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


# Лабораторная работа № 4

## По курсу «методы машинного обучения в АСОИУ»

### «Реализация алгоритма Policy Interation»

**Выполнил:**

студент ИУ5-23М
Семенов И.А.

**Проверил:**

Гапанюк Ю.Е.

Подпись:

29.04.2024

Москва, 2024

## Описание задания

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).

**Задача, которую решает агент**

Агент должен найти оптимальную политику, которая максимизирует общую награду за время работы в среде.

В среде Taxi-v3 агент должен перемещать такси так, чтобы подбирать и доставлять пассажиров к месту назначения, максимизируя при этом награду. Алгоритм итерации политики помогает агенту найти оптимальную стратегию для выполнения этой задачи.

Задача агента в среде Taxi-v3

**Описание среды**

Среда Taxi-v3 представляет собой сетку 5x5, на которой агент (такси) может перемещаться в четырех направлениях: вверх, вниз, влево или вправо. В городе есть несколько возможных мест посадки и высадки пассажиров.

Цель: Агент должен управлять такси, чтобы подобрать пассажира в одном месте и отвезти его к месту назначения, получая награду за успешное выполнение задачи.

Действия: Агент может выбрать одно из пяти действий:

Вверх (0)

Вниз (1)

Влево (2)

Вправо (3)

Подобрать или высадить пассажира (4)

**Состояния**: Состояния определяют положение такси, местоположение пассажира и место назначения.

**Награда**: Агент получает положительную награду за успешную посадку и доставку пассажира к месту назначения и отрицательную награду (наказание) за неудачные действия, например, выход за границы карты.

Процесс работы кода

**Инициализация**: Агент начинает с случайной политики — стратегии, определяющей действия для каждого состояния.

**Цикл итерации политики:** Агент оценивает текущую политику, вычисляя функцию ценности, и улучшает политику на основе этой функции ценности. Цикл продолжается до достижения сходимости.

Тестирование: После достижения оптимальной политики агент тестирует свою стратегию, перемещая такси по городу и максимизируя награду.

**Итоги**

В результате код реализует алгоритм итерации политики для того, чтобы агент научился эффективно перемещать такси в среде Taxi-v3, доставляя пассажиров к месту назначения и максимизируя награду.

## Текст программы и экранные формы с примерами выполнения программы

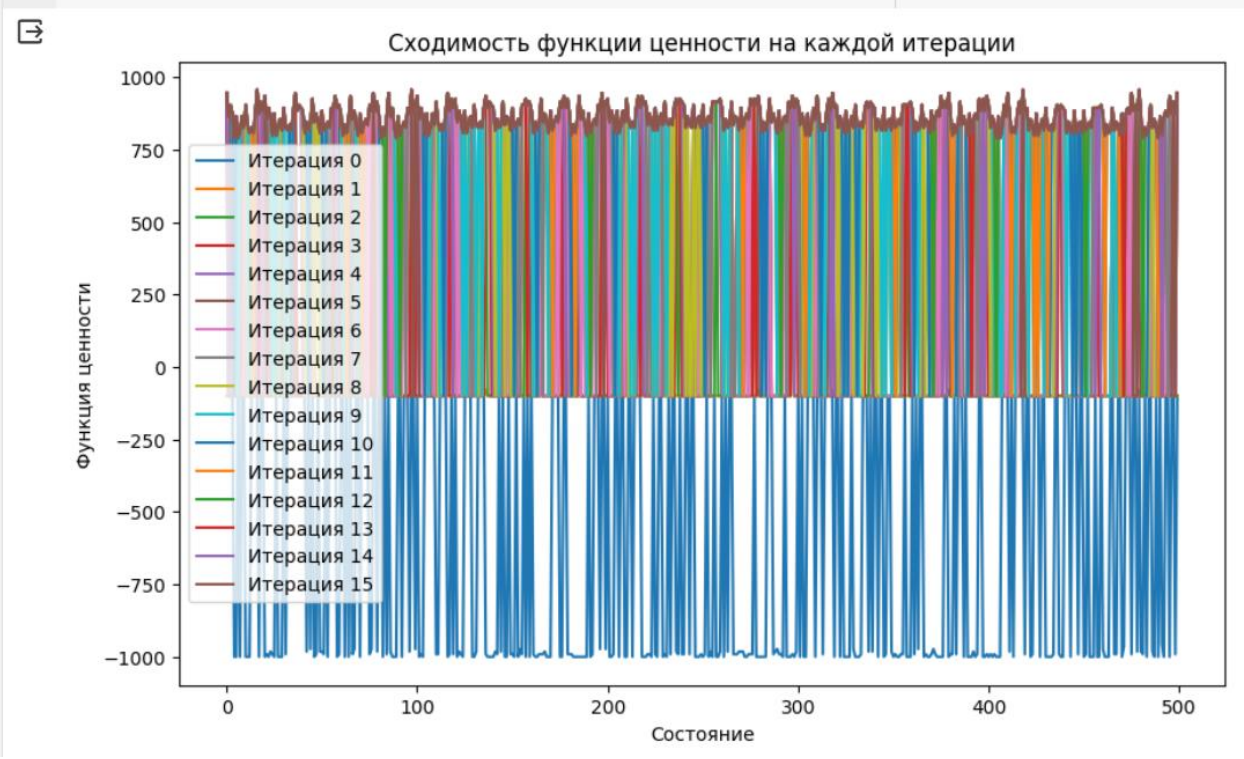
```python
import gym
import numpy as np

def policy_iteration(env, gamma=0.99, theta=1e-6):
    """
    Алгоритм итерации политики.

    Параметры:
    env (gym.Env): Среда обучения с подкреплением с дискретным пространством состояний и действий.
    gamma (float): Коэффициент дисконтирования.
    theta (float): Порог для сходимости значений.

    Возвращает:
    policy (np.array): Оптимальная политика.
    value (np.array): Оптимальная функция ценности.
    """
    # Инициализация случайной политики
    num_states = env.observation_space.n
    num_actions = env.action_space.n
    policy = np.random.choice(num_actions, num_states)

    def policy_evaluation(policy):
        """
        Оценка политики с использованием уравнения Беллмана.

        Параметры:
        policy (np.array): Текущая политика.

        Возвращает:
        value (np.array): Функция ценности.
        """
        value = np.zeros(num_states)
        while True:
            delta = 0
            for state in range(num_states):
                v = value[state]
                action = policy[state]
                value[state] = sum([prob * (reward + gamma * value[next_state])
                                    for prob, next_state, reward, done in env.P[state][action]])
                delta = max(delta, abs(v - value[state]))
            if delta < theta:
                break
        return value
```

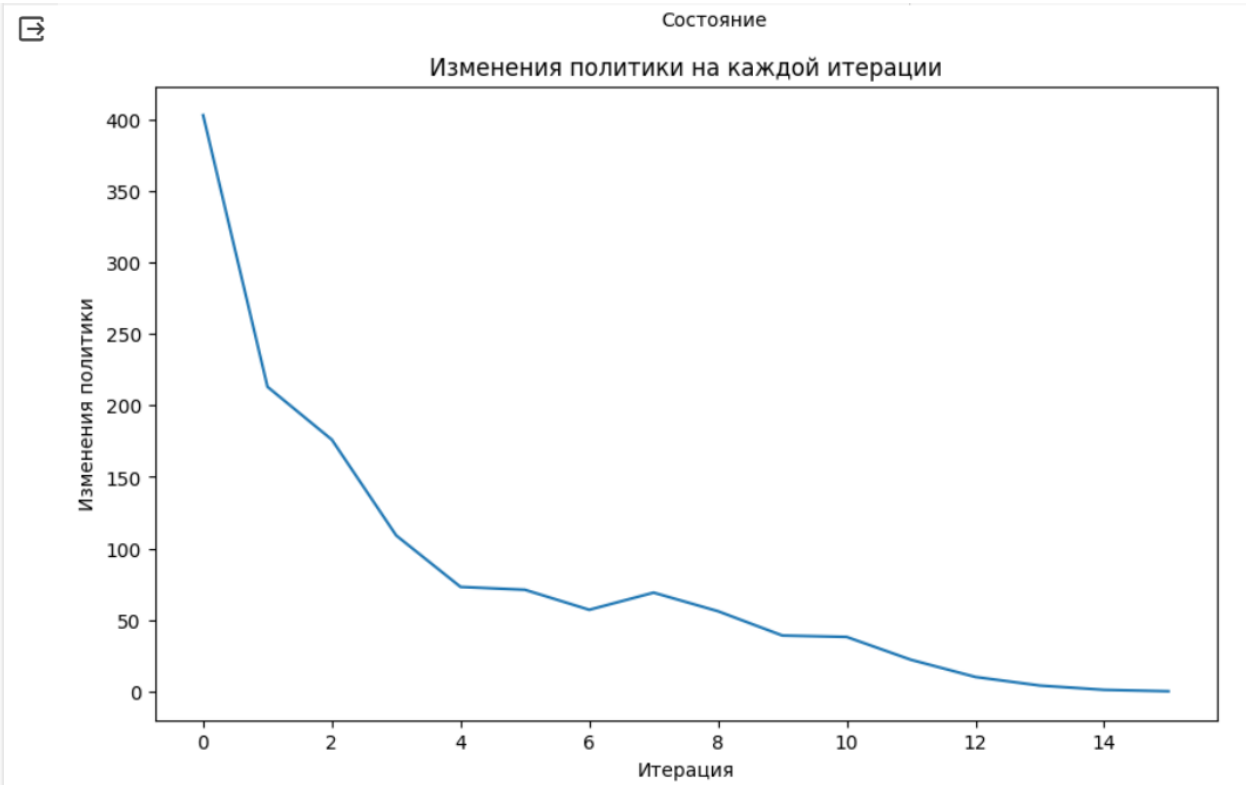
✓ 1 мин.

[8]

```
    def policy_improvement(value):
        """
        Улучшение политики на основе функции ценности.

        Параметры:
        value (np.array): Функция ценности.

        Возвращает:
        policy (np.array): Улучшенная политика.
        """
        policy_stable = True
        for state in range(num_states):
            old_action = policy[state]
            action_values = np.zeros(num_actions)

            # Расчет ожидаемой ценности для каждого действия
            for action in range(num_actions):
                action_values[action] = sum([prob * (reward + gamma * value[next_state])
                                             for prob, next_state, reward, done in env.P[state][action]])

            # Выбор действия с наивысшей ожидаемой ценностью
            policy[state] = np.argmax(action_values)

            # Проверка стабильности политики
            if old_action != policy[state]:
                policy_stable = False

        return policy_stable

    # Цикл итерации политики
    while True:
        # Оценка политики
        value = policy_evaluation(policy)

        # Улучшение политики
        if policy_improvement(value):
            break

    return policy, value

# Главный код для запуска алгоритма
if __name__ == "__main__":
    # Выберите подходящую среду из библиотеки gym (например, 'Taxi-v3' или 'CliffWalking-v0')
    env = gym.make('Taxi-v3')

    # Запуск алгоритма итерации политики
    optimal_policy, optimal_value = policy_iteration(env)
```

```
# Вывод оптимальной политики и функции ценности
print("Оптимальная политика:")
print(optimal_policy)

print("\nОптимальная функция ценности:")
print(optimal_value)

# Тестирование оптимальной политики в среде
state = env.reset()
total_reward = 0
done = False

while not done:
    action = optimal_policy[state]
    state, reward, done, _ = env.step(action)
    total_reward += reward
    env.render()

print(f"Общая награда: {total_reward}")
env.close()
```






```
Оптимальная политика:
[4 4 4 4 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 3
 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 4 4 4 4 0 0 0 0 0 0 0 0 0 0 5 0 0 1 1 1 1 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 2 2 2 2
 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 1
 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 2 2 2 2 0 0 0 0 2 2 2 2 1 2 0 2 1 1
 1 1 2 2 2 2 3 3 3 3 2 2 2 2 1 2 3 2 3 3 3 3 2 2 2 2 3 3 3 3 2 2 2 2 3 2 3
 2 3 3 3 3 2 2 2 2 3 3 3 3 0 0 0 0 3 2 3 0 3 3 3 3 1 1 1 1 3 3 3 3 0 0 0 0
 3 1 3 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 2 2 2 2 1 1 1 1 2
 2 2 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1
 1 1 0 0 0 0 1 2 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1
 1 4 4 4 4 1 1 1 1 1 1 5 1 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2 1 2 1 2 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1 4 4 4 4 1 2 1 5 1
 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 3]
```

**Вывод**

На основе полученных результатов можем сделать вывод о том, что алгоритм итерации политики в данном коде реализует метод для обучения агента в среде с подкреплением. Это алгоритм, который использует сочетание оценок текущей политики и ее улучшений для нахождения оптимальной политики, которая максимизирует суммарную награду за время работы агента в среде.