****Московский государственный технический университет им. Н.Э. Баумана  
Кафедра «Системы обработки информации и управления»

Лабораторная работа №4  
по дисциплине  
«Методы машинного обучения»  
на тему

# «Реализация алгоритма Policy Iteration»

Выполнил:  
студент группы ИУ5-23М  
Чжэн Сяохуэй

Москва — 2024 г.

**1. Цель лабораторной работы**

ознакомление с базовыми методами обучения с подкреплением.

**2. Задание**

1. 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](https://www.gymlibrary.dev/) (или аналогичной библиотеки).

**3. текст программы**

**import** gym

**import** numpy **as** np

**from** pprint **import** pprint

**class** PolicyIterationAgent:

'''

Класс, эмулирующий работу агента

'''

**def** \_\_init\_\_(self, env):

self**.**env **=** env

*# Пространство состояний*

self**.**observation\_dim **=** env**.**observation\_space**.**n

*# Пространство действий*

self**.**action\_dim **=** env**.**action\_space**.**n

*# Задание стратегии (политики)*

self**.**policy\_probs **=** np**.**full((self**.**observation\_dim, self**.**action\_dim), 1 **/** self**.**action\_dim)

*# 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

'''

valueFunctionVector **=** self**.**state\_values

**for** iterations **in** range(self**.**maxNumberOfIterations):

valueFunctionVectorNextIteration **=** np**.**zeros(shape**=**(self**.**observation\_dim))

**for** state **in** range(self**.**observation\_dim):

action\_probabilities **=** self**.**policy\_probs[state]

outerSum **=** 0

**for** action, prob **in** enumerate(action\_probabilities):

innerSum **=** 0

**for** probability, next\_state, reward, isTerminalState **in** self**.**env**.**P[state][action]:

innerSum **+=** probability **\*** (reward **+** self**.**gamma **\*** self**.**state\_values[next\_state])

outerSum **+=** self**.**policy\_probs[state][action] **\*** innerSum

valueFunctionVectorNextIteration[state] **=** outerSum

**if** np**.**max(np**.**abs(valueFunctionVectorNextIteration **-** valueFunctionVector)) **<** self**.**theta:

valueFunctionVector **=** valueFunctionVectorNextIteration

**break**

valueFunctionVector **=** valueFunctionVectorNextIteration

**return** valueFunctionVector

**def** policy\_improvement(self):

'''

Улучшение стратегии

'''

qvaluesMatrix **=** np**.**zeros((self**.**observation\_dim, self**.**action\_dim))

improvedPolicy **=** np**.**zeros((self**.**observation\_dim, self**.**action\_dim))

**for** state **in** range(self**.**observation\_dim):

**for** action **in** range(self**.**action\_dim):

**for** probability, next\_state, reward, isTerminalState **in** self**.**env**.**P[state][action]:

qvaluesMatrix[state, action] **+=** probability **\*** (reward **+** self**.**gamma **\*** self**.**state\_values[next\_state])

bestActionIndex **=** np**.**where(qvaluesMatrix[state, :] **==** np**.**max(qvaluesMatrix[state, :]))

improvedPolicy[state, bestActionIndex] **=** 1 **/** np**.**size(bestActionIndex)

**return** improvedPolicy

**def** policy\_iteration(self, cnt):

'''

Основная реализация алгоритма

'''

**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.')

**def** play\_agent(agent):

env2 **=** gym**.**make('Taxi-v3', render\_mode**=**'human')

state **=** env2**.**reset()

**if** isinstance(state, tuple):

state **=** state[0]

env2**.**render()

done **=** **False**

**while** **not** done:

p **=** agent**.**policy\_probs[state]

**if** isinstance(p, np**.**ndarray):

action **=** np**.**random**.**choice(agent**.**action\_dim, p**=**p)

**else**:

action **=** p

next\_state, reward, terminated, truncated, \_ **=** env2**.**step(action)

**if** isinstance(next\_state, tuple):

next\_state **=** next\_state[0]

env2**.**render()

state **=** next\_state

**if** terminated **or** truncated:

done **=** **True**

env2**.**close()

**def** main():

*# Создание среды*

env **=** gym**.**make('Taxi-v3')

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)

**if** \_\_name\_\_ **==** '\_\_main\_\_':

main()

4. экранные формы с примерами выполнения программы.

Стратегия:

array([[0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,

0.16666667],

[0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,

0.16666667],

[0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,

0.16666667],

...,

[0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,

0.16666667],

[0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,

0.16666667],

[0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,

0.16666667]])

Алгоритм выполнился за 1000 шагов.

Стратегия:

array([[0. , 0. , 0. , 0. , 1. , 0. ],

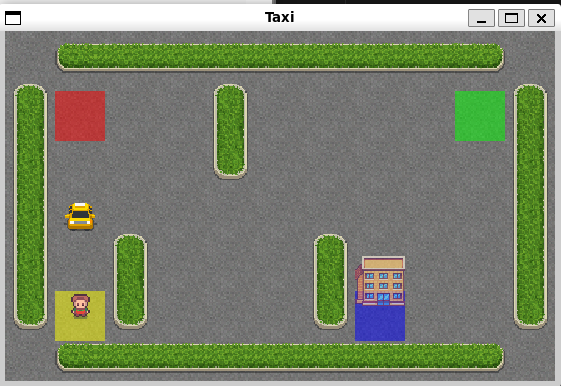
[0. , 0. , 0. , 0. , 1. , 0. ],

[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. ]])

**Список литературы**

[1] Гапанюк Ю. Е. COURSE\_MMO\_SPRING\_2024// GitHub. –– 2024. –– Режим доступа: https://github.com/ugapanyuk/courses\_current/wiki/COURSE\_MMO\_SPRING\_2024

[2] <https://www.kaggle.com/datasets>