



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, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

## **Отчет по лабораторной работе №4**

### **«Алгоритм Policy Iteration»**

Студент группы ИУ5-25М

Зозуля О.А.

Москва, 2023 г.

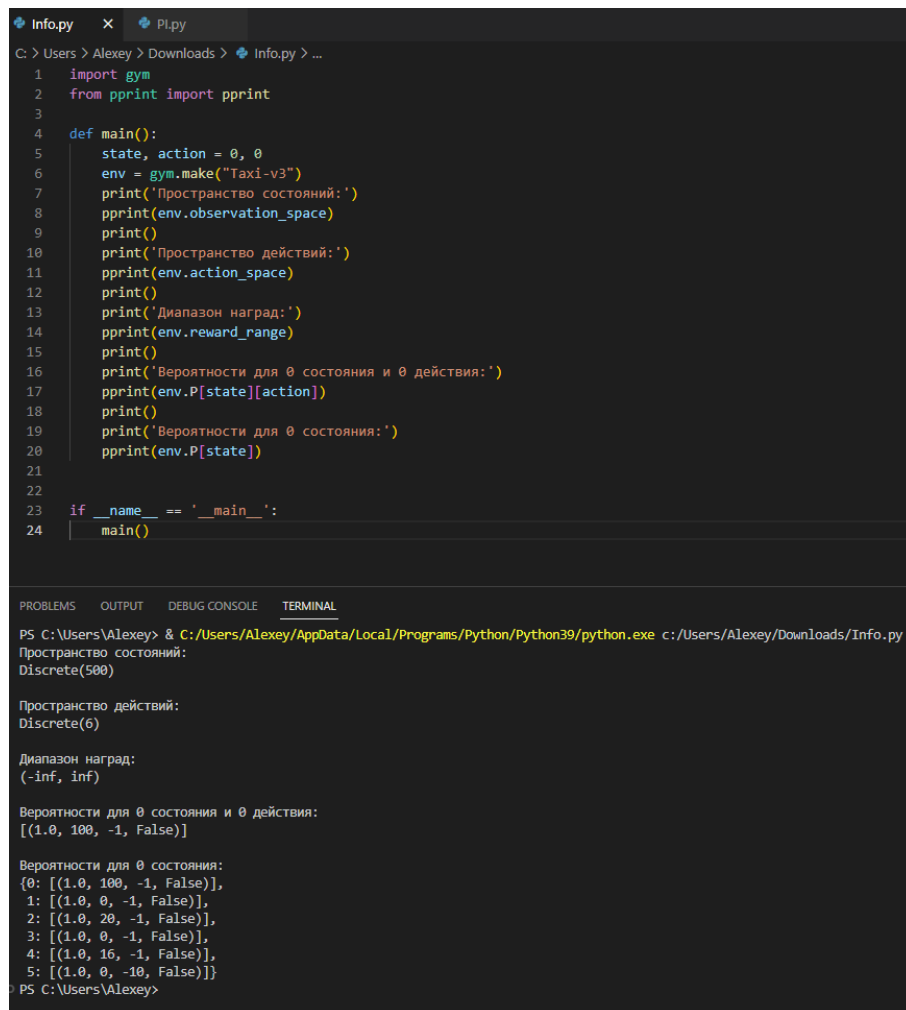
## Цель работы

Ознакомление с базовыми методами обучения с подкреплением.

## Задание

Реализовать алгоритм Policy Iteration для любой среды обучения с подкреплением.

## Описание выполнения



```
Info.py X PL.py
C: > Users > Alexey > Downloads > Info.py > ...
1 import gym
2 from pprint import pprint
3
4 def main():
5     state, action = 0, 0
6     env = gym.make("Taxi-v3")
7     print('Пространство состояний:')
8     pprint(env.observation_space)
9     print()
10    print('Пространство действий:')
11    pprint(env.action_space)
12    print()
13    print('Диапазон наград:')
14    pprint(env.reward_range)
15    print()
16    print('Вероятности для 0 состояния и 0 действия:')
17    pprint(env.P[state][action])
18    print()
19    print('Вероятности для 0 состояния:')
20    pprint(env.P[state])
21
22
23 if __name__ == '__main__':
24     main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Alexey> & C:/Users/Alexey/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Alexey/Downloads/Info.py

Пространство состояний:  
Discrete(500)

Пространство действий:  
Discrete(6)

Диапазон наград:  
(-inf, inf)

Вероятности для 0 состояния и 0 действия:  
[(1.0, 100, -1, False)]

Вероятности для 0 состояния:  
{0: [(1.0, 100, -1, False)],  
1: [(1.0, 0, -1, False)],  
2: [(1.0, 20, -1, False)],  
3: [(1.0, 0, -1, False)],  
4: [(1.0, 16, -1, False)],  
5: [(1.0, 0, -10, False)]}

PS C:\Users\Alexey>

Рисунок 1 – Информация о среде

```

1 import gym
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from pprint import pprint
5
6
7 class PolicyIterationAgent:
8     """
9     Класс, выполняющий работу агента
10     """
11     def __init__(self, env):
12         self.env = env
13         # Пространство состояний
14         self.observation_dim = 580
15         # Матрица действий # соответствует с документацией
16         self.action_variants = np.array([0,1,2,3,4,5])
17         # Задание стратегии (политики)
18         # Карта dim x 4 возможных действий
19         self.policy_probs = np.full((self.observation_dim, len(self.action_variants)), 0.25)
20         # Начальное значение для v(s)
21         self.state_values = np.zeros(shape=(self.observation_dim))
22         # Начальное значение параметров
23         self.maxNumberOfIterations = 1000
24         self.theta=1e-6
25         self.gamma=0.99
26
27     def print_policy(self):
28         """
29         Вывод матрицы стратегии
30         """
31         print('Стратегия:')
32         pprint(self.policy_probs)
33
34     def policy_evaluation(self):
35         """
36         Оценивание стратегии
37         """
38         # Предустановленное значение функции ценности
39         valueFunctionVector = self.state_values
40         for iterations in range(self.maxNumberOfIterations):
41             # Новое значение функции ценности
42             valueFunctionVectorNextIteration=np.zeros(shape=(self.observation_dim))
43             # Цикл по состояниям
44             for state in range(self.observation_dim):
45                 # Вероятности действий
46                 action_probabilities = self.policy_probs[state]
47                 # Сумма по действиям
48                 outerSum=0
49                 for action, prob in enumerate(action_probabilities):
50                     innerSum=0
51                     # Цикл по вероятностям действий
52                     for probability, next_state, reward, isTerminalState in self.env.P[state][action]:
53                         innerSum=innerSum+probability*(reward+self.gamma*self.state_values[next_state])
54                     outerSum+=outerSum+self.policy_probs[state][action]*innerSum
55                     valueFunctionVectorNextIteration[state]=outerSum
56                 if(np.max(np.abs(valueFunctionVectorNextIteration-valueFunctionVector))<self.theta):
57                     # Проверка сходимости алгоритма
58                     valueFunctionVector=valueFunctionVectorNextIteration
59                     break
60             valueFunctionVector=valueFunctionVectorNextIteration
61         return valueFunctionVector
62
63
64     def policy_improvement(self):
65         """
66         Улучшение стратегии
67         """
68         qValueMatrix=np.zeros((self.observation_dim, len(self.action_variants)))
69         improvedPolicy=np.zeros((self.observation_dim, len(self.action_variants)))
70         # Цикл по состояниям
71         for state in range(self.observation_dim):
72             for action in range(len(self.action_variants)):
73                 for probability, next_state, reward, isTerminalState in self.env.P[state][action]:
74                     qValueMatrix[state,action]=qValueMatrix[state,action]+probability*(reward+self.gamma*self.state_values[next_state])
75             # Нахождение лучшего действия
76             bestActionIndex=np.argmax(qValueMatrix[state,:])
77             improvedPolicy[state,bestActionIndex]=1/np.size(bestActionIndex)
78             return improvedPolicy
79
80     def policy_iteration(self, cnt):
81         """
82         Оценка оптимальной стратегии
83         """
84         policy_stable = False
85         for i in range(1, cnt+1):
86             self.state_values = self.policy_evaluation()
87             self.policy_probs = self.policy_improvement()
88             print('Алгоритм закончился за {} итераций.'.format(i))
89
90     def play_agent(self):
91         env2 = gym.make('taxi-v3', render_mode='human')
92         state = env2.reset()[0]
93         done = False
94         while not done:
95             p = agent.policy_probs[state]
96             if isinstance(p, np.ndarray):
97                 action = np.random.choice(len(agent.action_variants), p=p)
98             else:
99                 action = p
100             next_state, reward, terminated, truncated, _ = env2.step(action)
101             env2.render()
102             state = next_state
103             if terminated or truncated:
104                 done = True
105
106     def main():
107         """
108         # Создание среды
109         env = gym.make('Taxi-v3')
110         env.reset()
111         # Обучение агента
112         agent = PolicyIterationAgent(env)
113         agent.print_policy()
114         agent.policy_iteration(1000)
115         agent.print_policy()
116         # Проверка работы агента
117         play_agent(agent)
118
119     if __name__ == '__main__':
120         main()

```

Рисунок 2-3 – Реализация алгоритма Policy Iteration

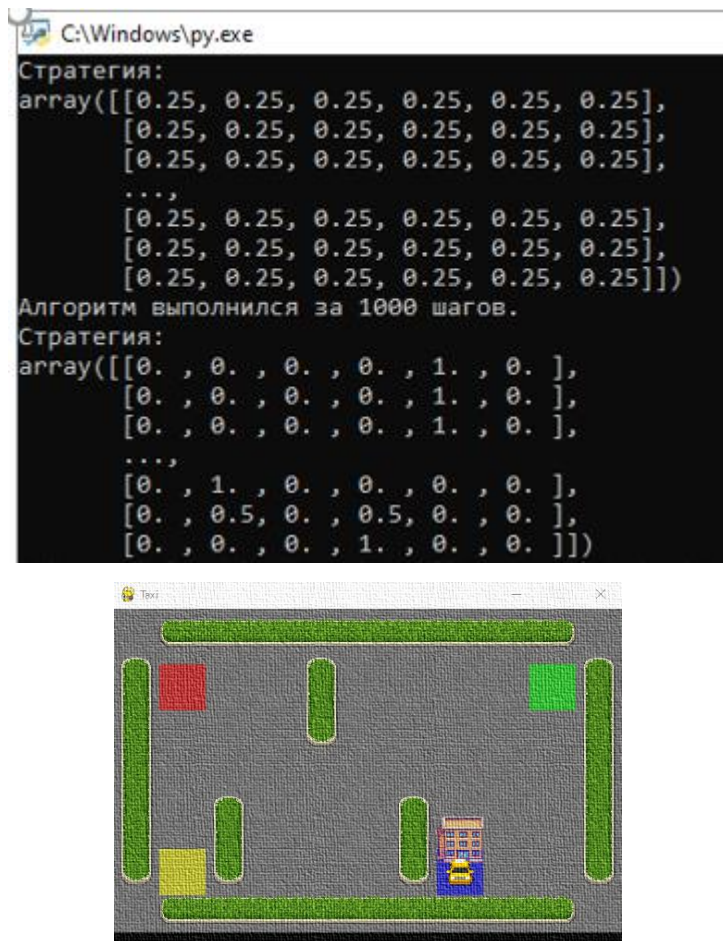


Рисунок 3-4 – Успешное обучение модели и ее начальная и конечные стратегии

## **Вывод**

Таким образом, удалось реализовать алгоритм Policy Iteration для среды обучения с подкреплением, таким образом ознакомившись с базовыми методами обучения с подкреплением.