -15.40183578]
 [   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.        ]]
```



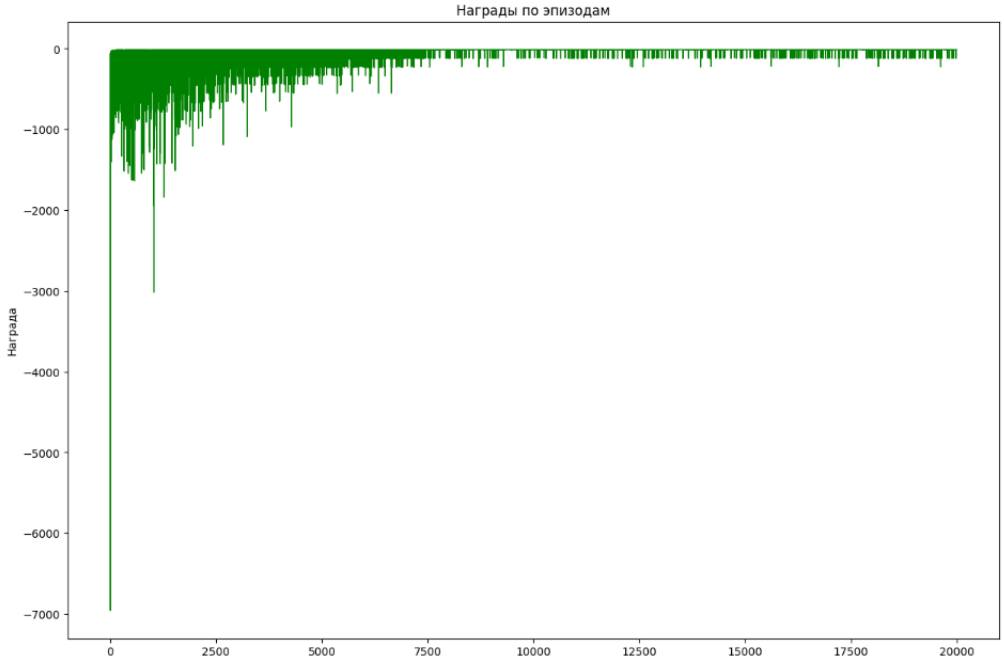
```
Action Space: Discrete(4)
Observation Space: Discrete(48)
Max Episode Steps: None
Nondeterministic: False
Reward Range: (-inf, inf)
Reward Threshold: None
Moviepy - Building video /content/video/CliffWalking-v0/94cad319-bdb2-42f5-9017-6209d1b1e255/rl-video-step-0.mp4.
Moviepy - Writing video /content/video/CliffWalking-v0/94cad319-bdb2-42f5-9017-6209d1b1e255/rl-video-step-0.mp4

                                                                Moviepy - Done !
Moviepy - video ready /content/video/CliffWalking-v0/94cad319-bdb2-42f5-9017-6209d1b1e255/rl-video-step-0.mp4
```

Модель Q-LEARNING

Реализует полный цикл обучения и тестирования агента с использованием алгоритма Q-Learning

```
[ ]  env = gym.make('CliffWalking-v0')
     agent = QLearning_Agent(env)
     agent.learn()
     agent.print_q()
     agent.draw_episodes_reward()
     play_agent(agent)
```

```
100%|██████████| 20000/20000 [00:14<00:00, 1424.37it/s]
Вывод Q-матрицы для алгоритма  Q-обучение
[[ -12.54267749  -12.30953819  -12.30992729  -12.50282846]
 [ -12.08574143  -11.54814658  -11.5481054   -12.72446456]
 [ -11.36249462  -10.76411869  -10.76412268  -11.91952303]
 [ -10.70125375   -9.96342725   -9.96342762  -11.47003321]
 [  -9.84967303   -9.14635892   -9.14635898  -10.72797389]
 [  -9.13371419   -8.31261177   -8.31261178   -9.90910121]
 [  -8.29135132   -7.46184885   -7.46184885   -9.12294194]
 [  -7.45623042   -6.59372333   -6.59372333   -8.28031009]
 [  -6.58170075   -5.70788096   -5.70788096   -7.42188441]
 [  -5.69842512   -4.80396016   -4.80396016   -6.57693918]
 [  -4.78999966   -3.881592     -3.881592     -5.54784001]
 [  -3.84034396   -3.85253885   -2.9404       -4.76362019]
 [ -13.0523722   -11.54888054  -11.54888054  -12.31786083]
 [ -12.31546427  -10.76416381  -10.76416381  -12.31788069]
 [ -11.54872478   -9.96343246   -9.96343246  -11.5488801 ]
 [ -10.7641447    -9.14635966   -9.14635966  -10.76416361]
 [  -9.9634297    -8.31261189   -8.31261189   -9.96343238]
 [  -9.14635887   -7.46184887   -7.46184887   -9.14635965]
 [  -8.31261171   -6.59372334   -6.59372334   -8.31261189]
 [  -7.46184883   -5.70788096   -5.70788096   -7.46184887]
 [  -6.59372333   -4.80396016   -4.80396016   -6.59372332]
 [  -5.70788093   -3.881592     -3.881592     -5.7078809 ]
 [  -4.80396016   -2.9404       -2.9404       -4.80396015]
 [  -3.881592     -2.9404       -1.98         -3.881592  ]
 [ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
 [ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
 [ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
 [  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
 [  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
 [  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
 [  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
 [  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
 [  -5.70788096   -3.881592   -111.31790293   -5.70788096]
 [  -4.80396016   -2.9404     -111.31790293   -4.80396016]
 [  -3.881592     -1.98       -111.31790293   -3.881592  ]
 [  -2.9404       -1.98         -1.           -2.9404    ]
 [ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
 [   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.        ]]
```



Номер эпизода

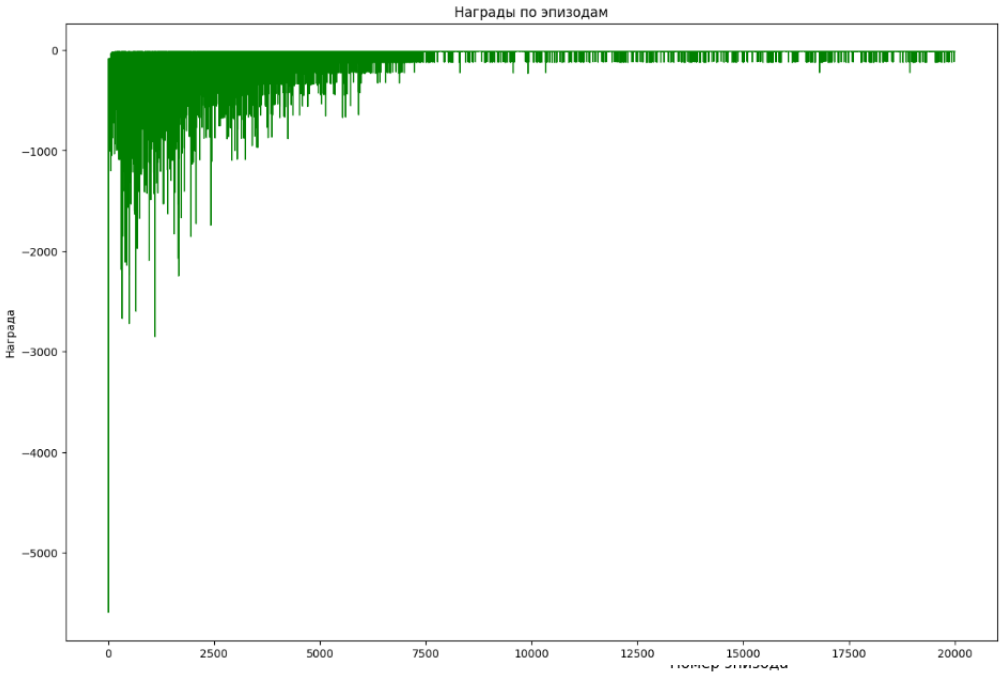
```
Action Space: Discrete(4)
Observation Space: Discrete(48)
Max Episode Steps: None
Nondeterministic: False
Reward Range: (-inf, inf)
Reward Threshold: None
Moviepy - Building video /content/video/CliffWalking-v0/5409ab00-1467-45ec-b6bd-af15351ee8a0/rl-video-step-0.mp4.
Moviepy - Writing video /content/video/CliffWalking-v0/5409ab00-1467-45ec-b6bd-af15351ee8a0/rl-video-step-0.mp4

Moviepy - Done !
Moviepy - video ready /content/video/CliffWalking-v0/5409ab00-1467-45ec-b6bd-af15351ee8a0/rl-video-step-0.mp4
```

Модель DQ-LEARNING

Реализует полный цикл обучения и тестирования агента с использованием алгоритма Double Q-Learning

```
[ ]  env = gym.make('CliffWalking-v0')
     agent = DoubleQLearning_Agent(env)
     agent.learn()
     agent.print_q()
     agent.draw_episodes_reward()
     play_agent(agent)
```

```
100%|██████████| 20000/20000 [00:12<00:00, 1581.78it/s]
Вывод Q-матриц для алгоритма  Двойное Q-обучение
Q1
[[ -13.11064383  -12.50473463  -12.31881091  -13.09260621]
 [ -12.55330099  -12.29483229  -11.55267167  -12.81809758]
 [ -12.16864805  -11.5626184   -10.77286276  -12.19637182]
 [ -11.01574591  -10.83916113   -9.97831115  -11.99993177]
 [ -10.27490026  -10.12073402   -9.27030568  -10.79705471]
 [  -9.20819269   -8.77144163   -8.31294515   -9.95021394]
 [  -7.76088052   -8.13656413   -7.46300672   -8.56091481]
 [  -7.0073083    -6.61066796   -6.07996228   -8.32963404]
 [  -6.01973097   -5.7136453    -5.89437979   -7.35984913]
 [  -5.62514691   -4.94340973   -4.80388799   -6.40580669]
 [  -3.64068085   -3.96191448   -3.86736896   -4.67935218]
 [  -3.31702241   -3.19900324   -2.94031646   -3.13257149]
 [ -13.07551817  -11.55289701  -11.54888054  -12.32087018]
 [ -12.33191629  -10.76970651  -10.76416381  -12.3209877 ]
 [ -11.62067703   -9.97566586   -9.96343246  -11.55371634]
 [ -10.80380784   -9.1622806    -9.14635966  -10.77351854]
 [ -10.02614367   -8.32337049   -8.31261189   -9.97453407]
 [  -9.1960121    -7.46184887   -7.50492677   -9.16580388]
 [  -8.36622911   -6.59378698   -6.59372334   -8.31140502]
 [  -8.19085735   -5.93614759   -5.70788096   -7.52613335]
 [  -6.57238668   -4.80396016   -4.80183      -6.3679334 ]
 [  -5.71991592   -3.88530453   -3.881592     -5.79873478]
 [  -5.25450083   -2.9404       -2.9404       -4.7418316 ]
 [  -4.05359961   -2.92155447   -1.98         -3.85996   ]
 [ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
 [ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
 [ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
 [  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
 [  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
 [  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
 [  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
 [  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
 [  -5.75408672   -3.881592   -111.31790293   -5.70788096]
 [  -4.80396016   -2.9404     -111.31790293   -4.80396016]
 [  -3.881592     -1.98       -111.31790293   -3.881592  ]
 [  -2.9404       -1.98         -1.           -2.9404    ]
 [ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
 [   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.        ]]
Q2
[[ -13.00672197  -12.85562625  -12.31822366  -12.98653985]
 [ -12.58560862  -11.98193082  -11.55128769  -12.85564622]
 [ -11.74295012  -11.44437915  -10.77806669  -12.34219151]
 [ -11.15504139  -11.07694566   -9.97190026  -10.89970469]
 [ -10.42519887   -9.97459944   -9.15217558  -10.94034821]
 [  -9.47607055   -9.38848201   -8.3310929    -9.38565864]
 [  -8.74498542   -7.48558873   -7.45867799   -8.79428811]
 [  -6.93327356   -6.77014736   -7.35354615   -7.91749137]
 [  -6.22959545   -5.70658362   -5.6951064    -6.30380601]
 [  -5.30021374   -5.09932974   -4.80405033   -5.91698547]
 [  -4.40851237   -3.7878874    -3.8830317    -3.72480616]
 [  -3.55586318   -3.46835935   -2.94073107   -4.13354015]
 [ -13.07568113  -11.55662941  -11.54888054  -12.32665461]
 [ -12.34726403  -10.76892194  -10.76416381  -12.31975769]
 [ -11.59258192   -9.97795505   -9.96343246  -11.55339734]
 [ -10.89986119   -9.16171102   -9.14635966  -10.77108641]
 [ -10.1935293    -8.3206789    -8.31261189   -9.97065441]
 [  -9.23585141   -7.46184887   -7.46592071   -9.1484976 ]
 [  -8.32217711   -6.60006691   -6.59372334   -8.31404777]
 [  -7.50663346   -5.71321225   -5.70788096   -7.28760956]
 [  -6.60044723   -4.80396016   -4.85103277   -6.70366026]
 [  -5.68857844   -3.88471108   -3.881592     -5.70590582]
 [  -5.04094779   -2.9404       -2.9404       -4.75857573]
 [  -3.86034743   -2.89827616   -1.98         -3.87416311]
 [ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
 [ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
 [ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
 [  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
 [  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
 [  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
 [  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
 [  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
 [  -5.70788096   -3.881592   -111.31790293   -5.70788096]
 [  -4.80396016   -2.9404     -111.31790293   -4.80396016]
 [  -3.881592     -1.98       -111.31790293   -3.881592  ]
 [  -2.9404       -1.98         -1.           -2.9404    ]
 [ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
 [   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.        ]]
```




```
Action Space: Discrete(4)
Observation Space: Discrete(48)
Max Episode Steps: None
Nondeterministic: False
Reward Range: (-inf, inf)
Reward Threshold: None
Moviepy - Building video /content/video/CliffWalking-v0/fd24cdc1-bde8-4623-8962-e879695b0be2/rl-video-step-0.mp4.
Moviepy - Writing video /content/video/CliffWalking-v0/fd24cdc1-bde8-4623-8962-e879695b0be2/rl-video-step-0.mp4

Moviepy - Done !
Moviepy - video ready /content/video/CliffWalking-v0/fd24cdc1-bde8-4623-8962-e879695b0be2/rl-video-step-0.mp4
```

Вывод

В ходе выполнения лабораторной работы ознакомился с тремя алгоритмами обучения с подкреплением: SARSA, Q-Learning и Double Q-Learning.

Реализовал эти алгоритмы в коде и провели их сравнение на задаче "обрыва" (CliffWalking-v0). Наблюдал, как агенты, обученные с помощью каждого из алгоритмов, справляются с задачей. Сравнил эффективность обучения, изучив графики накопленной награды за каждый эпизод для каждого алгоритма. В итоге получил практическое представление о работе алгоритмов обучения с подкреплением, их отличиях и о том, как они справляются с задачей "обрыва".