



Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления» – ИУ5

Факультет «Информатика и системы управления» – ИУ

Отчет по лабораторной работе №4
Курса Методы машинного обучения в АСОИУ
" Реализация алгоритмов обучения с подкреплением "

9

(количество листов)

Исполнитель

студент группы ИУ5-21м

Нижаметдинов М. Ш.

“ ____ ” _____ 2025 г.

Проверил

доцент кафедры ИУ5

Гапанюк Ю. Е.

“ ____ ” _____ 2025 г.

Москва, 2025 г.

Цель лабораторной работы: ознакомление с базовыми методами обучения с подкреплением.

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).

Ход работы

1. Подготовка

Создали виртуальное окружение:

```
python -m venv env
```

Перешли в него:

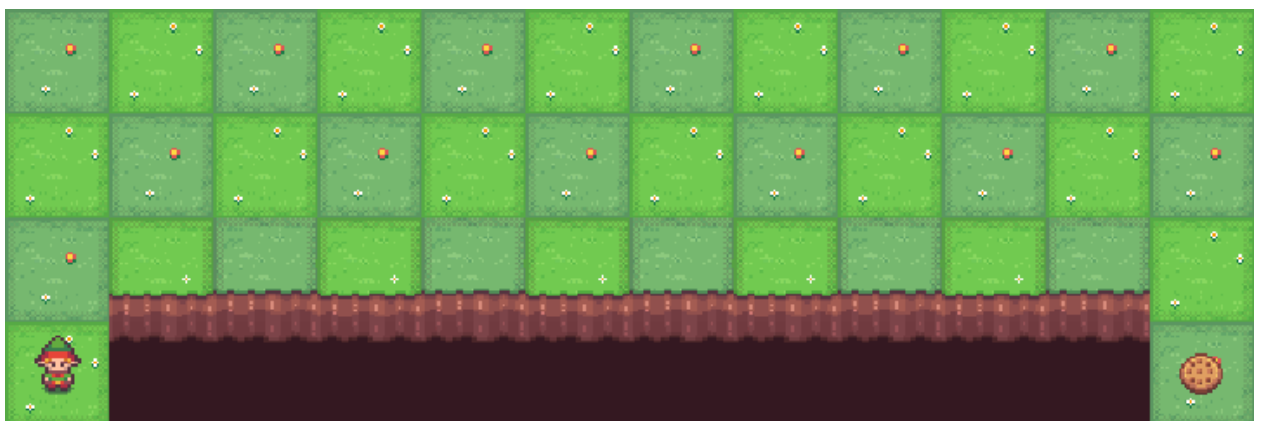
```
env/Scripts/activate.
```

Установили зависимости:

```
pip install -r .\examples\requirements.txt
```

2. Описание среды

Будем работать со средой Cliff Walking:



Поле представляет собой матрицу 4x12. Агент начинает проходить карту с ячейки [3, 0] (левый нижний угол). Ему необходимо достичь ячейки [3, 11], т.е. цель размещена в правом нижнем углу. Также агенту нельзя

наступать на обрыв – это ячейки [3, 1...10] (внизу, по центру). Если агент наступит на обрыв, он вернется к началу. Эпизод заканчивается, когда агент достигает цели.

Агент может совершить 4 действия:

- 0: переместиться вверх;
- 1: передвинуться вправо;
- 2: передвинуться вниз;
- 3: передвинуться влево.

За каждый шаг полагается -1 награда, а за шаг в обрыв — -100.

Выведем информацию о наборе с помощью следующего кода:

```
state, action = 0, 0
env = gym.make("CliffWalking-v0")
print('Пространство состояний:')
pprint(env.observation_space)
print()
print('Пространство действий:')
pprint(env.action_space)
print()
print('Вероятности для 0 состояния:')
pprint(env.P[state])
print('Вероятности для 46го состояния:')
pprint(env.P[46])
```

Вывод:

```
(env) PS C:\Users\Mi\my_documents\master_program\sem2\MMO\lab4> python .\cliff_walking_info.py
Пространство состояний:
Discrete(48)

Пространство действий:
Discrete(4)

Вероятности для 0 состояния:
{0: [(1.0, 0, -1, False)],
 1: [(1.0, 1, -1, False)],
 2: [(1.0, 12, -1, False)],
 3: [(1.0, 0, -1, False)]}
Вероятности для 34го состояния:
{0: [(1.0, 22, -1, False)],
 1: [(1.0, 35, -1, False)],
 2: [(1.0, 36, -100, False)],
 3: [(1.0, 33, -1, False)]}
Вероятности для 35го состояния:
{0: [(1.0, 23, -1, False)],
 1: [(1.0, 35, -1, False)],
 2: [(1.0, 47, -1, True)],
 3: [(1.0, 34, -1, False)]}
```

В итоге получаем: размерность пространства состояний 48 (это размерность поля 4x12), пространства действий 4 (вверх, право, вниз, влево).

Также в качестве примера просмотрели нулевое, 34ое и 35ое состояния из матрицы состояний env.P. Матрица env.P состоит из 48 строк (состояния от 0го до 47). Каждая строка состоит из объекта с 4мя действиями. Для каждого действия указан массив возможных состояний, в которое можно перейти из текущего состояния. Значение флага в конце массива равное True означает достижение цели. Формат строки:

```
{action: [(probability, nextstate, reward, done)]}
```

Таким образом, находясь в нулевом состоянии:

- можем остаться в нем при действиях 0 (вверх) и 3 (влево), т.к. врежемся в стенку карты;

- можем переместиться в состояние 1 при действии 1 (вправо);
- можем переместиться в состояние 12 при действии 2 (вниз);

Находясь в 34ом состоянии:

- переместимся в другое состояние при действиях 0, 1, 3;
- попадем в обрыв (-100 к награде) при действии 2 (вниз).

Находясь в 35ом состоянии:

- останемся в 35ом при действии 1 (вправо);
- переместимся в другое состояние при действиях 0, 3;
- достигнем цели (состояния 37) – флаг `done=True`

3. Реализация алгоритма

Произведем обучение с подкреплением для нашей модели. Для этого реализуем класс `PolicyIterationAgent`, эмулирующий работу агента.

Инициализируем класс, задав пространство состояний (`observation_dim = 48`), действия (0-3), политику попыток (25%, что выполнится одно из 4ех действий), начальные значения состояний (`state_values`), максимальное число итераций (10000) и начальные значения параметров `theta`, `gamma`:

```
def __init__(self, env):
    self.env = env
    self.observation_dim = 48
    self.actions_variants = np.array([0,1,2,3])
    self.policy_probs = np.full((self.observation_dim,
len(self.actions_variants)), 0.25)
    self.state_values = np.zeros(shape=(self.observation_dim))
    self.maxNumberOfIterations = 10000
    self.theta=1e-6
    self.gamma=0.99
```

Также добавим в класс метод вывода политики:

```
def print_policy(self):
    print('Стратегия:')
    pprint(self.policy_probs)
```

Метод оценивания стратегии:

```
def policy_evaluation(self):
    valueFunctionVector = self.state_values
    for iterations in range(self.maxNumberOfIterations):
        valueFunctionVectorNextIteration=np.zeros(shape=(self.ob
servation_dim))
        for state in range(self.observation_dim):
            action_probabilities = self.policy_probs[state]
            outerSum=0
            for action, prob in enumerate(action_probabilities):
```

```

        innerSum=0
        for probability, next_state, reward,
isTerminalState in self.env.P[state][action]:
            innerSum=innerSum+probability*(reward+self.g
amma*self.state_values[next_state])
        outerSum=outerSum+self.policy_probs[state][actio
n]*innerSum
        valueFunctionVectorNextIteration[state]=outerSum
        if (np.max(np.abs(valueFunctionVectorNextIteration-
valueFunctionVector))<self.theta):
            valueFunctionVector=valueFunctionVectorNextIteration
            break
        valueFunctionVector=valueFunctionVectorNextIteration
    return valueFunctionVector

```

Метод для улучшения стратегии:

```

def policy_improvement(self):
    qvaluesMatrix=np.zeros((self.observation_dim,
len(self.actions_variants)))
    improvedPolicy=np.zeros((self.observation_dim,
len(self.actions_variants)))
    for state in range(self.observation_dim):
        for action in range(len(self.actions_variants)):
            for probability, next_state, reward, isTerminalState
in self.env.P[state][action]:
                qvaluesMatrix[state,action]=qvaluesMatrix[state,
action]+probability*(reward+self.gamma*self.state_values[next_st
ate])

                bestActionIndex=np.where(qvaluesMatrix[state,:]==np.max(
qvaluesMatrix[state,:]))
                improvedPolicy[state,bestActionIndex]=1/np.size(bestActi
onIndex)
    return improvedPolicy

```

Реализация алгоритма:

```

def policy_iteration(self, cnt):
    policy_stable = False
    for i in range(1, cnt+1):
        self.state_values = self.policy_evaluation()
        self.policy_probs = self.policy_improvement()
    print(f'Алгоритм выполнен за {i} шагов.')

```

Проигрывание сцены:

```
def play_agent(agent):
    env2 = gym.make('CliffWalking-v0', render_mode='human')
    state = env2.reset()[0]
    done = False
    while not done:
        p = agent.policy_probs[state]
        if isinstance(p, np.ndarray):
            action = np.random.choice(len(agent.actions_variants),
p=p)
        else:
            action = p
        next_state, reward, terminated, truncated, _ =
env2.step(action)
        env2.render()
        state = next_state
        if terminated or truncated:
            done = True
```

Теперь создадим основную функцию для выполнения и запустим скрипт:

```
def main():
    # Создание среды
    env = gym.make('FrozenLake-v1')
    env.reset()
    # Обучение агента
    agent = PolicyIterationAgent(env)
    agent.print_policy()
    agent.policy_iteration(10000)
    agent.print_policy()
    # Проигрывание сцены для обученного агента
    play_agent(agent)
```

Наш агент долго блуждал по карте, но в итоге достиг цели. Получили следующий массив стратегии

остальных 0,33). При 36ом состоянии невыгодно двигаться вправо (второе значения массива – 0, остальные 0,33). Находясь в обрыве (хотя это невозможно) в состоянии 37 – нужно двигаться влево или вверх, в состояниях 38-45 – только вверх, в состоянии 46 – вправо или вверх. Находясь в состоянии 47, нужно не двигаться влево (в обрыв).