Рубежный контроль №2
ИУ5-24М
Желтова Александра

Тема: Методы обучения с подкреплением. Для одного из алгоритмов временных различий, реализованных Вами в соответствующей лабораторная работе:

SARSA

осуществите подбор гиперпараметров. Критерием оптимизации должна являться суммарная награда.

Среда обучения с подкреплением Cliff Walking

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


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 ', 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()
```

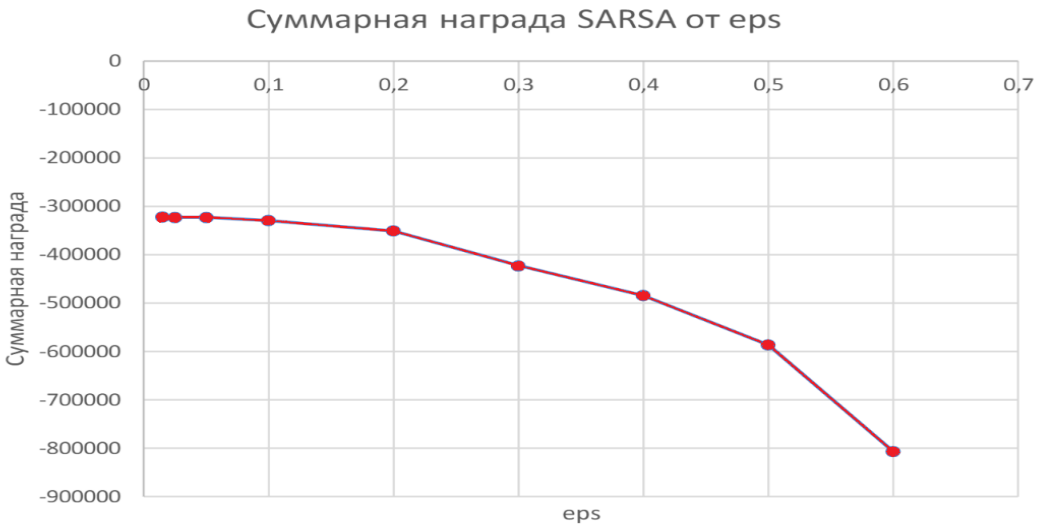
```
    else:
      return self.greedy(state)

  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='blue')
    plt.title('Награды по эпизодам' )
    plt.xlabel('Номер эпизода')
    plt.ylabel( 'Награда')
    plt.show()

  def learn():
    pass


class SARSAgent(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):
        self.Q[state][action] = self.Q[state][action] + self.lr * (rew + self.gamma *
self.Q[next_state][next_action] -self.Q[state][action])
        next_state, rew, done, truncated, _ = self.env.step(action)
        next_action = self.make_action(next_state)
        state = next_state
        action = next_action
        tot_rew += rew
      return np.max(episodes_reward)

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

  def print_q(self):
    print('Вывод Q-матриц', self.ALGO_NAME)
```

```
    print('Q1')
    print(self.Q)


def play_agent(agent):
  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
    tot_rew +=reward

def run_sarsa():
    env = gym.make('CliffWalking-v0')
    agent = SARSAgent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent (agent)

def main():
  run_sarsa()
  print (np.max(rewards))

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

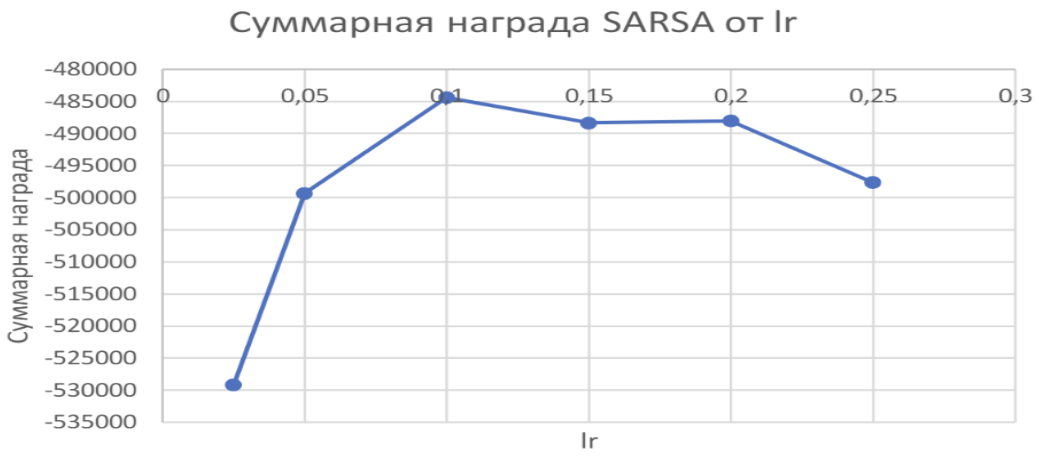
Подбор параметров:

Начальные гиперпараметры: eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000

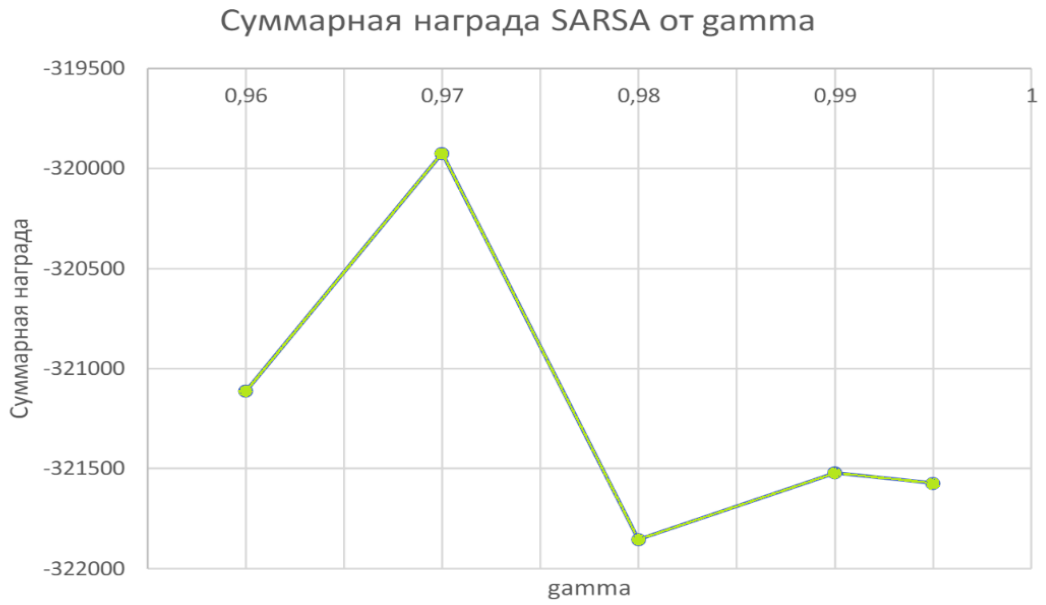
Изменим параметр eps и посмотрим зависимость суммарной награды от eps:



Изменим скорость обучения и посмотрим зависимость суммарной награды от скорости обучения:



Изменим параметр gamma и посмотрим зависимость суммарной награды от gamma:



Вывод: исходя из проверки зависимости подобранных гиперпараметров между собой можно выделить следующие наилучшие значения:

eps~0.02, lr=0.1, gamma=0.97, num_episodes =20000.

Стоит отметить, что для SARS-алгоритма уменьшение eps позволяет его ускорить.