Лабораторная работа №4

Андреев А.В. ИУ5-25М

Реализация алгоритма Policy Iteration

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).

In [1]:

```
!pip install gymnasium
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Requirement already satisfied: gymnasium in c:\users\user\appdata\roaming
\python\python38\site-packages (0.28.1)
Requirement already satisfied: importlib-metadata>=4.8.0; python_version <
"3.10" in c:\users\user\appdata\roaming\python\python38\site-packages (fro
m gymnasium) (6.6.0)
Requirement already satisfied: typing-extensions>=4.3.0 in c:\users\user\a
ppdata\roaming\python\python38\site-packages (from gymnasium) (4.6.3)
Requirement already satisfied: farama-notifications>=0.0.1 in c:\users\use
r\appdata\roaming\python\python38\site-packages (from gymnasium) (0.0.4)
Requirement already satisfied: numpy>=1.21.0 in c:\users\user\appdata\roam
ing\python\python38\site-packages (from gymnasium) (1.24.3)
Requirement already satisfied: cloudpickle>=1.2.0 in d:\program files\anac
onda3\lib\site-packages (from gymnasium) (1.6.0)
Requirement already satisfied: jax-jumpy>=1.0.0 in c:\users\user\appdata\r
oaming\python\python38\site-packages (from gymnasium) (1.0.0)
Requirement already satisfied: zipp>=0.5 in d:\program files\anaconda3\lib
\site-packages (from importlib-metadata>=4.8.0; python_version < "3.10"->g
ymnasium) (3.4.0)
```

In [2]:

```
import gymnasium as gym
import numpy as np
from pprint import pprint
```

In [3]:

```
class PolicyIterationAgent:
    '''
    Класс, эмулирующий работу агента
    '''
    def __init__(self, env):
        self.env = env
        # Пространство состояний
        self.observation_dim = 500
        # Массив действий в соответствии с документацией
        # https://gymnasium.farama.org/environments/toy_text/taxi/
        self.actions_variants = np.array([0,1,2,3,4,5])
        # 0: Move south (down)
        # 1: Move north (up)
        # 2: Move east (right)
        # 3: Move west (left)
        # 4: Pickup passenger
        # 5: Drop off passenger
        # Задание стратегии (политики)
        # Карта 5x5 и 6 возможных действия
        self.policy_probs = np.full((self.observation_dim, len(self.actions_variants)), 0
        # 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[st
                        innerSum=innerSum+probability*(reward+self.gamma*self.state_value
                    outerSum=outerSum+self.policy_probs[state][action]*innerSum
                valueFunctionVectorNextIteration[state]=outerSum
            if(np.max(np.abs(valueFunctionVectorNextIteration-valueFunctionVector))<self.
                # 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]
                    qvaluesMatrix[state,action]=qvaluesMatrix[state,action]+probability*(

            # 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'{i} шагов.')
```

In [4]:

```
def play_agent(agent):
    env2 = gym.make('Taxi-v3', 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
```

In [5]:

```
# Создание среды
env = gym.make('Taxi-v3')
env.reset()
# Обучение агента
agent = PolicyIterationAgent(env)
agent.policy_iteration(1000)
agent.print_policy()
```

```
1000 шагов.
Стратегия:
array([[0. , 0. , 0. , 0. , 1. , 0. ],
       [0. , 0. , 0. , 0. , 1. , 0. ],
       [0. , 0. , 0. , 0. , 1. , 0. ],
       ...,
       [0. , 1. , 0. , 0. , 0. , 0. ],
       [0. , 0.5, 0. , 0.5, 0. , 0. ],
       [0. , 0. , 0. , 1. , 0. , 0. ]])
```

In [6]:

```
!pip install pygame
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Requirement already satisfied: pygame in c:\users\user\appdata\roaming\pyt
hon\python38\site-packages (2.4.0)
```

In [7]:

```
import os
os.environ['SDL_VIDEODRIVER']='dummy'
import pygame
pygame.display.set_mode((640,480))
# Проигрывание сцены для обученного агента
play_agent(agent)
```

In [8]:

```
play_agent(agent)
```

In [ ]:

```

```