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)

ФАКУЛЬТЕТ Информатика и системы управления и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

Лабораторная работа №4 по курсу «Методы машинного обучения в автоматизированных системах обработки информации и управления»

Подготовили:

У Жун

ИУ5И-25М

25.05.2024

Проверил:

Гапанюк Ю. Е.

2024 г.

Цель лабораторной работы:

ознакомление с базовыми методами обучения с подкреплением.

Задание:

На основе рассмотренного на лекции примера реализуйте алгоритм Policy Iteration для любой среды обучения с подкреплением (кроме рассмотренной на лекции среды Toy Text / Frozen Lake) из библиотеки Gym (или аналогичной библиотеки).

Алгоритм итерации стратегии состоит из двух основных частей: оценка политики (policy evaluation) и улучшение политики (policy improvement).

Выбор окружения: Я выбрала окружение CartPole из библиотеки Gym.

Дискретизация состояний: я дискретизировала пространство состояний среды CartPole, чтобы использовать табличный метод для хранения функций значений и политик.

```
In [1]: pip install gym numpy
```

```
Requirement already satisfied: gym in d:\anaconda3\lib\site-packages (0.26.2)
Requirement already satisfied: numpy in d:\anaconda3\lib\site-packages (1.24.3)
Requirement already satisfied: cloudpickle>=1.2.0 in d:\anaconda3\lib\site-packages (from gym) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in d:\anaconda3\lib\site-packages (from gym) (0.0.8)
Note: you may need to restart the kernel to use updated packages.
```

```
In [4]: pip install pygame
```

```
Collecting pygame
  Obtaining dependency information for pygame from https://files.pythonhosted.org/packages/82/61/93ae7afbd931a70510cfdf0a7bb0007540020b8d80bc1d8762ebdc46479b/pygame-2.5.2-cp311-cp311-win_amd64.whl.metadata
  Downloading pygame-2.5.2-cp311-cp311-win_amd64.whl.metadata (13 kB)
Downloading pygame-2.5.2-cp311-cp311-win_amd64.whl (10.8 MB)
   ---------------------------------------- 0.0/10.8 MB ? eta -:--:--
   ---------------------------------------- 0.0/10.8 MB ? eta -:--:--
   ---------------------------------------- 0.0/10.8 MB ? eta -:--:--
   ---------------------------------------- 0.0/10.8 MB 162.5 kB/s eta 0:01:07
   ---------------------------------------- 0.0/10.8 MB 162.5 kB/s eta 0:01:07
   ---------------------------------------- 0.0/10.8 MB 162.5 kB/s eta 0:01:07
   ---------------------------------------- 0.0/10.8 MB 108.9 kB/s eta 0:01:39
   ---------------------------------------- 0.0/10.8 MB 108.9 kB/s eta 0:01:39
   ---------------------------------------- 0.0/10.8 MB 115.5 kB/s eta 0:01:33
   ---------------------------------------- 0.0/10.8 MB 115.5 kB/s eta 0:01:33
   ---------------------------------------- 0.0/10.8 MB 115.5 kB/s eta 0:01:33
   ---------------------------------------- 0.0/10.8 MB 115.5 kB/s eta 0:01:33
   ---------------------------------------- 0.0/10.8 MB 115.5 kB/s eta 0:01:33
   ---------------------------------------- 0.0/10.8 MB 115.5 kB/s eta 0:01:33
```

```
:
import gym
import numpy as np

# Создание среды
env = gym.make('CartPole-v1')
env.reset()

n_actions = env.action_space.n
state_shape = env.observation_space.shape
n_states = 10

state_bins = [np.linspace(-4.8, 4.8, n_states),
              np.linspace(-5, 5, n_states),
              np.linspace(-0.418, 0.418, n_states),
              np.linspace(-5, 5, n_states)]

def discretize_state(state):
    state_idx = []
    for i in range(len(state)):
        state_idx.append(np.digitize(state[i], state_bins[i]) - 1)
    return tuple(state_idx)

policy = np.random.choice(n_actions, size=(n_states, n_states, n_states, n_states))
value_function = np.zeros((n_states, n_states, n_states, n_states))

gamma = 0.99
theta = 1e-6
max_iterations = 1000
```

```python
def policy_evaluation(policy, value_function, gamma, theta):
    while True:
        delta = 0
        for s1 in range(n_states):
            for s2 in range(n_states):
                for s3 in range(n_states):
                    for s4 in range(n_states):
                        v = value_function[s1, s2, s3, s4]
                        state = (s1, s2, s3, s4)
                        action = policy[state]
                        env.state = [state_bins[i][state[i]] for i in range(4)]
                        next_state, reward, done, truncated, _ = env.step(action)
                        next_state = discretize_state(next_state)
                        if done or truncated:
                            value_function[state] = reward
                        else:
                            value_function[state] = reward + gamma * value_function[next_state]
                        delta = max(delta, abs(v - value_function[state]))
        if delta < theta:
            break
    return value_function
```

```python
def policy_improvement(policy, value_function, gamma):
    policy_stable = True
    for s1 in range(n_states):
        for s2 in range(n_states):
            for s3 in range(n_states):
                for s4 in range(n_states):
                    state = (s1, s2, s3, s4)
                    old_action = policy[state]
                    action_values = np.zeros(n_actions)
                    for action in range(n_actions):
                        env.state = [state_bins[i][state[i]] for i in range(4)]
                        next_state, reward, done, truncated, _ = env.step(action)
                        next_state = discretize_state(next_state)
                        if done or truncated:
                            action_values[action] = reward
                        else:
                            action_values[action] = reward + gamma * value_function[next_state]
                    policy[state] = np.argmax(action_values)
                    if old_action != policy[state]:
                        policy_stable = False
    return policy, policy_stable
```

```python
for i in range(max_iterations):
    value_function = policy_evaluation(policy, value_function, gamma, theta)
    policy, policy_stable = policy_improvement(policy, value_function, gamma)
    if policy_stable:
        print(f"Политика стабилизировалась на итерации {i+1}")
        break

env = gym.make('CartPole-v1', render_mode='human')
state, _ = env.reset()
env.render()
for _ in range(1000):
    state_idx = discretize_state(state)
    action = policy[state_idx]
    next_state, reward, done, truncated, _ = env.step(action)
    state = next_state
    env.render()
    if done or truncated:
        state, _ = env.reset()

env.close()
```

```
Политика стабилизировалась на итерации 2
```

Рис.1-агент был обучен с помощью алгоритма итерации политик

появление окна визуализации среды CartPole и стабильное выполнение агента в этой среде означает, что задание успешно выполнено.

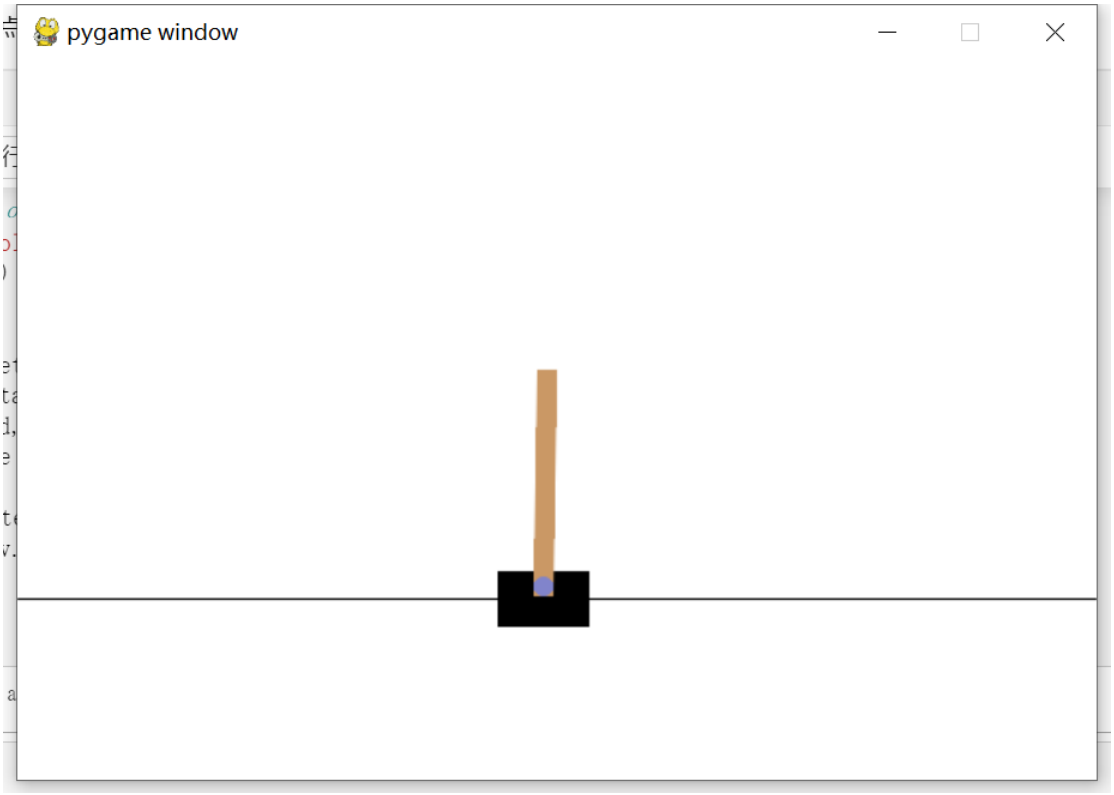

Рис. 2 – Окно визуализации среды CartPole