****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 ИУ5И-21М  
Фань Лицзе

Москва — 2024 г.

**1. Цель лабораторной работы**

ознакомление с базовыми методами обучения с подкреплением.

**2. Задание**

На основе рассмотренного на лекции примера реализуйте алгоритм Policy Iteration для любой среды обучения с подкреплением (кроме рассмотренной на лекции).

1. **Текст программы**

import gym

import numpy as np

from pprint import pprint

Эти строки импортируют необходимые библиотеки и модули: Gym для среды, NumPy для численных вычислений и pprint для красивого вывода.

  class PolicyIterationAgent:

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

self.env = env

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

self.actions\_variants = np.arange(env.action\_space.n)

self.policy\_probs = np.full((self.observation\_dim, len(self.actions\_variants)), 1 / len(self.actions\_variants))

self.state\_values = np.zeros(self.observation\_dim)

self.maxNumberOfIterations = 1000

self.theta = 1e-6

self.gamma = 0.9

Здесь определяется класс PolicyIterationAgent для реализации алгоритма итерации по стратегии. В методе **init** инициализируются атрибуты агента, такие как среда, размер пространства наблюдений, пространства действий, вероятности стратегии, значения состояний и другие.

  def print\_policy(self):

print('Текущая политика:')

pprint(self.policy\_probs)

Это метод для вывода текущей стратегии. Он использует функцию pprint для красивого вывода вероятностей стратегии.

   def policy\_evaluation(self):

for \_ in range(self.maxNumberOfIterations):

delta = 0

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

v = 0

for action, action\_prob in enumerate(self.policy\_probs[state]):

for prob, next\_state, reward, done in self.env.P[state][action]:

v += action\_prob \* prob \* (reward + self.gamma \* self.state\_values[next\_state])

delta = max(delta, abs(self.state\_values[state] - v))

self.state\_values[state] = v

if delta < self.theta:

break

Это метод оценки стратегии, использующий шаг оценки в алгоритме итерации по стратегии для вычисления функции ценности состояния на основе текущей стратегии.

def policy\_improvement(self):

policy\_stable = True

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

old\_action = np.argmax(self.policy\_probs[state])

action\_values = np.zeros(len(self.actions\_variants))

for action in range(len(self.actions\_variants)):

for prob, next\_state, reward, done in self.env.P[state][action]:

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

best\_action = np.argmax(action\_values)

if old\_action != best\_action:

policy\_stable = False

self.policy\_probs[state] = np.eye(len(self.actions\_variants))[best\_action]

return policy\_stable

Это метод улучшения стратегии, который обновляет стратегию на основе текущей функции ценности состояния с целью получения более оптимальной стратегии.

     def policy\_iteration(self):

iteration = 0

while True:

self.policy\_evaluation()

if self.policy\_improvement():

print(f'Политика стабилизировалась после {iteration} итераций.')

break

iteration += 1

Это метод итерации по стратегии, который выполняет оценку и улучшение стратегии последовательно до стабилизации стратегии.

def play\_agent(agent):

state = agent.env.reset()

done = False

total\_reward = 0

while not done:

action = np.argmax(agent.policy\_probs[state])

state, reward, done, \_ = agent.env.step(action)

total\_reward += reward

return total\_reward

Это функция для имитации выполнения стратегии агента, возвращающая общую награду за выполнение стратегии в среде.

     def main():

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

agent = PolicyIterationAgent(env)

agent.policy\_iteration()

agent.print\_policy()

print('Награда агента:', play\_agent(agent))

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

main()

Это основная функция, которая запускает весь процесс: создает экземпляр среды, создает экземпляр агента, выполняет алгоритм итерации по стратегии, выводит оптимальную стратегию и выводит общую награду за выполнение стратегии агента.