

Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



Лабораторная работа № 4

по дисциплине «Методы машинного обучения»

Алгоритм Policy Iteration

ИСПОЛНИТЕЛЬ:

студент ИУ5-23М

Анцифров Н.С.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю. Е.

___ " _____ " 2024 г.

Москва, 2024

Задание лабораторной работы

На основе рассмотренного на лекции примера реализовать 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

Описание среды

Возьмём из библиотеки Gym среду Taxi-v3: https://www.gymnasium.dev/environments/toy_text/taxi/

Задача представляет собой задачу о такси из книги Тома Диттериха "Обучение с иерархическим подкреплением с помощью декомпозиции функции MAXQ Value".

На карте есть 4 определенных места, обозначенных R(ed), G(reen), Y(ellow) и B(lue). Когда начинается поездка, такси выезжает из случайного квадрата, а пассажир оказывается в случайном месте. Такси подъезжает к месту нахождения пассажира, забирает его, отвозит в пункт назначения (другое из 4 указанных мест), а затем высаживает пассажира. Как только пассажир высажен, поездка заканчивается.

Есть 500 состояний:

- карта размером 5x5;
- 4 локации;
- 5 состояний пассажира (4 выхода и в такси).

Есть 6 действий:

- 0: двигаться на юг;
- 1: двигаться на север;
- 2: двигаться на запад;
- 3: двигаться на восток;
- 4: посадить пассажира;
- 5: высадить пассажира.

Существует 400 состояний, до которых можно добраться во время поездки. Пропущенные состояния соответствуют ситуациям, в которых пассажир находится в том же месте, что и пункт назначения, поскольку это обычно сигнализирует об окончании поездки. 4 дополнительных состояния можно наблюдать сразу после успешного завершения поездки, когда и пассажир, и такси находятся в пункте назначения. Всего получается 404 доступных дискретных состояния.

Каждое пространство состояний представлено кортежем: (taxi_row, taxi_col, passenger_location, destination).

Точки посадки пассажира:

- 0: R(ed);
- 1: G(reen);
- 2: Y(ellow);
- 3: B(lue);
- 4: в такси.

Пункты назначения (пункты высадки):

- 0: R(ed);
- 1: G(reen);
- 2: Y(ellow);
- 3: B(lue).

Награды:

- -1 за каждый шаг, если не предусмотрено иное вознаграждение;
- +20 за доставку пассажира;
- -10 за некорректное выполнение действий "погрузка" и "высадка".

Код программы

In [1]:

```
import gym
import numpy as np
import matplotlib.pyplot as plt
from pprint import pprint
import pandas as pd
from gym.envs.toy_text.taxi import TaxiEnv
```

In [2]:

```
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
```

```
        self.actions_variants = np.array([0,1,2,3,4,5])
```

```
        # Задание стратегии (политики)
```

```
        self.policy_probs = np.full((self.observation_dim, len(self.actions_variants)), 0.16666666)
```

```
        # 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
```

```
        """
```

```
        if self.policy_probs[0][0] != 0.16666666:
```

```
            #np.set_printoptions(threshold=np.inf)
```

```
            x = TaxiEnv()
```

```
            pos = {0:'R', 1:'G',2:'Y', 3:'B', 4:'T'}
```

```
            print("""
```

```
            +-----+
```

```
            |R: | : :G|
```

```
            | : | : :|
```

```
            | : : :|
```

```
            | | : | :|
```

```
            |Y| : |B: |
```

```
            +-----+
```

```
            """)
```

```
            print('состояние: x,y,пассажир,назначение')
```

```
            print('Стратегия:')
```

```
            for i in range(len(self.policy_probs)):
```

```
                t_x,t_y,passeng,dest = x.decode(i)
```

```
                print((t_x,t_y,pos[passeng],pos[dest]), self.policy_probs[i])
```

```
            #np.set_printoptions(threshold=False)
```

```
        else:
```

```
            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[next_state])

        # 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 выполнен за {i} шагов.')

```

In [3]:

```
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
```

Работа программы

Выведем стратегии:

```
In [4]:
env = gym.make('Taxi-v3')
env.reset()
# Обучение агента
agent = PolicyIterationAgent(env)
agent.print_policy()
agent.policy_iteration(1000)
agent.print_policy()

Стратегия:
array([[0.16666666, 0.16666666, 0.16666666, 0.16666666, 0.16666666,
        0.16666666],
       [0.16666666, 0.16666666, 0.16666666, 0.16666666, 0.16666666,
        0.16666666],
       [0.16666666, 0.16666666, 0.16666666, 0.16666666, 0.16666666,
        0.16666666],
       ...,
       [0.16666666, 0.16666666, 0.16666666, 0.16666666, 0.16666666,
        0.16666666],
       [0.16666666, 0.16666666, 0.16666666, 0.16666666, 0.16666666,
        0.16666666],
       [0.16666666, 0.16666666, 0.16666666, 0.16666666, 0.16666666,
        0.16666666]])
```

Алгоритм выполнялся за 1000 шагов.

```
+-----+
|R: | : :G|
| : | : :|
| : : : |
| | : | :|
|Y| : |B: |
+-----+
```

состояние: х,у,пассажир,назначение

Стратегия:

```
(0, 0, 'R', 'R') [0. 0. 0. 0. 1. 0.]
(0, 0, 'R', 'G') [0. 0. 0. 0. 1. 0.]
(0, 0, 'R', 'Y') [0. 0. 0. 0. 1. 0.]
(0, 0, 'R', 'B') [0. 0. 0. 0. 1. 0.]
(0, 0, 'G', 'R') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'G', 'G') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'G', 'Y') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'G', 'B') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'Y', 'R') [1. 0. 0. 0. 0. 0.]
(0, 0, 'Y', 'G') [1. 0. 0. 0. 0. 0.]
(0, 0, 'Y', 'Y') [1. 0. 0. 0. 0. 0.]
(0, 0, 'Y', 'B') [1. 0. 0. 0. 0. 0.]
(0, 0, 'B', 'R') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'B', 'G') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'B', 'Y') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'B', 'B') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'T', 'R') [0. 0. 0. 0. 0. 1.]
(0, 0, 'T', 'G') [0.5 0. 0.5 0. 0. 0.]
(0, 0, 'T', 'Y') [1. 0. 0. 0. 0. 0.]
(0, 0, 'T', 'B') [0.5 0. 0.5 0. 0. 0.]
(0, 1, 'R', 'R') [0. 0. 0. 1. 0. 0.]
```

(0, 1, 'R', 'G') [0. 0. 0. 1. 0. 0.]
(0, 1, 'R', 'Y') [0. 0. 0. 1. 0. 0.]
(0, 1, 'R', 'B') [0. 0. 0. 1. 0. 0.]
(0, 1, 'G', 'R') [1. 0. 0. 0. 0. 0.]
(0, 1, 'G', 'G') [1. 0. 0. 0. 0. 0.]
(0, 1, 'G', 'Y') [1. 0. 0. 0. 0. 0.]
(0, 1, 'G', 'B') [1. 0. 0. 0. 0. 0.]
(0, 1, 'Y', 'R') [0.5 0. 0. 0.5 0. 0.]
(0, 1, 'Y', 'G') [0.5 0. 0. 0.5 0. 0.]
(0, 1, 'Y', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 1, 'Y', 'B') [0.5 0. 0. 0.5 0. 0.]
(0, 1, 'B', 'R') [1. 0. 0. 0. 0. 0.]
(0, 1, 'B', 'G') [1. 0. 0. 0. 0. 0.]
(0, 1, 'B', 'Y') [1. 0. 0. 0. 0. 0.]
(0, 1, 'B', 'B') [1. 0. 0. 0. 0. 0.]
(0, 1, 'T', 'R') [0. 0. 0. 1. 0. 0.]
(0, 1, 'T', 'G') [1. 0. 0. 0. 0. 0.]
(0, 1, 'T', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 1, 'T', 'B') [1. 0. 0. 0. 0. 0.]
(0, 2, 'R', 'R') [1. 0. 0. 0. 0. 0.]
(0, 2, 'R', 'G') [1. 0. 0. 0. 0. 0.]
(0, 2, 'R', 'Y') [1. 0. 0. 0. 0. 0.]
(0, 2, 'R', 'B') [1. 0. 0. 0. 0. 0.]
(0, 2, 'G', 'R') [0. 0. 1. 0. 0. 0.]
(0, 2, 'G', 'G') [0. 0. 1. 0. 0. 0.]
(0, 2, 'G', 'Y') [0. 0. 1. 0. 0. 0.]
(0, 2, 'G', 'B') [0. 0. 1. 0. 0. 0.]
(0, 2, 'Y', 'R') [1. 0. 0. 0. 0. 0.]
(0, 2, 'Y', 'G') [1. 0. 0. 0. 0. 0.]
(0, 2, 'Y', 'Y') [1. 0. 0. 0. 0. 0.]
(0, 2, 'Y', 'B') [1. 0. 0. 0. 0. 0.]
(0, 2, 'B', 'R') [0.5 0. 0.5 0. 0. 0.]
(0, 2, 'B', 'G') [0.5 0. 0.5 0. 0. 0.]
(0, 2, 'B', 'Y') [0.5 0. 0.5 0. 0. 0.]
(0, 2, 'B', 'B') [0.5 0. 0.5 0. 0. 0.]
(0, 2, 'T', 'R') [1. 0. 0. 0. 0. 0.]
(0, 2, 'T', 'G') [0. 0. 1. 0. 0. 0.]
(0, 2, 'T', 'Y') [1. 0. 0. 0. 0. 0.]
(0, 2, 'T', 'B') [0.5 0. 0.5 0. 0. 0.]
(0, 3, 'R', 'R') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'R', 'G') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'R', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'R', 'B') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'G', 'R') [0. 0. 1. 0. 0. 0.]
(0, 3, 'G', 'G') [0. 0. 1. 0. 0. 0.]
(0, 3, 'G', 'Y') [0. 0. 1. 0. 0. 0.]
(0, 3, 'G', 'B') [0. 0. 1. 0. 0. 0.]
(0, 3, 'Y', 'R') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'Y', 'G') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'Y', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'Y', 'B') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'B', 'R') [1. 0. 0. 0. 0. 0.]
(0, 3, 'B', 'G') [1. 0. 0. 0. 0. 0.]
(0, 3, 'B', 'Y') [1. 0. 0. 0. 0. 0.]
(0, 3, 'B', 'B') [1. 0. 0. 0. 0. 0.]
(0, 3, 'T', 'R') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'T', 'G') [0. 0. 1. 0. 0. 0.]
(0, 3, 'T', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 3, 'T', 'B') [1. 0. 0. 0. 0. 0.]
(0, 4, 'R', 'R') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'R', 'G') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'R', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'R', 'B') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'G', 'R') [0. 0. 0. 0. 1. 0.]
(0, 4, 'G', 'G') [0. 0. 0. 0. 1. 0.]
(0, 4, 'G', 'Y') [0. 0. 0. 0. 1. 0.]
(0, 4, 'G', 'B') [0. 0. 0. 0. 1. 0.]
(0, 4, 'Y', 'R') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'Y', 'G') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'Y', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'Y', 'B') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'B', 'R') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'B', 'G') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'B', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'B', 'B') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'T', 'R') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'T', 'G') [0. 0. 0. 0. 0. 1.]
(0, 4, 'T', 'Y') [0.5 0. 0. 0.5 0. 0.]
(0, 4, 'T', 'B') [0.5 0. 0. 0.5 0. 0.]
(1, 0, 'R', 'R') [0. 1. 0. 0. 0. 0.]
(1, 0, 'R', 'G') [0. 1. 0. 0. 0. 0.]
(1, 0, 'R', 'Y') [0. 1. 0. 0. 0. 0.]

```

(4, 1, 'Y', 'Y') [0. 1. 0. 0. 0. 0.]
(4, 1, 'Y', 'B') [0. 1. 0. 0. 0. 0.]
(4, 1, 'B', 'R') [0. 0.5 0.5 0. 0. 0. ]
(4, 1, 'B', 'G') [0. 0.5 0.5 0. 0. 0. ]
(4, 1, 'B', 'Y') [0. 0.5 0.5 0. 0. 0. ]
(4, 1, 'B', 'B') [0. 0.5 0.5 0. 0. 0. ]
(4, 1, 'T', 'R') [0. 1. 0. 0. 0. 0.]
(4, 1, 'T', 'G') [0. 0.5 0.5 0. 0. 0. ]
(4, 1, 'T', 'Y') [0. 1. 0. 0. 0. 0.]
(4, 1, 'T', 'B') [0. 0.5 0.5 0. 0. 0. ]
(4, 2, 'R', 'R') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'R', 'G') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'R', 'Y') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'R', 'B') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'G', 'R') [0. 1. 0. 0. 0. 0.]
(4, 2, 'G', 'G') [0. 1. 0. 0. 0. 0.]
(4, 2, 'G', 'Y') [0. 1. 0. 0. 0. 0.]
(4, 2, 'G', 'B') [0. 1. 0. 0. 0. 0.]
(4, 2, 'Y', 'R') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'Y', 'G') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'Y', 'Y') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'Y', 'B') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'B', 'R') [0. 1. 0. 0. 0. 0.]
(4, 2, 'B', 'G') [0. 1. 0. 0. 0. 0.]
(4, 2, 'B', 'Y') [0. 1. 0. 0. 0. 0.]
(4, 2, 'B', 'B') [0. 1. 0. 0. 0. 0.]
(4, 2, 'T', 'R') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'T', 'G') [0. 1. 0. 0. 0. 0.]
(4, 2, 'T', 'Y') [0. 0.5 0. 0.5 0. 0. ]
(4, 2, 'T', 'B') [0. 1. 0. 0. 0. 0.]
(4, 3, 'R', 'R') [0. 1. 0. 0. 0. 0.]
(4, 3, 'R', 'G') [0. 1. 0. 0. 0. 0.]
(4, 3, 'R', 'Y') [0. 1. 0. 0. 0. 0.]
(4, 3, 'R', 'B') [0. 1. 0. 0. 0. 0.]
(4, 3, 'G', 'R') [0. 0.5 0.5 0. 0. 0. ]
(4, 3, 'G', 'G') [0. 0.5 0.5 0. 0. 0. ]
(4, 3, 'G', 'Y') [0. 0.5 0.5 0. 0. 0. ]
(4, 3, 'G', 'B') [0. 0.5 0.5 0. 0. 0. ]
(4, 3, 'Y', 'R') [0. 1. 0. 0. 0. 0.]
(4, 3, 'Y', 'G') [0. 1. 0. 0. 0. 0.]
(4, 3, 'Y', 'Y') [0. 1. 0. 0. 0. 0.]
(4, 3, 'Y', 'B') [0. 1. 0. 0. 0. 0.]
(4, 3, 'B', 'R') [0. 0. 0. 0. 1. 0.]
(4, 3, 'B', 'G') [0. 0. 0. 0. 1. 0.]
(4, 3, 'B', 'Y') [0. 0. 0. 0. 1. 0.]
(4, 3, 'B', 'B') [0. 0. 0. 0. 1. 0.]
(4, 3, 'T', 'R') [0. 1. 0. 0. 0. 0.]
(4, 3, 'T', 'G') [0. 0.5 0.5 0. 0. 0. ]
(4, 3, 'T', 'Y') [0. 1. 0. 0. 0. 0.]
(4, 3, 'T', 'B') [0. 0. 0. 0. 0. 1.]
(4, 4, 'R', 'R') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'R', 'G') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'R', 'Y') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'R', 'B') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'G', 'R') [0. 1. 0. 0. 0. 0.]
(4, 4, 'G', 'G') [0. 1. 0. 0. 0. 0.]
(4, 4, 'G', 'Y') [0. 1. 0. 0. 0. 0.]
(4, 4, 'G', 'B') [0. 1. 0. 0. 0. 0.]
(4, 4, 'Y', 'R') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'Y', 'G') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'Y', 'Y') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'Y', 'B') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'B', 'R') [0. 0. 0. 0. 1. 0.]
(4, 4, 'B', 'G') [0. 0. 0. 0. 1. 0.]
(4, 4, 'B', 'Y') [0. 0. 0. 0. 1. 0.]
(4, 4, 'B', 'B') [0. 0. 0. 0. 1. 0.]
(4, 4, 'T', 'R') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'T', 'G') [0. 1. 0. 0. 0. 0.]
(4, 4, 'T', 'Y') [0. 0.5 0. 0.5 0. 0. ]
(4, 4, 'T', 'B') [0. 0. 0. 0. 1. 0.]
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. ]])

```

Проигрывание сцены для обученного агента:

```

In [5]:
play_agent(agent)

```