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 №4: «Реализация алгоритма Policy Iteration»

Выполнила:
студентка группы ИУ5-24М
Мащенко Е. И.

Проверил:
Балашов А.М.

2023

Цель 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).

Выполнение работы

Для реализации алгоритма Policy Iteration была выбрана среда обучения Cliff Walking из библиотеки gym. Согласно документации, агент может находиться в 48 состояниях и осуществлять 4 действия.

Лабораторная работа №4

В [4]:
```
! pip install gymnasium
```

```
Requirement already satisfied: gymnasium in c:\users\user\anaconda3\lib\site-packages (0.28.1)
Requirement already satisfied: cloudpickle>=1.2.0 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (1.6.0)
Requirement already satisfied: importlib-metadata>=4.8.0; python_version < "3.10" in c:\users\user\anaconda3\lib\site-packag
es (from gymnasium) (6.6.0)
Requirement already satisfied: jax-jumpy>=1.0.0 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (1.0.0)
Requirement already satisfied: numpy>=1.21.0 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (1.24.3)
Requirement already satisfied: farama-notifications>=0.0.1 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (0.
0.4)
Requirement already satisfied: typing-extensions>=4.3.0 in c:\users\user\anaconda3\lib\site-packages (from gymnasium) (4.6.
3)
Requirement already satisfied: zipp>=0.5 in c:\users\user\anaconda3\lib\site-packages (from importlib-metadata>=4.8.0; pytho
n_version < "3.10"->gymnasium) (3.4.0)
```

В [5]:
```
import gymnasium as gym
import numpy as np
from pprint import pprint
import matplotlib.pyplot as plt
```

В [6]:
```
class PolicyIterationAgent:
    '''
    Класс, эмулирующий работу агента
    '''
    def __init__(self, env):
        self.env = env
        # Пространство состояний
        self.observation_dim = 4*12
        # Массив действий в соответствии с документацией
        self.actions_variants = np.array([0,1,2,3])
        # Задание стратегии (политики)
        # Карта 4x4 и 4 возможных действия
        self.policy_probs = np.full((self.observation_dim, len(self.actions_variants)), 0.25)
        # 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.observation_dim))
            # 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 self.env.P[state][action]:
                        innerSum=innerSum+probability*(reward+self.gamma*self.state_values[next_state])
                    outerSum=outerSum+self.policy_probs[state][action]*innerSum
                valueFunctionVectorNextIteration[state]=outerSum
            if(np.max(np.abs(valueFunctionVectorNextIteration-valueFunctionVector))<self.theta):
                # 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_variants)))
        improvedPolicy=np.zeros((self.observation_dim, len(self.actions_variants)))
        # 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 self.env.P[state][action]:
                    qvaluesMatrix[state,action]=qvaluesMatrix[state,action]+probability*(reward+self.gamma*self.state_values[

            # Находим лучшие индексы
            bestActionIndex=np.where(qvaluesMatrix[state,:]==np.max(qvaluesMatrix[state,:]))
            # Обновление стратегии
            improvedPolicy[state,bestActionIndex]=1/np.size(bestActionIndex)
        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.')
```

В [7]:
```
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
```

В [8]:
```
!pip install pygame
```

```
Collecting pygame
  Using cached pygame-2.4.0-cp38-cp38-win_amd64.whl (10.6 MB)
Installing collected packages: pygame
Successfully installed pygame-2.4.0
```

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

```
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],
```

Пример агента в конечном состоянии: