**Virtual Initio Programming: WS24 & Ex24 Sample Answers**

**WS24**

**Exercise 1:** As with the previous worksheet, the lower epsilon\_reduce and learning\_rate are, the more likely the program is to learn the correct algorithm but the longer it will take. These values seem to work most of the time and its relatively quick.

import simclient.simrobot as initio

import time, random

initio.init()

actions = ['forward','backward','left','right']

def action\_reward(action\_list, default):

action\_rewards = {}

for i in range(0, 2):

for j in range(0, 2):

for k in (action\_list):

action\_rewards[((i, j), k)] = default

return action\_rewards

def execute\_action(action):

if (action == "forward"):

initio.forward(10)

elif (action == "backward"):

initio.reverse(10)

elif (action == "left"):

initio.spinLeft(10)

elif (action == "right"):

initio.spinRight(10)

else:

initio.stop()

time.sleep(3)

def best\_action(state):

max\_reward = 0

for act in actions:

if (reward\_dictionary[(state, act)] > max\_reward):

action = act

max\_reward = reward\_dictionary[(state, act)]

return action

reward\_dictionary = action\_reward(actions, 1)

rewards = {(1, 1):1, (1, 0):2, (0, 1):0, (0, 0):1}

epsilon = 1

epsilon\_reduce = 0.05

learning\_rate = 0.5

while (epsilon > 0):

explore = random.random()

state = (initio.irLeftLine(), initio.irRightLine())

if (explore < epsilon):

action = random.choice(actions)

print("Random Action: " + action)

else:

action = best\_action(state)

print("Best Action: " + action)

execute\_action(action)

reward = rewards[(initio.irLeftLine(), initio.irRightLine())]

reward\_dictionary[(state, action)] = reward\_dictionary[(state, action)] + (reward - reward\_dictionary[(state, action)])\*learning\_rate

if (reward == 2):

epsilon = epsilon - epsilon\_reduce

print("New epsilon: " + str(epsilon))

initio.stop()

print(reward\_dictionary)

for key in reward\_dictionary:

print("Average reward for " + str(key) + " is " + str(reward\_dictionary[key]))

**Ex24**

**Exercise 1:** I did some tweaking of the speed epsilon reduced to get this working. Students may also want to experiment with the learning rate and the rewards. Depending on the set up of the testing environment the robot may just learn to spin on the spot.

import simclient.simrobot as initio

import time, random

initio.init()

actions = ['forward','left','right']

def action\_reward(action\_list, default):

action\_rewards = {}

for i in range(0, 2):

for j in range(0, 2):

for k in range(0, 2):

for a in (action\_list):

action\_rewards[((i, j, k), a)] = default

return action\_rewards

def execute\_action(action):

if (action == "forward"):

initio.forward(10)

elif (action == "left"):

initio.spinLeft(10)

elif (action == "right"):

initio.spinRight(10)

else:

initio.stop()

time.sleep(3)

def best\_action(state):

max\_reward = 0

for act in actions:

if (reward\_dictionary[(state, act)] > max\_reward):

action = act

max\_reward = reward\_dictionary[(state, act)]

return action

reward\_dictionary = action\_reward(actions, 1)

rewards = {(0, 0, 0):2, (0, 0, 1):1, (0, 1, 0):1, (0, 1, 1):1, (1, 0, 0):0, (1, 0, 1):0, (1, 1, 0):0, (1, 1, 1):0}

epsilon = 1

epsilon\_reduce = 0.02

learning\_rate = 0.5

while (epsilon > 0):

explore = random.random()

state = (initio.getDistance() < 20, initio.irLeft(), initio.irRight())

if (explore < epsilon):

action = random.choice(actions)

print("Random Action: " + action)

else:

action = best\_action(state)

print("Best Action: " + action)

execute\_action(action)

reward = rewards[(initio.irCentre(), initio.irLeft(), initio.irRight())]

reward\_dictionary[(state, action)] = reward\_dictionary[(state, action)] + (reward - reward\_dictionary[(state, action)])\*learning\_rate

if (reward == 2):

epsilon = epsilon - epsilon\_reduce

print("New epsilon: " + str(epsilon))

initio.stop()

print(reward\_dictionary)

for key in reward\_dictionary:

print("Average reward for " + str(key) + " is " + str(reward\_dictionary[key]))

**Exercise 2:** I spent some time experimenting with the rewards before I got this working. I have left print statements used for figuring this out in.

import simclient.simrobot as initio

import time, random

initio.init()

actions = ['forward','left','right']

def action\_reward(action\_list, default):

action\_rewards = {}

for i in range(0, 2):

for j in range(0, 2):

for k in range(0, 2):

for a in (action\_list):

action\_rewards[((i, j, k), a)] = default

return action\_rewards

def execute\_action(action):

if (action == "forward"):

initio.forward(10)

elif (action == "left"):

initio.spinLeft(10)

elif (action == "right"):

initio.spinRight(10)

else:

initio.stop()

time.sleep(3)

def best\_action(state):

max\_reward = 0

for act in actions:

if (reward\_dictionary[(state, act)] > max\_reward):

action = act

max\_reward = reward\_dictionary[(state, act)]

return action

reward\_dictionary = action\_reward(actions, 1)

rewards = {(0, 0, 0):3, (0, 0, 1):4, (0, 1, 0):0, (0, 1, 1):4, (1, 0, 0):3, (1, 0, 1):3, (1, 1, 0):2, (1, 1, 1):2}

epsilon = 1

epsilon\_reduce = 0.02

learning\_rate = 0.5

while (epsilon > 0):

explore = random.random()

state = (initio.getDistance() < 20, initio.irLeft(), initio.irRight())

print(str(state))

if (explore < epsilon):

action = random.choice(actions)

print("Random Action: " + action)

else:

action = best\_action(state)

print("Best Action: " + action)

execute\_action(action)

reward = rewards[(initio.getDistance() < 20, initio.irLeft(), initio.irRight())]

print(str(reward))

reward\_dictionary[(state, action)] = reward\_dictionary[(state, action)] + (reward - reward\_dictionary[(state, action)])\*learning\_rate

if (reward == 4):

epsilon = epsilon - epsilon\_reduce

print("New epsilon: " + str(epsilon))

initio.stop()

print(reward\_dictionary)

for key in reward\_dictionary:

print("Average reward for " + str(key) + " is " + str(reward\_dictionary[key]))



University of Liverpool, 2020

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).