

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 лабораторной работе №5: «Обучение на основе временных различий»

Выполнила:

студентка группы ИУ5-24М

Мащенко Е. И.

Проверил:

Балашов А.М.

Цель 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).

Выполнение работы

Для реализации алгоритмов SARSA, Q-обучение и двойное Q-обучение была выбрана среда обучения Cliff Walking из библиотеки gym. Согласно документации, агент может находиться в 48 состояниях и осуществлять 4 действия.

Лабораторная работа №5

from tqdm import tqdm

B [1]: ▶ ! pip install gymnasium Requirement already satisfied: gymnasium in c:\users\user\appdata\local\programs\python\python38\lib\site-packages (0.28.1) Requirement already satisfied: typing-extensions>=4.3.0 in c:\users\user\appdata\local\programs\python\python38\lib\site-pac kages (from gymnasium) (4.6.3)Requirement already satisfied: jax-jumpy>=1.0.0 in c:\users\user\appdata\local\programs\python\python38\lib\site-packages (f rom gymnasium) (1.0.0) Requirement already satisfied: numpy>=1.21.0 in c:\users\user\appdata\local\programs\python\python38\lib\site-packages (from gymnasium) (1.24.3) Requirement already satisfied: importlib-metadata>=4.8.0; python_version < "3.10" in c:\users\user\appdata\local\programs\py (from gymnasium) (1.3.0) Requirement already satisfied: farama-notifications>=0.0.1 in c:\users\user\appdata\local\programs\python\python38\lib\sitepackages (from gymnasium) (0.0.4) packages (from gymnasium) (0.0.4) Requirement already satisfied: zipp>=0.5 in c:\user\user\appdata\local\programs\python\python38\lib\site-packages (from importlib-metadata>=4.8.0; python_version < "3.10"->gymnasium) (3.15.0) WARNING: You are using pip version 20.1.1; however, version 23.1.2 is available. You should consider upgrading via the 'c:\users\user\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command. B [14]: ▶ import gymnasium as gym import numpy as np
from pprint import pprint import matplotlib.pyplot as plt

```
class BasicAgent:
                   Базовый агент, от которого наследуются стратегии обучения
                    # Наименование алгоритма
                   ALGO_NAME = '---'
                   def __init__(self, env, eps=0.1):
    # Cpeda
                         self.env = env
                         # Размерности О-матрицы
                         self.nA = env.action_space.n
self.nS = env.observation_space.n
                         #u cama mampuqa
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:
                             ч Если состояние вернулось с виде кортежа, то вернуть только номер состояния return state[0]
                             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()
                             # иначе действие, соответствующее максимальному Q-значению return self.greedy(state)
                    def draw_episodes_reward(self):
# Построение графика наград по эпизодам
                        # Построение графика наград по эпизодам fig, ах = 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') slt.sbev('
                         plt.show()
                    def learn():
                         Реализация алгоритма обучения
                         pass
```

```
def learn(self):
    Обучение на основе алгоритма SARSA
    self.episodes_reward = []
     # Цикл по эпизодам
    for ep in tqdm(list(range(self.num_episodes))):
         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)
```

```
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 уменьшени
self.eps_decay=0.00005
                        self.eps_threshold=0.01
                   def learn(self):
                        Обучение на основе алгоритма Q-Learning
                        self.episodes_reward = []
                        for ep in tqdm(list(range(self.num_episodes))):
                             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 вывора 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.0[state][action] = self.0[state][action] + self.Lr * \
# (rew + self.gamma * self.0[next_state][next_action] - self.0[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)
```

```
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)
                     self.Q2 = np.zeros((self.nS, self.nA))
                     # Learning rate
self.lr=lr
                     # Коэффициент дисконтирования
                    self.gamma = gamma
# Konuyecm6o ənusodo6
                     self.num_episodes=num_episodes
                     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(self.Q2)
                 def learn(self):
                     Обучение на основе алгоритма Double Q-Learning
                     self.episodes_reward = []
                     for ep in tqdm(list(range(self.num_episodes))):
                          state = self.get_state(self.env.reset())
                          # Флаг штатного завершения эпизода
                          done = False
                          # Флаг нештат
                                          ого завершения эпизода
                          truncated = False
                         # Суммарная награда по эпизоду
tot_rew = 0
                          # По мере заполнения О-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:</pre>
                                   # 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])
                                  ** Обновление второй таблицы
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)
```

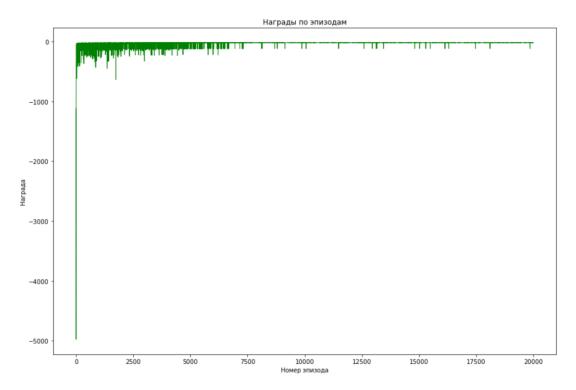
```
B [7]: ▶ !pip install pygame
                  Requirement already satisfied: pygame in c:\user\user\appdata\local\programs\python\python38\lib\site-packages (2.4.0)
                 WARNING: You are using pip version 20.1.1; however, version 23.1.2 is available.

You should consider upgrading via the 'c:\users\user\appdata\local\programs\python\python38\python.exe -m pip install --upgr
                  ade pip' command.
  B [9]: M 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)
                            action = agent.greedy(state)
next_state, reward, terminated, truncated, _ = env2.step(action)
env2.render()
state = next_state
if terminated or truncated:
                                  done = True
agent.learn()
                       agent.print_q()
agent.draw_episodes_reward()
                       play_agent(agent)
                 def run_q_learning():
    env = gym.make('CliffWalking-v0')
    agent = Qlearning_Agent(env)
    agent.learn()
    agent.print_q()
                       agent.draw_episodes_reward()
play_agent(agent)
                 def run_double_q_learning():
    env = gym.make('CliffWalking-v0')
    agent = DoubleQLearning_Agent(env)
```

agent.learn()
agent.print_q()
agent.draw_episodes_reward()
play_agent(agent)

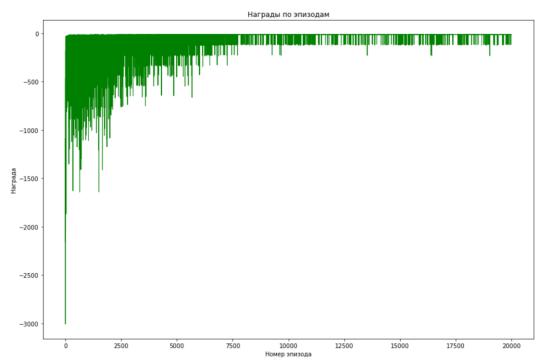
Результаты выполнения алгоритма SARSA:

_	_				
		для алгоритма	SARSA		
[[-13.22476858	-12.40212943	-14.13351968	-13.247598	
[-12.46633265	-11.60886082	-13.25961018	-13.396148	-
[-11.68673114	-10.82536428	-12.54205618	-12.710645	-
[-10.91088291	-10.03473341	-11.97256549	-11.856415	_
[-10.05443604	-9.22075139	-10.98368409	-11.055003	
[-9.23314151	-8.3810884	-10.22364864	-10.263065	_
[-8.39169009	-7.51667793	-9.32313527	-9.474565	
[-7.75986635	-6.62958223	-8.43696967	-8.607948	
[-6.6798499	-5.72368267	-6.28731379	-7.755576	66]
[-5.78921951	-4.80705929	-5.21929322	-6.864956	61]
[-4.8399588	-3.90070344	-4.15689719	-5.960879	31]
[-3.90460483	-3.93017388	-3.10931157	-5.049847	93]
]	-13.20842669	-14.50896221	-14.88832524	-14.035093	91]
Ī	-12.44366262	-15.77727851	-34.78649623	-17.205087	16]
]	-11.6689988	-17.0636726	-36.88351869	-21.570874	83]
Ĩ	-10.90885586	-15.99412427	-34.34885333	-19.292791	13]
į	-10.06685617	-14.18469097	-34.83027979	-18.154640	52]
į	-9.27321473	-13.25742295	-17.5419469	-15.478550	89]
į	-8.39728543	-12.07373043	-15.57379833	-15.092401	891
į	-7.52746239	-11.8302303	-16.99889938	-12.903029	09 Î
į	-9.86518267	-4.91969819	-29.37857346	-11.212914	791
į	-7.26189146	-3.92794254	-13.07460017	-8.213406	
į	-5.51632757	-2.94596416	-4.30541736	-5.657224	
į	-4.1042407	-2.98318162	-1.9800118	-4.026898	
į	-13.98414661	-14.18701482	-15.72825592	-14.706064	
į	-13.23945739	-27.93025716	-127.9352286	-22.180688	-
į	-19.49136846	-30.88756084	-125.80338836	-25.992329	-
į	-15.95913324	-26.23852616	-127.82948837	-21.465040	-
į	-16.13513697	-30.56279977	-120.8398651	-25.525091	-
į	-14.2020161	-19.38864097	-114.18258746	-23.409785	
į	-12.77299937	-21.80720316	-100.94675116	-38.871388	-
ŗ	-14.71866128	-14.51043024	-114.02839452	-21.794676	-
į	-11.79133711	-23.13539561	-107.41719746	-21.428721	-
į	-8.01059803	-20.72081699	-119.6939017	-22.234732	
į	-6.12954221	-2.01736421	-123.35381298	-17.903164	-
į	-3.16618017	-1.98319686	-1.	-3.043613	-
į	-14.74045584	-114.60629757	-15.45143492	-15.402768	
į	0.	0.	0.	0.	1
į	0.	0.	0.	0.	j
į	0.	0.	0.	0.	j
į	0.	0.	0.	0.	j
į	0.	0.	0.	0.	j
[0.	0.	0.	0.]
[0.	0.	0.	0.	j
[0.	0.	0.	0.]
Ĺ	0.	0.	0.	0.	i
[0.	0.	0.	0.]
Ĺ	0.	0.	0.	0.	11
L	٠.	٠.	٠.	٠.	11



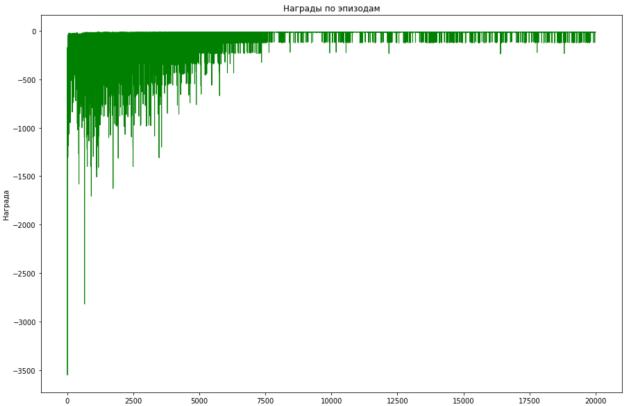
Результаты выполнения алгоритма Q-обучение:

Вывод О-матрицы	лля алгоритма	Q-обучение		
[[-12.56014099	-12.30862203	-12.30813032	-12.5182381	21
[-12.09782015	-11.54860205	-11.54857925	-12.5039394	
[-11.43275066	-10.76414036	-10.76414006	-12.1685764	-
[-10.68119855	-9.96343009	-9.96342995	-11.5031827	-
[-9.92672148	-9.14635935	-9.14635933	-10.7450809	-
[-9.11663454	-8.31261183	-8.31261184	-9.9036806	
[-8.30222003	-7.46184886	-7.46184886	-9.1191062	-
[-7.45700115	-6.59372334	-6.59372334	-8.3052103	-
[-6.58743985	-5.70788096	-5.70788096	-7.4433366	
[-5.70275479	-4.80396016	-4.80396016	-6.5915416	
[-4.78896046	-3.881592	-3.881592	-5.6551662	
[-3.87388995	-3.85669066	-2.9404	-4.7803537	
[-13.0494886	-11.54888054	-11.54888054	-12.3177979	
[-12.31699985	-10.76416381	-10.76416381	-12.3178627	
[-11.54875704	-9.96343246	-9.96343246	-11.5488795	
[-10.76415471	-9.14635966	-9.14635966	-10.7641637	-
	-8.31261189	-8.31261189	-9.9634324	-
[-9.96343069 [-9.14635932	-7.46184887	-7.46184887	-9.1463596	-
-8.31261176	-6.59372334	-6.59372334	-8.3126118	
[-8.31261176 [-7.46184886	-5.70788096	-5.70788096	-7.4618488	-
	-4.80396016	-4.80396016	-6.5937233	
[-6.59372332 [-5.70788093	-3.881592	-3.881592	-5.7078808	
[-4.80396015	-2.9404	-2.9404	-4.8039601	
[-4.80396015 [-3.881592	-2.9404	-1.98	-3.881592	-
	-10.76416381	-12.31790293	-11.5488805	
	-9.96343246	-111.31790293	-11.5488805	-
[-10.76416381	-9.14635966	-111.31790293	-10.7641638	-
[-9.96343246	-8.31261189	-111.31790293	-9.9634324	
[-9.14635966 [-8.31261189	-7.46184887	-111.31790293	-9.1463596	
[-8.31261189	-6.59372334	-111.31790293	-8.3126118	
[-7.46184887	-5.70788096	-111.31790293	-7.4618488	-
[-6.59372334	-4.80396016	-111.31790293	-6.5937233	-
[-5.70788096	-3.881592	-111.31790293	-5.7078809	
-4.80396016	-2.9404	-111.31790293	-4.8039601	
[-3.881592	-1.98	-111.31790293	-3.881592]
[-2.9404	-1.98	-1.	-2.9404]
	-111.31790293	-12.31790293	-12.3179029	
[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.]]



Результаты выполнения алгоритма двойное Q-обучение:

```
Вывод Q-матриц для алгоритма
                                Двойное Q-обучение
[[ -14.72368332 -14.44139918
                                 -12.32201971
                                                -15.39951456]
                  -13.88274759
                                                 -14.4834763
   -13.84014705
                                  -11.61397923
  -13.01118045
                 -11.69981379
                                 -10.76479727
                                                 -12.89794643]
  -11.2196401
                  -12.03426601
                                 -10.00903399
                                                -12.75786237]
  -10.36020818
                  -9.13769096
                                  -8.09187603
                                                -11.1847185 ]
    -8.89588977
                                  -8.56975067
                   -8.31365248
                                                 -10.16000227]
    -8.08165511
                   -7.46310595
                                   -7.46186547
                                                   -8.99673211]
                   -6.75166673
    -7.04636956
                                  -6.59372208
                                                  -8.056808441
    -6.28626435
                   -5.69877612
                                  -5.7078722
                                                   -7.02035336]
    -5.01945141
                   -4.93186079
                                  -4.93115864
                                                  -5.44802543]
    -4.31967822
                   -3.87330027
                                   -3.72845749
                                                  -4.28719339]
    -3.62824656
                   -3.66345184
                                  -2.94043034
                                                  -3.8814804
   -13.17169285
                 -11.58379149
                                 -11.54888054
                                                -12.32203364]
   -12.39268404
                 -10.79588252
                                 -10.76416381
                                                 -12.33282328]
  -11.8812196
                  -10.18572479
                                  -9.96343246
                                                 -11.57839974]
   -10.74310069
                   -9.15323511
                                   -9.14635966
                                                 -10.76437316]
  -12.3782129
                   -8.33887401
                                   -8.40789865
                                                 -10.01311137]
  [[ -14.72077416
                 -15.0800717
                               -12.33951287 -14.92702132]
     -14.71444435
                  -13.65796101
                               -11.55697814
                                           -14.72033705]
     -12.82335539
                  -13.41573727
                               -10.86946365
                                           -14.9287179
     -11.98949528
                  -10.8853534
                                -9.91980932
                                           -11.46147513
      -9.36302299
                   -9.41417943
                               -11.41533546
                                           -10.16645376]
      -8.79614702
                   -8.3127866
                                -8.24830202
                                           -10.4880784
      -8.21460256
                   -7.61348758
                                -7.46184887
                                            -9.02102836]
      -7.41234814
                   -6.54418667
                               -6.59374686
                                            -7.84301672
      -6.41120013
                   -5.80258036
                                -5.70785737
                                            -7.2587489
      -5.35330181
                   -4.84773028
                               -4.79859502
                                            -4.8826891
      -4.0276935
                   -3.88011614
                                -4.03354006
                                            -4.72920931
      -3.18708222
                   -3.36695608
                               -2.9403798
                                            -3.83303216
                  -11.56699018
     -13,13615202
                               -11.54888054
                                           -12.3330435
                  -10.8054778
     -12.58387206
                               -10.76416381
                                           -12.34855159
     -11.65754981
                   -9.97855505
                                -9.96343246
                                            -11.55181744
     -11.16808309
                   -9.48967342
                                -9.14635966
                                            -10.86120421
     -11.08957629
                   -8.31261813
                                -8.28321184
                                            -9.92502949
      -9.41318239
                   -7.46184887
                                            -9.41597598]
                                -7.4886378
```



Номер эпизода

Пример агента в конечном состоянии:

