Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**«Методы машинного обучения в автоматизированных системах обработки информации и управления»**

**Рубежный контроль №2**

**«Методы обучения с подкреплением»**

**ИСПОЛНИТЕЛЬ:**

Демирев Н.К.

Группа ИУ5-21М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2023 г.

Москва 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.).

## Листинг

### BasicAgent.py

import numpy as np

import plotly.express as px

import pandas as pd

import os

import pygame

os.environ['SDL\_VIDEODRIVER']='dummy'

pygame.display.set\_mode((640,480))

class BasicAgent:

    ALGO\_NAME = 'Base'

    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 = []

        self.all\_reward = 0

    def print\_q(self):

        self.all\_reward = np.sum(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):

        y = self.episodes\_reward

        df = pd.DataFrame(data={

            'Номер эпизода': list(range(1, len(y)+1)),

            'Награда': y

        })

        fig = px.line(df, x="Номер эпизода", y="Награда", title='Награды по эпизодам', height=400, width=600)

        fig.show()

    def learn(self):

        pass

### SARSA\_Agent.py

import os

import pygame

from BasicAgent import BasicAgent

os.environ['SDL\_VIDEODRIVER']='dummy'

pygame.display.set\_mode((640,480))

class SARSA\_Agent(BasicAgent):

    ALGO\_NAME = 'SARSA'

    def \_\_init\_\_(self, env, eps=0.4, lr=0.1, gamma=0.98):

        super().\_\_init\_\_(env, eps)

        self.lr=lr

        self.gamma = gamma

        self.eps\_decay=0.00005

        self.eps\_threshold=0.01

    def learn(self, num\_episodes=20000):

        self.episodes\_reward = []

        for ep in list(range(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)

### main.py

import gymnasium as gym

import os

import pygame

from tabulate import tabulate

import time

import numpy as np

from tqdm import tqdm

from SARSA\_Agent import SARSA\_Agent

os.environ['SDL\_VIDEODRIVER']='dummy'

pygame.display.set\_mode((640,480))

def run\_sarsa():

    all\_rewards = []

    parameters = []

    lr\_list = np.linspace(0.0005, 0.005, num=5)

    gamma\_list = np.linspace(0.9, 1, num=5)

    eps\_list = np.linspace(0.05, 0.9, num=9)

    env = gym.make('Taxi-v3')

    for lr in tqdm(lr\_list, bar\_format=' {l\_bar}{bar:20}{r\_bar}{bar:-10b}', colour='CYAN'):

        for gamma in gamma\_list:

            for ep in eps\_list:

                agent = SARSA\_Agent(env, lr=lr, gamma=gamma, eps=ep)

                agent.learn(100)

                agent.print\_q()

                all\_rewards.append(agent.all\_reward)

                parameters.append([lr, gamma, ep])

    return all\_rewards, parameters

def main():

    all\_rewards, parameters = run\_sarsa()

    print(tabulate(

        {

            'Максимальная награда:' : [np.max(all\_rewards)],

            'Значения гиперпараметров' : parameters[np.argmax(np.max(all\_rewards))]

        },

        headers='keys',

        tablefmt='psql'))

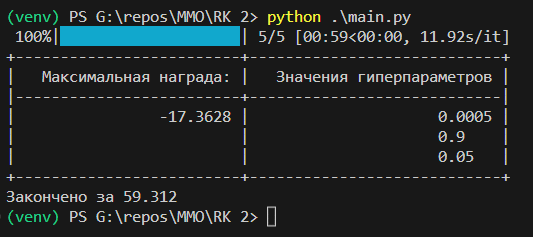
    print(f"Закончено за {time.process\_time():.3f}")

if \_\_name\_\_ == '\_\_main\_\_':

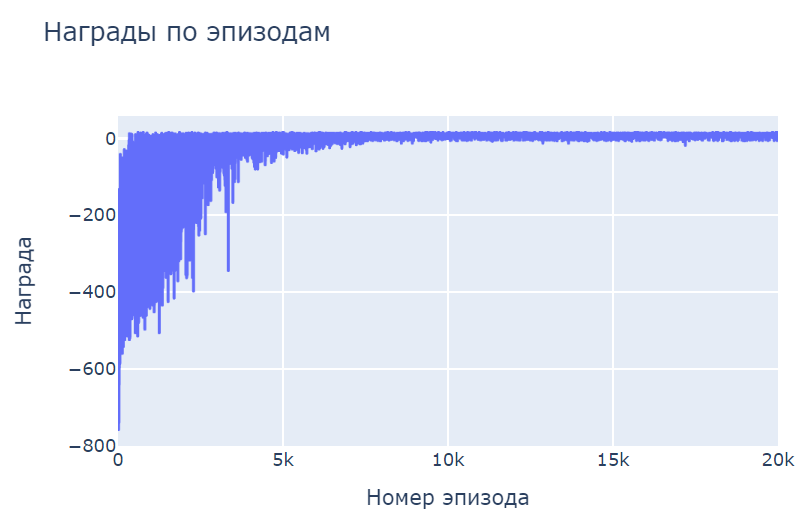
    main()

## Подбор гиперпараметров

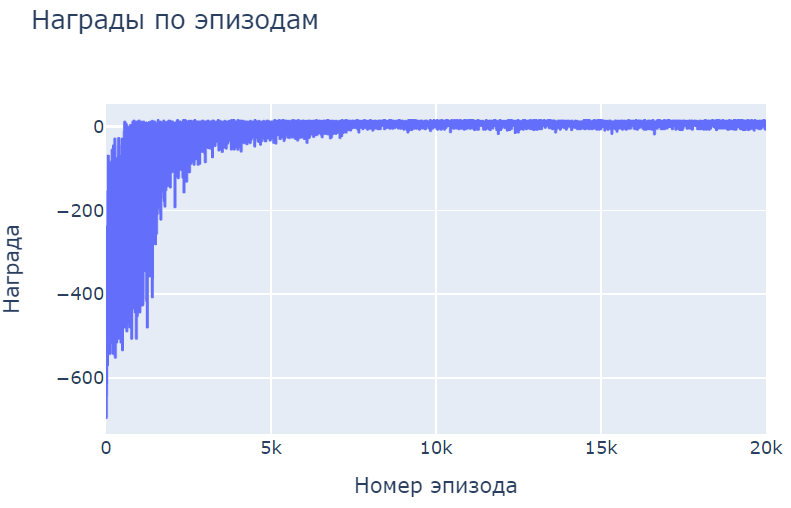
* lr от 0.0005 до 0.005 – 5 значений с равным шагом
* Gamma от 0.9 до 1 – 5 значений с равным шагом
* Eps от 0.05 до 0.9 – 9 значений с равным шагом



### Q-обучение



### SARSA



### Двойное Q-обучение

