Московский Государственный Технический Университет имени Н.Э.Баумана

Факультет Информатика и системы управления

Кафедра ИУ-5

«Системы обработки информации и управления»

Отчёт по Рубежному Контролю No 2

Методы обработки данных

Выполнили стуленты группы ИУ5И

Шэнь Цюцзе 22М

Москва 2022г.

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. Программа

```
import numpy as np
import gym
import random
import matplotlib.pyplot as plt

env = gym.make("Taxi-v3")
env.reset()
env.render()
action_size = env.action_space.n
print("Action size ", action_size)

state_size = env.observation_space.n
print("State size ", state_size)
qtable = np.zeros((state_size, action_size))
print(qtable)
total_episodes = 5000  # Total episodes
total_test_episodes = 100  # Total test episodes
max_steps = 99  # Max steps per episode

learning_rate = 0.7  # Learning rate
```

```
gamma = 0.618  # Discounting rate

# Exploration parameters
epsilon = 1.0  # Exploration rate
max_epsilon = 1.0  # Exploration probability at start
min_epsilon = 0.01  # Minimum exploration probability
decay_rate = 0.01  # Exponential decay rate for exploration prob

# Tracking metrics
rewards_per_episode = []  # List to store rewards per episode

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0  # Total rewards accumulated in the episode

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking
the biggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state, :])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward
(r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') -
Q(s,a)]
        qtable[state, action] = qtable[state, action] + learning_rate * (
                    reward + gamma * np.max(qtable[new_state, :]) -
qtable[state, action])

        total_rewards += reward  # Accumulate the rewards
```

```
        # Our new state is state
        state = new_state

        # If done: finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon) * np.exp(-
decay_rate * episode)

    rewards_per_episode.append(total_rewards)  # Append the total rewards
for this episode to the list

    # Print progress
    if (episode + 1) % 1000 == 0:
        print(f"Episode: {episode+1}/{total_episodes}")

env.reset()
test_rewards = []

for episode in range(total_test_episodes):
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # Take the action (index) that has the maximum expected future
reward given that state
        action = np.argmax(qtable[state, :])

        new_state, reward, done, info = env.step(action)

        total_rewards += reward

        if done:
            test_rewards.append(total_rewards)
            break
        state = new_state

# Calculate and print the average reward per test episode
avg_test_reward = np.mean(test_rewards)
print("Average test reward:", avg_test_reward)
```

```
# Plot rewards per episode
plt.plot(rewards_per_episode)
plt.xlabel("Episode")
plt.ylabel("Total Reward")
plt.title("Total Reward per Episode")
plt.show()
```

4. результат

```
Episode: 1000/5000
Episode: 2000/5000
Episode: 3000/5000
Episode: 4000/5000
Episode: 5000/5000
Average test reward: 8.03
```