Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №4-2
по дисциплине
«Методы машинного обучения»

Выполнил:
студент группыИУ5И-23М

Лю Цзычжан

Москва — 2025 г.

1. Цель лабораторной работы

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

2. 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).
-

3. Ход выполнения работы

3.1. SARSA

```
class SARSA_Agent(BasicAgent):
    ALGO_NAME = 'SARSA'

    def __init__(self, env, eps=0.1, lr=0.01, gamma=0.9, episodes=10000):
        super().__init__(env, eps)
        self.lr = lr
        self.gamma = gamma
        self.episodes = episodes

    def learn(self):
        for _ in tqdm(range(self.episodes)):
            state = self.env.reset()
            tot_rew = 0
            done = False
            while not done:
                action = self.make_action(state)
                results = self.env.step(action)
                if len(results) == 5:
                    next_state, reward, terminated, truncated, _ = results
                    done = terminated or truncated
                else:
                    next_state, reward, done, _ = results
                next_action = self.make_action(next_state)
                self.Q[state][action] += self.lr * (reward + self.gamma * self.Q[next_state][next_action] - self.Q[state][action])
                state = next_state
                tot_rew += reward
            self.episodes_reward.append(tot_rew)
```

3.2. Q-обучение

```
class QLearning_Agent(BasicAgent):
    ALGO_NAME = 'Q-Learning'

    def __init__(self, env, eps=0.1, lr=0.01, gamma=0.9, episodes=10000):
        super().__init__(env, eps)
        self.lr = lr
        self.gamma = gamma
        self.episodes = episodes

    def learn(self):
        for _ in tqdm(range(self.episodes)):
            state = self.env.reset()
            tot_rew = 0
            done = False
            while not done:
                action = self.make_action(state)
                results = self.env.step(action)
                if len(results) == 5:
                    next_state, reward, terminated, truncated, _ = results
                    done = terminated or truncated
                else:
                    next_state, reward, done, _ = results
                self.Q[state][action] += self.lr * (reward + self.gamma * np.max(self.Q[next_state]) - self.Q[state][action])
                state = next_state
                tot_rew += reward
            self.episodes_reward.append(tot_rew)
```

3.3. Двойное Q-обучение

```
class DoubleQLearning_Agent(BasicAgent):
    ALGO_NAME = 'Double Q-Learning'

    def __init__(self, env, eps=0.1, lr=0.01, gamma=0.9, episodes=10000):
        super().__init__(env, eps)
        self.Q2 = np.zeros((self.nS, self.nA))
        self.lr = lr
        self.gamma = gamma
        self.episodes = episodes

    def learn(self):
        for _ in tqdm(range(self.episodes)):
            state = self.env.reset()
            tot_rew = 0
            done = False
            while not done:
                action = self.make_action(state)
                results = self.env.step(action)
                if len(results) == 5:
                    next_state, reward, terminated, truncated, _ = results
                    done = terminated or truncated
                else:
                    next_state, reward, done, _ = results
                if np.random.rand() < 0.5:
                    self.Q[state][action] += self.lr * (reward + self.gamma * self.Q2[next_state][np.argmax(self.Q[next_state])] - self.Q[state][action])
                else:
                    self.Q2[state][action] += self.lr * (reward + self.gamma * self.Q[next_state][np.argmax(self.Q2[next_state])] - self.Q2[state][action])
                state = next_state
                tot_rew += reward
            self.episodes_reward.append(tot_rew)
```

Результат:

```
100%|██████████| 10000/10000 [00:10<00:00, 969.05it/s]
Вывод Q-матрицы для алгоритма Double Q-Learning
[[2.09090553e-03 5.33733616e-04 6.26703234e-04 4.49300743e-04]
 [3.12935469e-04 1.48974999e-08 7.74795377e-09 1.67898370e-06]
 [2.50569007e-07 0.00000000e+00 0.00000000e+00 2.06973157e-06]
 [1.82063050e-10 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.96880081e-03 6.70628072e-04 3.91370857e-04 2.14062950e-04]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.25465774e-04 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [5.42908171e-04 6.22978901e-03 8.01717927e-04 9.90410518e-04]
 [1.32582247e-02 1.62276108e-03 2.53574290e-03 4.91380123e-04]
 [1.66801823e-02 0.00000000e+00 3.11899508e-03 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [3.46738535e-04 6.84192399e-02 4.37174426e-03 3.01393197e-03]
 [3.49830082e-03 1.00000000e-02 1.63234199e-03 1.88758909e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
```