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-23М
Ван Тяньшо

Москва — 2024 г.

1. Цель лабораторной работы

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 среды Toy Text / Frozen Lake) из библиотеки Gym (или аналогичной библиотеки).

3. Текст программы

Агент реализуется с помощью метода итераций стратегии, который используется для решения задачи интенсивного обучения Taxi-v3.Повторение стратегии включает в себя два основных этапа: оценку стратегии и совершенствование стратегии до тех пор, пока стратегия не станет стабильной.

Шаг 1: Импорт необходимых библиотек

```
import gym
import numpy as np
from pprint import pprint
```

Шаг 2: Секция оценки стратегии

```
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

    def print_policy(self):
        print('Текущая политика:')
        pprint(self.policy_probs)

    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
```

Шаг 3: Часть совершенствования 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
```

Шаг 4: Часть итерации 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

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

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()
```

4. Результат выполнения кода

```
/usr/local/lib/python3.10/dist-packages/gym/core.py:317: DeprecationWarning: WARN: Initializing wrapper in old step API which returns one bool instead of two. It is recomm
  deprecation(
/usr/local/lib/python3.10/dist-packages/gym/wrappers/step_api_compatibility.py:39: DeprecationWarning: WARN: Initializing environment in old step API which returns one boo
  deprecation(
Политика стабилизировалась после 12 итераций.
Текущая политика:
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., 1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0.]])
Награда агента: 12
/usr/local/lib/python3.10/dist-packages/gym/utils/passive_env_checker.py:241: DeprecationWarning: `np.bool8` is a deprecated alias for `np.bool_`.  (Deprecated NumPy 1.24)
  if not isinstance(terminated, (bool, np.bool8)):
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Разведочный анализ данных. Исследование
и визуализация данных» [Электронный ресурс] // GitHub. — 2019. — Режим доступа:
https://github.com/ugapanyuk/ml_course/wiki/LAB_EDA_VISUALIZATION (дата
обращения: 13.02.2019)

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