



**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»
Кафедра ИУ5 «Системы обработки информации и управления»**

**Отчет по лабораторной работы №4
по дисциплине «Методы машинного обучения»
по теме «Реализация алгоритма Policy Iteration»**

**Выполнил:
студент группы № ИУ5-21М
Торжков М.С.
подпись, дата**

**Проверила:
Балашов А.М.
подпись, дата**

2023 г.

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).

```
! pip install gymnasium
import gymnasium as gym
import numpy as np
from pprint import pprint
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Collecting gymnasium

Downloading gymnasium-0.28.1-py3-none-any.whl (925 kB)

0:00:00 925.5/925.5 kB 13.5 MB/s eta

Requirement already satisfied: numpy>=1.21.0 in

/usr/local/lib/python3.10/dist-packages (from gymnasium) (1.22.4)

Collecting jax-jumpy>=1.0.0 (from gymnasium)

Downloading jax_jumpy-1.0.0-py3-none-any.whl (20 kB)

Requirement already satisfied: cloudpickle>=1.2.0 in

/usr/local/lib/python3.10/dist-packages (from gymnasium) (2.2.1)

Requirement already satisfied: typing-extensions>=4.3.0 in

/usr/local/lib/python3.10/dist-packages (from gymnasium) (4.5.0)

Collecting farama-notifications>=0.0.1 (from gymnasium)

Downloading Farama_Notifications-0.0.4-py3-none-any.whl (2.5 kB)

Installing collected packages: farama-notifications, jax-jumpy, gymnasium

Successfully installed farama-notifications-0.0.4 gymnasium-0.28.1 jax-jumpy-1.0.0

```
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)), 0.25)
```

```
        # 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[nex
t_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'{i} шагов.')

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

# 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. ]])

```

```
!pip install pygame
```

```

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

```

```

Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pygame in
/usr/local/lib/python3.10/dist-packages (2.3.0)

```

```
play_agent(agent)
```


