Caden Kroonenberg

Problem Set 9

EECS 649

4-9-22

**1**

EMV = ($10) + ($1,000,000) + ($0) = $0.70

EU(Accept) = U(Sk +10) + U(Sk + 1,000,000) + U(Sk)

= U(Sk +1) + U(Sk + 1,000,000)

EU(Reject) = U(Sk+1)

Accept if U(Sk +1) + U(Sk + 1,000,000) > U(Sk+1)

Accept if U(Sk + 1,000,000) > U(Sk+1)

Accept if U(Sk + 1,000,000) > 1,600,000\*U(Sk+1)

**2**

1. EMV = ($2) + ($22) + ($23) + … + ($2n)

= =∞. 2n grows at a faster rate than n.

1. $5
2. EU(Accept) = U(Sc-c+2) + U(Sc-c+4) + … + U(Sc-c+N)

= (alog2(2) + b) + (alog2(4) + b) + … (alog2(N) + b)

=

=

= 2a + b

EU(Reject) = U(Sc) = alog2(c) + b

1. 2a + b > alog2(c) + b

= 2a > alog2(c)

= 2 > log2(c)

= 22 > 2log2(c)

= 4 > c

\*assuming a > 0

You should play the game if your current wealth is less than $4.

**3**

a. Diagram

Description automatically generated

b. Gain = 0.7($500) + 0.3(-$200) = $290

c.

P(q | pass) = P(pass | q)P(q)/P(pass)

= α\*0.8\*0.7

= α\*0.56/(0.56α +0.105α)

= 0.842

P(-q | pass) = P(pass | -q)P(-q)/P(pass)

= α\*0.35\*0.3

= α\*0.105/(0.56α +0.105α)

= 0.158

P(pass) = P(pass | q)P(q)/P(q | pass)

= 0.8\*0.7/0.842

= 0.665

P(-pass) = 1-P(pass)

= 0.335

P(q | -pass) = P(-pass | q)P(q)/P(-pass)

= 0.2\*0.7/0.335

= 0.418

P(-q | -pass) = P(-pass | -q)P(-q)/P(-pass)

= 0.65\*0.3/0.335

= 0.582

d.

EU(buy used | pass) = 0.842($450) + 0.158(-$250) = $339.40

EU(buy new | pass) = -$50

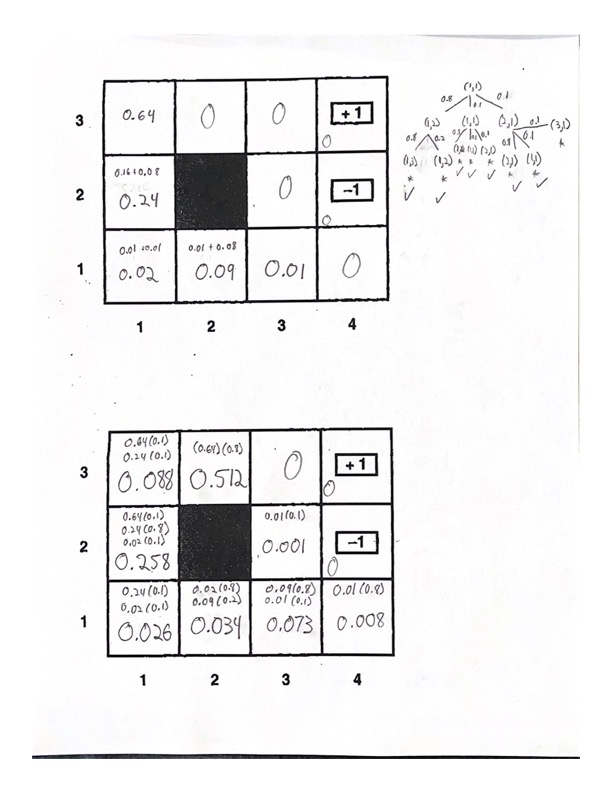
Action = argmaxa = buy used

EU(buy used | fail) = 0.418($450) + 0.582(-$250) = $42.60

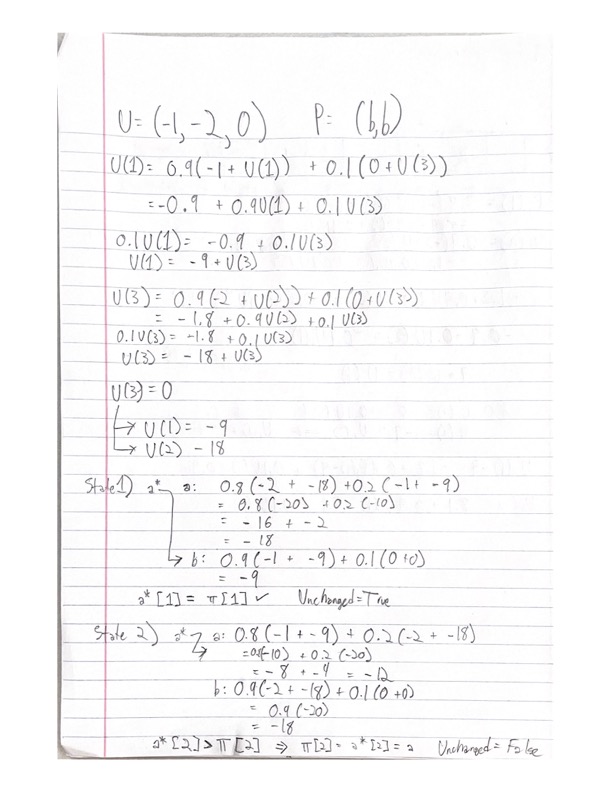
EU(buy new | fail) = -$50

Action = argmaxa = buy used

**4**



**5**



Text, letter

Description automatically generated

**6**

# Author: Caden Kroonenberg

# Value Iteration of 4x3 world shown in Fig. 17.1 of R&N

import numpy as np

R = -1

*def* actionRewardFunction(*initialPosition*, *action*):

if *initialPosition* in termination\_states:

return *initialPosition*, 0

# reward = R

finalPosition = np.array(*initialPosition*) + np.array(*action*)

if -1 in finalPosition or finalPosition[0] == gridSize[0] or finalPosition[1] == gridSize[1] or (finalPosition == [1,1]).all():

finalPosition = *initialPosition*

return finalPosition, R

*def* otherActions(*action*):

if *action* == 0 or *action* == 2: # If Right or Left

return 1, 3 # Return Up, Down

else: # If Up or Down

return 0, 2 # Return Right, Left

gamma = 1

gridSize = [4,3]

termination\_states = [[3,2], [3,1]]

states = [[i,j] for i in range(gridSize[0]) for j in range(gridSize[1])]

states.remove([1,1]) # blocked state

actions = {0: [1,0], 1: [0,1], 2: [-1,0], 3: [0,-1]} # [Right, Up, Left, Down]

values = np.zeros((4,3))

values[3][2] = 1

values[3][1] = -1

# Print epoch 0 values (initial values)

print("Iteration 0 (Initial Values)")

print(values)

print("")

# Value Iteration

for i in range(100):

copyValues = np.copy(values)

for s in states:

# Rewards for each action

q\_values = {a: 0 for a in actions}

for a in actions:

# intended direction

s\_, reward = actionRewardFunction(s, actions[a])

q\_values[a] += 0.8\*(reward + gamma\*values[s\_[0], s\_[1]])

for a\_ in otherActions(a): # unintended direction

s\_, reward = actionRewardFunction(s, actions[a\_])

q\_values[a] += 0.1\*(reward + gamma\*values[s\_[0], s\_[1]])

copyValues[s[0], s[1]] = np.max(list(q\_values.values())) # util'[state] = max a in A(s)

# Check for convergence

comparison = values == copyValues

# Update value array

values = copyValues

# Print values if convergence was reached

if comparison.all():

print("Iteration {} (Final Iteration)".format(i+1))

print(values)

print()

# Stop iterating after convergence

break

# Print ever 10 iterations

if (i+1) % 10 == 0:

print("Iteration {}".format(i+1))

print(values)

print()

Text

Description automatically generated**A picture containing text, white

Description automatically generated**I found the policy and utility values as seen on the left. The output of my program is seen below. This policy tends towards the +1 terminal state and typically avoids the -1 terminal state (except in state 4,1).

3

2

1

3

4

2

1

2

1

Policy for R = -1, γ = 1

Values for R = -1, γ = 1