Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

Отчёт к лабораторным работам по курсу

«Методы машинного обучения»

Лабораторная работа №4 «Реализация алгоритма Policy Iteration.»

Выполнил:

студент(ка) группы ИУ5И-21М Лю Бэйбэй

подпись, дата

Проверил:

к.т.н., доц., Виноградовой М.В.

подпись, дата

Москва, 2022 г.

1. 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).

2. Текст программы и экранные формы с примерами выполнения программы.

Импортирование необходимых библиотек.

```
try:
    import lspi
except ImportError:
    !pip install git+https://github.com/qdevpsi3/rl-lspi.git
    import lspi
```

```
import numpy as np
import random
```

```
def hamming(agent, optimal_policy):
        nS = agent.env.observation_space.n
        agent_policy = np.array([agent.predict(s) for s in range(nS)])
        dist = np.sum(optimal_policy != agent_policy)
        return dist
# build the environment
nS = 4
env = lspi.envs.ChainWalkEnv(nS)

# build the agent
degree = 2
preprocess_obs = lambda x: x + 1
agent = lspi.agents.PolynomialAgent(env, degree, preprocess_obs)

# build the trainer
gamma = 0.9
memory_size = 500
memory_type = 'sample'
eval_type = 'sherman_morrison'
baseline = lspi.baselines.LSPolicyIteration(env, agent, gamma, memory_size,
                                            memory_type, eval_type)

# define optimal policy
optimal_policy = np.array([1, 1, 0, 0])

# build the memory
baseline.init_memory()

# run the algorithm
n_iter = 5
```

```
dist = hamming(agent, optimal_policy)
print('iteration = {:02d} - distance to optimal policy : {}'.format(0, dist))
for it in range(1, n_iter + 1):
    baseline.train_step()
    dist = hamming(agent, optimal_policy)
    print('iteration = {:02d} - distance to optimal policy : {}'.format(
        it, dist))
```

```
iteration = 00 - distance to optimal policy : 2
iteration = 01 - distance to optimal policy : 1
iteration = 02 - distance to optimal policy : 0
iteration = 03 - distance to optimal policy : 0
iteration = 04 - distance to optimal policy : 0
iteration = 05 - distance to optimal policy : 0
```