|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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.



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))

