

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Лабораторная работа № 4
по дисциплине «Методы машинного обучения»

Тема: «Реализация алгоритма Policy Iteration.»

ИСПОЛНИТЕЛЬ:
группа ИУ5И-21М

Ли Яцзинь
ФИО

подпись "

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю .Е

Москва - 2024

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).

Мы определяем матрицу перехода состояний и матрицу функции вознаграждения, а также пишем код Python для реализации алгоритма итеративной оценки политики. Выполните 20 итераций функции значения состояния, выведите функцию значения после каждой итерации и проанализируйте ее сходимость.

1. Определите матрицу перехода состояний и матрицу функции вознаграждения

Мы используем простой пример, чтобы представить проблему MDP с 3 состояниями и 2 действиями.

Матрица перехода состояний представляет собой массив 3×3 , указывающий вероятность следующего состояния, соответствующего каждому состоянию и действию.

Матрица функции вознаграждения представляет собой массив 3×2 , представляющий немедленную награду, соответствующую каждому состоянию и действию.

```

transition_probs = np.array([
    [0.8, 0.2, 0.0], # 当前状态为0, 执行动作0, 下一个状态为0的概率为0.8, 为1的概率为0.2
    [0.0, 0.5, 0.5], # 当前状态为0, 执行动作1, 下一个状态为1的概率为0.5, 为2的概率为0.5
    [0.7, 0.3, 0.0], # 当前状态为1, 执行动作0, 下一个状态为0的概率为0.7, 为1的概率为0.3
    [0.0, 0.0, 1.0], # 当前状态为1, 执行动作1, 下一个状态为2的概率为1.0
    [0.0, 1.0, 0.0], # 当前状态为2, 执行动作0, 下一个状态为1的概率为1.0
    [0.0, 0.0, 1.0]]) # 当前状态为2, 执行动作1, 下一个状态为2的概率为1.0

rewards = np.array([
    [5, 0], # 当前状态为0, 执行动作0, 即时奖励为5
    [0, 0], # 当前状态为0, 执行动作1, 即时奖励为0
    [0, 10]]) # 当前状态为1, 执行动作0, 即时奖励为0; 当前状态为1, 执行动作1, 即时奖励为10

```

Рисунок 1- Код шаг1

2.Написание итеративного алгоритма оценки политики

```

def policy_evaluation(transition_probs, rewards, policy, discount_factor=1.0, theta=0.00001, max_iterations=20):
    """
    迭代策略评估算法
    """
    num_states, num_actions, _ = np.shape(transition_probs)
    V = np.zeros(num_states) # 初始化状态值函数为0

    for _ in range(max_iterations):
        delta = 0
        for s in range(num_states):
            v = 0
            for a in range(num_actions):
                for next_s in range(num_states):
                    v += policy[s][a] * transition_probs[s][a][next_s] * (rewards[s][a] + discount_factor * V[next_s])
            delta = max(delta, np.abs(v - V[s]))
            V[s] = v
        if delta < theta:
            break
    return V

```

Рисунок 2.-Код шаг2

3.Определите начальную стратегию, запустите алгоритм оценки итеративной стратегии и выведите функцию значения состояния после каждой итерации.

```


return V

# 定义初始策略
# 这里简单地假设每个状态下都采取相同的策略, 即均匀随机选择动作
num_states, num_actions, _ = np.shape(transition_probs)
policy = np.ones([num_states, num_actions]) / num_actions

# 运行迭代策略评估算法, 输出每次迭代后的状态值函数
for i in range(1, 21):
    V = policy_evaluation(transition_probs, rewards, policy, max_iterations=i)
    print(f"Iteration {i}:")
    print("Value Function:", V)
    print(f"-----")

```

Рисунок 3-Код шаг2



```
Iteration 1:
Value Function: [2.5    0.875  5.4375]

Iteration 2:
Value Function: [ 5.165625   4.65796875 10.04773438]

Iteration 3:
Value Function: [ 8.70847266  8.77052793 14.40913115]

Iteration 4:
Value Function: [12.65535663 12.94951958 18.67932537]

Iteration 5:
Value Function: [16.76430585 17.14959767 22.91446152]

Iteration 6:
Value Function: [20.9366969  21.35751433 27.13598792]

Iteration 7:
Value Function: [25.13380576 25.56845312 31.35222052]

Iteration 8:
Value Function: [29.34053603 29.78056584 35.56639318]

Iteration 9:
Value Function: [33.55101075 33.99313523 39.77976421]

Iteration 10:
Value Function: [37.76294268 38.20588233 43.99282327]

Iteration 11:
Value Function: [41.9754417  42.41869858 48.20576092]
```

Рисунок 4.-Функция значения состояния после каждой итерации