ЛАБОРАТОРНАЯ РАБОТА № 5

«ИССЛЕДОВАНИЕ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ»

Цель работы

Исследование методов недетерминированного поиска решений задач, приобретение навыков программирования интеллектуальных агентов, функционирующих в недетерминированных средах, исследование методов построения агентов, обучаемых на основе алгоритмов обучения с подкреплением.

Ход работы

Для выполнения задачи 1 «Итерация по значениям» был написан код для методов runValueIteration, computeActionFromValues и computeQValueFromValues в файле valueIterationAgents.py.

Листинг 1 – Код runValueIteration

def runValueIteration(self):

for i in range(self.iterations):

states = self.mdp.getStates()

temp\_counter = util.Counter()

for state in states:

max\_val = float("-inf")

for action in self.mdp.getPossibleActions(state):

q\_value = self.computeQValueFromValues(state, action)

if q\_value > max\_val:

max\_val = q\_value

temp\_counter[state] = max\_val

self.values = temp\_counter

Листинг 2 – Код computeActionFromValues

def computeActionFromValues(self, state):

best\_action = None

max\_val = float("-inf")

for action in self.mdp.getPossibleActions(state):

q\_value = self.computeQValueFromValues(state, action)

if q\_value > max\_val:

max\_val = q\_value

best\_action = action

return best\_action

Листинг 3 – Код computeQValueFromValues

def computeQValueFromValues(self, state, action):

action\_prob\_pairs = self.mdp.getTransitionStatesAndProbs(state, action)

total = 0

for next\_state, prob in action\_prob\_pairs:

reward = self.mdp.getReward(state, action, next\_state)

total += prob \* (reward + self.discount \* self.values[next\_state])

return total

util.raiseNotDefined()

Результат выполнения autograder –q q1 представлен на рисунке 1.

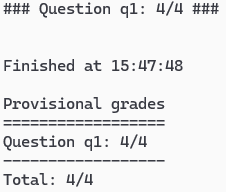


Рисунок 1 – Результат autograder –q q1.

Для решения задачи 2 «анализ перехода через мост» был помещён свой код в функцию question2 в файле analysis.py.

Листинг 4 – Код question2

def question2():

answerDiscount = 0.9

answerNoise = 0.002

return answerDiscount, answerNoise

Результат выполнения autograder –q q2 представлен на рисунке 2.

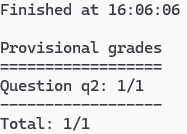


Рисунок 2 – Результат выполнения autograder –q q2

Для выполнения задачи 3 «Реализация политик» был внесен код в методы от question3a до question3e в файле analysis.py.

Листинг 5 – Код методов от question3a до question3e

def question3a():

answerDiscount = 0.5

answerNoise = 0

answerLivingReward = -1

return answerDiscount, answerNoise, answerLivingReward

# If not possible, return 'NOT POSSIBLE'

def question3b():

answerDiscount = 0.2

answerNoise = 0.2

answerLivingReward = -1

return answerDiscount, answerNoise, answerLivingReward

# If not possible, return 'NOT POSSIBLE'

def question3c():

answerDiscount = 0.95

answerNoise = 0

answerLivingReward = -1

return answerDiscount, answerNoise, answerLivingReward

# If not possible, return 'NOT POSSIBLE'

def question3d():

answerDiscount = 0.2

answerNoise = 0.5

answerLivingReward = 2

return answerDiscount, answerNoise, answerLivingReward

# If not possible, return 'NOT POSSIBLE'

def question3e():

answerDiscount = 0.95

answerNoise = 0

answerLivingReward = 1000

return answerDiscount, answerNoise, answerLivingReward

Результат выполнения autograder –q q3 представлен на рисунке 3.

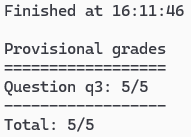


Рисунок 3 – Результат выполнения autograder –q q3

Для решения задачи 4 «Асинхронная итерация по значениям» был реализован класс AsynchronousValueIterationAgent из файла valueIterationAgents.py.

Листинг 6 – Код класса AsynchronousValueIterationAgent

class AsynchronousValueIterationAgent(ValueIterationAgent):

def \_\_init\_\_(self, mdp, discount = 0.9, iterations = 1000):

ValueIterationAgent.\_\_init\_\_(self, mdp, discount, iterations)

def runValueIteration(self):

states = self.mdp.getStates()

num\_states = len(states)

for i in range(self.iterations):

state = states[i % num\_states]

if not self.mdp.isTerminal(state):

values = []

for action in self.mdp.getPossibleActions(state):

q\_value = self.computeQValueFromValues(state, action)

values.append(q\_value)

self.values[state] = max(values)

Результат выполнения autograder –q q4 представлен на рисунке 4.

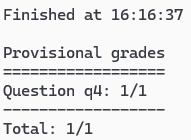


Рисунок 4 – Результат выполнения autograder –q q4

Для решения задачи 5 «Итерации по приоритетным значениям» был реализован класс PrioritizedSweepingValueIterationAgent из файла valueIterationAgents.py.

Листинг 7 – Код класса PrioritizedSweepingValueIterationAgent

class PrioritizedSweepingValueIterationAgent(AsynchronousValueIterationAgent):

def \_\_init\_\_(self, mdp, discount = 0.9, iterations = 100, theta = 1e-5):

self.theta = theta

ValueIterationAgent.\_\_init\_\_(self, mdp, discount, iterations)

def runValueIteration(self):

pq = util.PriorityQueue()

predecessors = {}

for state in self.mdp.getStates():

if not self.mdp.isTerminal(state):

for action in self.mdp.getPossibleActions(state):

for nextState, prob in self.mdp.getTransitionStatesAndProbs(state, action):

if nextState in predecessors:

predecessors[nextState].add(state)

else:

predecessors[nextState] = {state}

for state in self.mdp.getStates():

if not self.mdp.isTerminal(state):

values = []

for action in self.mdp.getPossibleActions(state):

q\_value = self.computeQValueFromValues(state, action)

values.append(q\_value)

diff = abs(max(values) - self.values[state])

pq.update(state, - diff)

for i in range(self.iterations):

if pq.isEmpty():

break

temp\_state = pq.pop()

if not self.mdp.isTerminal(temp\_state):

values = []

for action in self.mdp.getPossibleActions(temp\_state):

q\_value = self.computeQValueFromValues(temp\_state, action)

values.append(q\_value)

self.values[temp\_state] = max(values)

for p in predecessors[temp\_state]:

if not self.mdp.isTerminal(p):

values = []

for action in self.mdp.getPossibleActions(p):

q\_value = self.computeQValueFromValues(p, action)

values.append(q\_value)

diff = abs(max(values) - self.values[p])

if diff > self.theta:

pq.update(p, -diff)

Результат выполнения autograder –q q5 представлен на рисунке 5.

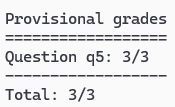


Рисунок 5 – Результат выполнения autograder –q q5

Для решения задачи 6 «Q-обучение» были реализованы методы update, computeValueFromQValues, getQValue и computeActionFromQValues из файла qlearningAgents.py.

Листинг 8 – Код методов update, computeValueFromQValues, getQValue и computeActionFromQValues

def getQValue(self, state, action):

return self.q\_values[(state, action)]

#util.raiseNotDefined()

def computeValueFromQValues(self, state):

legalActions = self.getLegalActions(state)

if len(legalActions)==0:

return 0.0

tmp = util.Counter()

for action in legalActions:

tmp[action] = self.getQValue(state, action)

return tmp[tmp.argMax()]

#util.raiseNotDefined()

def computeActionFromQValues(self, state):

actions = self.getLegalActions(state)

best\_action = None

max\_val = float('-inf')

for action in actions:

q\_value = self.q\_values[(state, action)]

if max\_val < q\_value:

max\_val = q\_value

best\_action = action

return best\_action

#util.raiseNotDefined()

def update(self, state, action, nextState, reward):

old\_q\_value = self.getQValue(state, action)

old\_part = (1 - self.alpha) \* old\_q\_value

reward\_part = self.alpha \* reward

if not nextState:

self.q\_values[(state, action)] = old\_part + reward\_part

else:

nextState\_part = self.alpha \* self.discount \* self.getValue(nextState)

self.q\_values[(state, action)] = old\_part + reward\_part + nextState\_part

Результат выполнения autograder –q q6 представлен на рисунке 6.

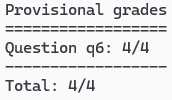


Рисунок 6 – Результат выполнения autograder –q q6

Для решения задачи 7 «Эпсилон-жадная стратегия» был дописан метод getAction из файла qlearningAgents.py.

Листинг 9 – Код getAction

def getAction(self, state):

legalActions = self.getLegalActions(state)

action = None

legal\_actions = self.getLegalActions(state)

explore = util.flipCoin(self.epsilon)

if explore:

return random.choice(legal\_actions)

else:

return self.getPolicy(state)

#util.raiseNotDefined()

return action

Результат выполнения autograder –q q7 представлен на рисунке 7.

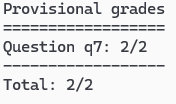


Рисунок 7 – Результат выполнения autograder –q q7

Для решения задачи 8 «Переход по мосту» был написан метод question8 из файла analysis.py.

Листинг 10 – Код question8

def question8():

answerEpsilon = None

answerLearningRate = None

return 'NOT POSSIBLE'

Результат выполнения autograder –q q8 представлен на рисунке 8.

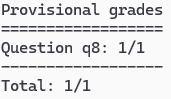


Рисунок 8 – Результат выполнения autograder –q q8

Для решения задачи 9 «Q-обучение и Пакман» был реализован класс PacmanQAgent из файла qLearningAgent.py.

Листинг 11 – Код PacmanQAgent

class PacmanQAgent(QLearningAgent):

def \_\_init\_\_(self, epsilon=0.05,gamma=0.8,alpha=0.2, numTraining=0, \*\*args):

args['epsilon'] = epsilon

args['gamma'] = gamma

args['alpha'] = alpha

args['numTraining'] = numTraining

self.index = 0 # This is always Pacman

QLearningAgent.\_\_init\_\_(self, \*\*args)

def getAction(self, state):

action = QLearningAgent.getAction(self,state)

self.doAction(state,action)

return action

Результат выполнения autograder –q q9 представлен на рисунке 9.

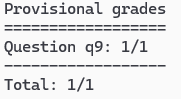


Рисунок 9 – Результат выполнения autograder –q q9

Для решения задачи 10 «q-обучение с аппроксимацией» был реализован класс ApproximateQAgent из файла qLearningAgent.py.

Листинг 12 – Код ApproximateQAgent

class ApproximateQAgent(PacmanQAgent):

def \_\_init\_\_(self, extractor='IdentityExtractor', \*\*args):

self.featExtractor = util.lookup(extractor, globals())()

PacmanQAgent.\_\_init\_\_(self, \*\*args)

self.weights = util.Counter()

def getWeights(self):

return self.weights

def getQValue(self, state, action):

features = self.featExtractor.getFeatures(state, action)

total = 0

for i in features:

total += features[i] \* self.weights[i]

return total

def update(self, state, action, nextState, reward):

diff = (reward + self.discount \* self.getValue(nextState)) - self.getQValue(state, action)

features = self.featExtractor.getFeatures(state, action)

for i in features:

self.weights[i] = self.weights[i] + self.alpha \* diff \* features[i]

def final(self, state):

PacmanQAgent.final(self, state)

if self.episodesSoFar == self.numTraining:

pass

Результат выполнения autograder –q q10 представлен на рисунке 10.

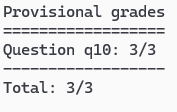


Рисунок 10 – Результат выполнения autograder –q q10

Выводы

В ходе выполнения данной лабораторной работы были исследованы методы недетерминированного поиска решений задач, приобретены навыки программирования интеллектуальных агентов, функционирующих в недетерминированных средах, исследованы методы построения агентов, обучаемых на основе алгоритмов обучения с подкреплением.