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 АСОИУ»

«Реализация алгоритма Policy Iteration»

Выполнил:

ИУ5-22М Киричков Е. Е.

20.05.2024

Проверил:

Балашов А.М.

Москва, 2024

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).

Программа предназначена для демонстрации работы агента, обученного с использованием алгоритма итерации по стратегиям (Policy Iteration) в среде CliffWalking из библиотеки OpenAI Gym. Среда CliffWalking представляет собой сетку размером 4x12, где агент должен пройти от начальной клетки до целевой, избегая падения с обрыва.

Ввод [1]:
```
! pip install gym
! pip install pygame
```

```
Requirement already satisfied: gym in /Users/evseykirichkov/anaconda3/lib/
python3.11/site-packages (0.26.2)
Requirement already satisfied: numpy>=1.18.0 in /Users/evseykirichkov/anac
onda3/lib/python3.11/site-packages (from gym) (1.26.4)
Requirement already satisfied: cloudpickle>=1.2.0 in /Users/evseykirichko
v/anaconda3/lib/python3.11/site-packages (from gym) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in /Users/evseykirichko
v/anaconda3/lib/python3.11/site-packages (from gym) (0.0.8)
Requirement already satisfied: pygame in /Users/evseykirichkov/anaconda3/l
ib/python3.11/site-packages (2.5.2)
```

Ввод [7]:
```
import gym
import numpy as np
import matplotlib.pyplot as plt
from pprint import pprint
from IPython.display import Image
```

Ввод [3]:

```
class PolicyIterationAgent:
    '''
    Класс, эмулирующий работу агента
    '''

    def __init__(self, env):
        self.env = env
        # Пространство состояний
        self.observation_dim = 48
        # Массив действий в соответствии с документацией
        # https://www.gymlibrary.dev/environments/toy_text/cliff_walking/
        self.actions_variants = np.array([0, 1, 2, 3])
        # Задание стратегии (политики)
        # Карта 4x12 и 4 возможных действия
        self.policy_probs = np.full((self.observation_dim, len(self.actions
        # Начальные значения для v(s)
        self.state_values = np.zeros(shape=(self.observation_dim))
        # Начальные значения параметров
        self.maxNumberOfIterations = 1000
        self.theta = 1e-6
        self.gamma = 0.99

    def print_policy(self):
        '''
        Вывод матриц стратегии
        '''
        print('Стратегия:')
        pprint(self.policy_probs)

    def policy_evaluation(self):
        '''
        Оценивание стратегии
        '''
        # Предыдущее значение функции ценности
        valueFunctionVector = self.state_values
        for iterations in range(self.maxNumberOfIterations):
            # Новое значение функции ценности
            valueFunctionVectorNextIteration = np.zeros(shape=(self.observa
            # Цикл по состояниям
            for state in range(self.observation_dim):
                # Вероятности действий
                action_probabilities = self.policy_probs[state]
                # Цикл по действиям
                outerSum = 0
                for action, prob in enumerate(action_probabilities):
                    innerSum = 0
                    # Цикл по вероятностям действий
                    for probability, next_state, reward, isTerminalState in
                        innerSum = innerSum + probability * (reward + self.
                    outerSum = outerSum + self.policy_probs[state][action]
                valueFunctionVectorNextIteration[state] = outerSum
            if (np.max(np.abs(valueFunctionVectorNextIteration - valueFunct
                # Проверка сходимости алгоритма
                valueFunctionVector = valueFunctionVectorNextIteration
                break
            valueFunctionVector = valueFunctionVectorNextIteration
        return valueFunctionVector

    def policy_improvement(self):
        '''
        Улучшение стратегии
        '''
        qvaluesMatrix = np.zeros((self.observation_dim, len(self.actions_va
```

```
        improvedPolicy = np.zeros((self.observation_dim, len(self.actions_v
        # Цикл по состояниям
        for state in range(self.observation_dim):
            for action in range(len(self.actions_variants)):
                for probability, next_state, reward, isTerminalState in sel
                    qvaluesMatrix[state, action] = qvaluesMatrix[state, act
                                reward + self.gamma * self.state_values[nex

            # Находим лучшие индексы
            bestActionIndex = np.where(qvaluesMatrix[state, :] == np.max(qv
            # Обновление стратегии
            improvedPolicy[state, bestActionIndex] = 1 / np.size(bestAction
        return improvedPolicy

    def policy_iteration(self, cnt):
        '''
        Основная реализация алгоритма
        '''
        policy_stable = False
        for i in range(1, cnt + 1):
            self.state_values = self.policy_evaluation()
            self.policy_probs = self.policy_improvement()
        print(f'Алгоритм выполнился за {i} шагов.')
```

Ввод [4]:

```
def play_agent(agent):
    env2 = gym.make('CliffWalking-v0', render_mode='human')
    state = env2.reset()[0]
    done = False
    while not done:
        p = agent.policy_probs[state]
        if isinstance(p, np.ndarray):
            action = np.random.choice(len(agent.actions_variants), p=p)
        else:
            action = p
        next_state, reward, terminated, truncated, _ = env2.step(action)
        env2.render()
        state = next_state
        if terminated or truncated:
            done = True
```

Ввод [5]:

```
def main():
    # Создание среды
    env = gym.make('CliffWalking-v0')
    env.reset()
    # Обучение агента
    agent = PolicyIterationAgent(env)
    agent.print_policy()
    agent.policy_iteration(1000)
    agent.print_policy()
    # Проигрывание сцены для обученного агента
    play_agent(agent)
```

Ввод [6]:
```
if __name__ == '__main__':
    main()
```

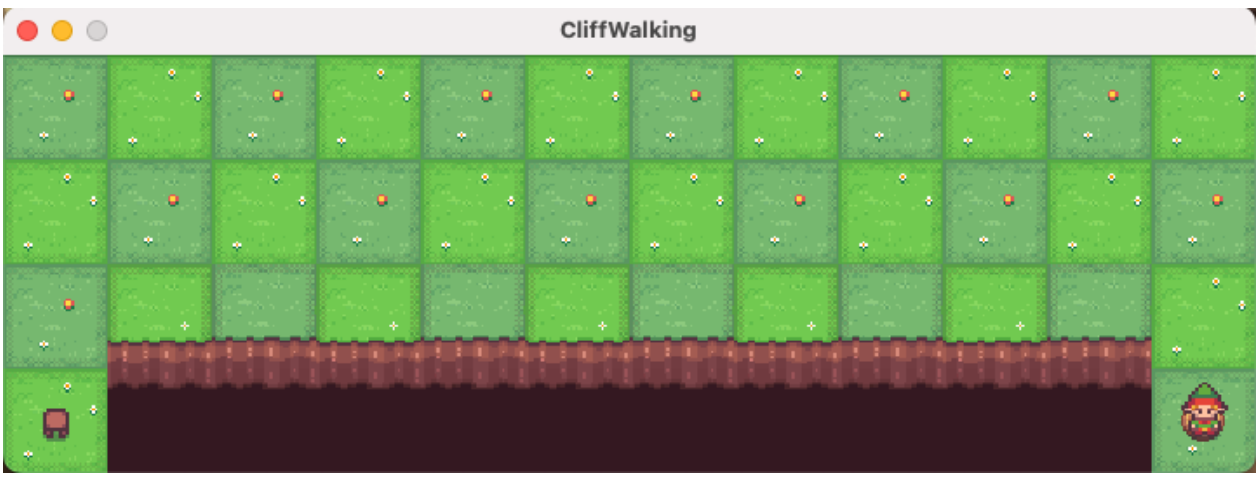
```
Стратегия:
array([[0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25],
       [0.25, 0.25, 0.25, 0.25]])
Алгоритм выполнился за 1000 шагов.
Стратегия:
array([[0.        , 0.5       , 0.5       , 0.        ],
       [0.33333333, 0.33333333, 0.33333333, 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.33333333, 0.        , 0.33333333, 0.33333333],
       [0.        , 0.        , 0.5       , 0.5       ],
```

```
[0.        , 0.        , 1.        , 0.        ],
[0.        , 0.5       , 0.5       , 0.        ],
[0.        , 0.5       , 0.5       , 0.        ],
[0.        , 0.33333333, 0.33333333, 0.33333333],
[0.        , 0.33333333, 0.33333333, 0.33333333],
[0.        , 0.33333333, 0.33333333, 0.33333333],
[0.        , 0.33333333, 0.33333333, 0.33333333],
[0.        , 0.33333333, 0.33333333, 0.33333333],
[0.        , 0.33333333, 0.33333333, 0.33333333],
[0.        , 0.        , 0.5       , 0.5       ],
[0.        , 0.        , 0.5       , 0.5       ],
[0.        , 0.        , 1.        , 0.        ],
[0.        , 0.33333333, 0.33333333, 0.33333333],
[0.        , 0.5       , 0.        , 0.5       ],
[0.33333333, 0.33333333, 0.        , 0.33333333],
[0.33333333, 0.33333333, 0.        , 0.33333333],
[0.33333333, 0.33333333, 0.        , 0.33333333],
[0.33333333, 0.33333333, 0.        , 0.33333333],
[0.33333333, 0.33333333, 0.        , 0.33333333],
[0.33333333, 0.33333333, 0.        , 0.33333333],
[0.33333333, 0.33333333, 0.        , 0.33333333],
[0.33333333, 0.33333333, 0.        , 0.33333333],
[0.        , 0.5       , 0.        , 0.5       ],
[0.        , 0.33333333, 0.33333333, 0.33333333],
[0.33333333, 0.        , 0.33333333, 0.33333333],
[0.5       , 0.        , 0.        , 0.5       ],
[1.        , 0.        , 0.        , 0.        ],
[1.        , 0.        , 0.        , 0.        ],
[1.        , 0.        , 0.        , 0.        ],
[1.        , 0.        , 0.        , 0.        ],
[1.        , 0.        , 0.        , 0.        ],
[1.        , 0.        , 0.        , 0.        ],
[1.        , 0.        , 0.        , 0.        ],
[1.        , 0.        , 0.        , 0.        ],
[0.5       , 0.5       , 0.        , 0.        ],
[0.33333333, 0.33333333, 0.33333333, 0.        ]])
```

```
/Users/evseykirichkov/anaconda3/lib/python3.11/site-packages/gym/utils/pas
sive_env_checker.py:233: DeprecationWarning: `np.bool8` is a deprecated al
ias for `np.bool_`.  (Deprecated NumPy 1.24)
  if not isinstance(terminated, (bool, np.bool8)):
```

Ввод [8]:

```
image_path = "data/Screen_CliffWalking.png"
display(Image(filename=image_path))
```



Итог

Итогом работы программы является обученный агент, способный эффективно находить оптимальный путь в среде CliffWalking, минимизируя количество штрафов за падение с обрыва. Обученный агент демонстрирует свою работу, используя найденную стратегию.

Ввод [ ]: