Caden Kroonenberg
Problem Set 9
EECS 649
4-9-22

1

$$\text{EMV} = \frac{1}{50}(\$10) + \frac{1}{2{,}000{,}000}(\$1{,}000{,}000) + \frac{1{,}959{,}999}{2{,}000{,}000}(\$0) = \$0.70$$

$$\text{EU(Accept)} = \frac{1}{50}U(S_{k+10}) + \frac{1}{2{,}000{,}000}U(S_{k+1{,}000{,}000}) + \frac{1{,}959{,}999}{2{,}000{,}000}U(S_k)$$
$$= \frac{1}{5}U(S_{k+1}) + \frac{1}{2{,}000{,}000}U(S_{k+1{,}000{,}000})$$

$$\text{EU(Reject)} = U(S_{k+1})$$

$$\text{Accept if } \frac{1}{5}U(S_{k+1}) + \frac{1}{2{,}000{,}000}U(S_{k+1{,}000{,}000}) > U(S_{k+1})$$
$$\text{Accept if } \frac{1}{2{,}000{,}000}U(S_{k+1{,}000{,}000}) > \frac{4}{5}U(S_{k+1})$$
$$\text{Accept if } U(S_{k+1{,}000{,}000}) > 1{,}600{,}000 * U(S_{k+1})$$

2

a. $\text{EMV} = \frac{1}{2}(\$2) + \frac{1}{4}(\$2^2) + \frac{1}{8}(\$2^3) + \ldots + \frac{1}{n}(\$2^n)$
$= \sum_{i=1}^{n=\infty} \frac{2^i}{2^i} = \sum_{i=1}^{n=\infty} 1 = \infty$. 2^n grows at a faster rate than n.

b. \$5

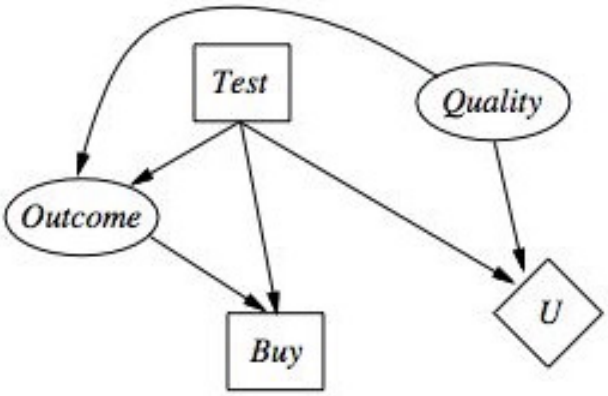
c. $\text{EU(Accept)} = \frac{1}{2}U(S_{c-c+2}) + \frac{1}{4}U(S_{c-c+4}) + \ldots + \frac{1}{N}U(S_{c-c+N})$
$= \frac{1}{2}(a\log_2(2) + b) + \frac{1}{4}(a\log_2(4) + b) + \ldots \frac{1}{N}(a\log_2(N) + b)$
$= \sum_{i=1}^{N=\infty} \frac{a*i}{2^i} + \sum_{i=1}^{N=\infty} \frac{b}{2^i}$
$= \sum_{i=1}^{N=\infty} \frac{a*i}{2^i} + \sum_{i=1}^{N=\infty} \frac{b}{2^i}$
$= 2a + b$

$\text{EU(Reject)} = U(S_c) = a\log_2(c) + b$

d. $2a + b > a\log_2(c) + b$
$= 2a > a\log_2(c)$
$= 2 > \log_2(c)$
$= 2^2 > 2^{\log 2(c)}$
$= 4 > c$
*assuming $a > 0$
You should play the game if your current wealth is less than \$4.

3

a.



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

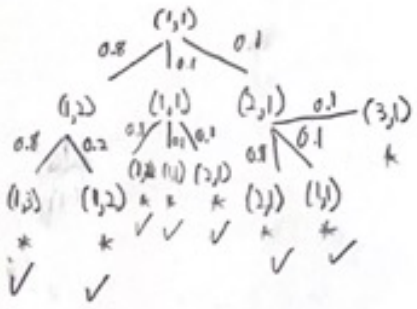
d.

EU(buy used | pass) = 0.842($450) + 0.158(-$250) = $339.40
EU(buy new | pass) = -$50
Action = argmax_a = buy used

EU(buy used | fail) = 0.418($450) + 0.582(-$250) = $42.60
EU(buy new | fail) = -$50
Action = argmax_a = buy used

4

	1	2	3	4
3	0.64	0	0	+1
2	0.16+0.08 0.24		0	−1
1	0.01+0.01 0.02	0.01+0.08 0.09	0.01	0



	1	2	3	4
3	0.64(0.1) 0.24(0.1) 0.088	(0.64)(0.8) 0.512	0	+1
2	0.64(0.1) 0.24(0.8) 0.02(0.1) 0.258		0.01(0.1) 0.001	−1
1	0.24(0.1) 0.02(0.1) 0.026	0.02(0.8) 0.09(0.2) 0.034	0.09(0.8) 0.01(0.1) 0.073	0.01(0.8) 0.008

5

$U = (-1, -2, 0)$ $P = (b, b)$

$U(1) = 0.9(-1 + U(1)) + 0.1(0 + U(3))$

$= -0.9 + 0.9U(1) + 0.1U(3)$

$0.1U(1) = -0.9 + 0.1U(3)$

$U(1) = -9 + U(3)$

$U(3) = 0.9(-2 + U(2)) + 0.1(0 + U(3))$

$= -1.8 + 0.9U(2) + 0.1U(3)$

$0.1U(3) = -1.8 + 0.1U(3)$

$U(3) = -18 + U(3)$

$U(3) = 0$

→ $U(1) = -9$

→ $U(2) = -18$

State 1) a^* → a: $0.8(-2 + -18) + 0.2(-1 + -9)$

$= 0.8(-20) + 0.2(-10)$

$= -16 + -2$

$= -18$

→ b: $0.9(-1 + -9) + 0.1(0 + 0)$

$= -9$

$a^*[1] = \pi[1]$ ✓ Unchanged = True

State 2) a^* → a: $0.8(-1 + -9) + 0.2(-2 + -18)$

$= 0.8(-10) + 0.2(-20)$

$= -8 + -4 = -12$

b: $0.9(-2 + -18) + 0.1(0 + 0)$

$= 0.9(-20)$

$= -18$

$a^*[2] > \pi[2] \Rightarrow \pi[2] = a^*[2] = a$ Unchanged = False

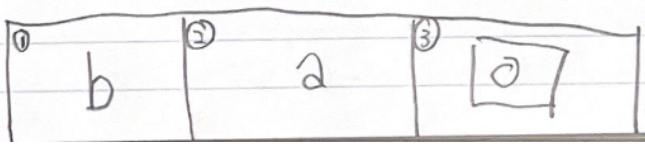
$$U(1) = 0.9(-1 + U(1)) + 0.1(0 + U(3))$$
$$= -0.9 + 0.9U(1) + 0.1U(3)$$
$$0.1U(1) = -0.9 + 0.1U(3)$$
$$U(1) = -9 + U(3)$$

$$U(2) = 0.8(-1 + U(1)) + 0.2(-2 + U(2))$$
$$= -0.8 + 0.8U(1) + -0.4 + 0.2U(2)$$
$$0.8U(2) = -0.8 + 0.8U(1) + -0.4$$
$$U(2) = -1 + U(1) - 1/2$$
$$= U(1) - 1.5$$

$U(3) = 0$
$U(1) = -9$
$U(2) = -10.5$

State 1) a^* → a: $0.8(-2 + -10.5) + 0.2(-1 + -9)$
$= 0.8(-12.5) + 0.2(-10)$
$= -10 - 2 = -12$
→ b. $0.9(-1 + -9) + 0.1(0 + 0)$
$= -9$
$a^*[1] = \pi[1]$ Unchanged = true

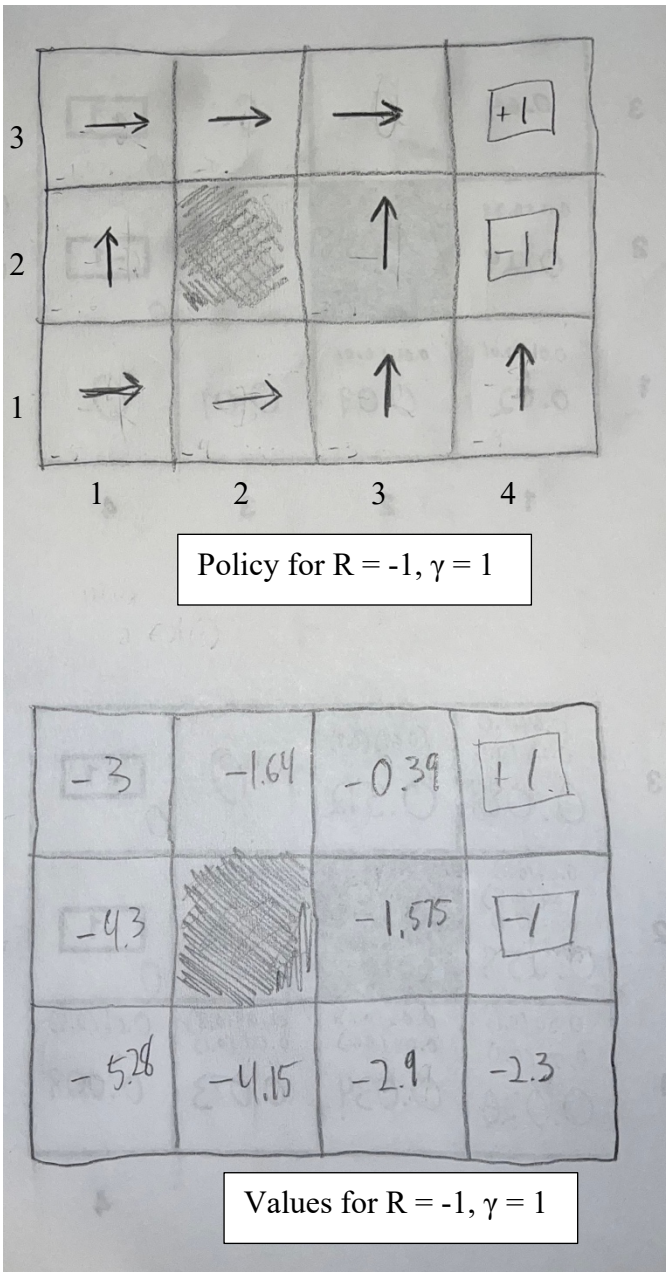
State 2) a^* → a: $0.8(-1 + -9) + 0.2(-2 + -10.5)$
$= 0.8(-10) + 0.2(-12.5)$
$= -8 + -2.5$
$= -10.5$
b: $0.9(-2 + -10.5) + 0.1(0 + 0)$
$= 0.9(-12.5)$
$= -11.25$
$a^*[2] = \pi[2]$ Unchanged = true DONE!



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("")
```

```python
# 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()
```



Policy for R = -1, γ = 1

Values for R = -1, γ = 1

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).

```
→  ps9 git:(main) ✗ python3 value_iteration.py
Iteration 0 (Initial Values)
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0. -1.  1.]]

Iteration 10
[[-5.27173816 -4.29436539 -3.05056054]
 [-4.15360645  0.         -1.64710553]
 [-2.90702187 -1.57524235 -0.39722488]
 [-2.32273423 -1.          1.        ]]

Iteration 20
[[-5.28566078 -4.30350728 -3.05350937]
 [-4.15843228  0.         -1.64726026]
 [-2.90843296 -1.57534246 -0.39726027]
 [-2.32315917 -1.          1.        ]]

Iteration 30
[[-5.285664   -4.30351027 -3.05351027]
 [-4.15843322  0.         -1.64726027]
 [-2.90843322 -1.57534247 -0.39726027]
 [-2.32315925 -1.          1.        ]]

Iteration 40
[[-5.285664   -4.30351027 -3.05351027]
 [-4.15843322  0.         -1.64726027]
 [-2.90843322 -1.57534247 -0.39726027]
 [-2.32315925 -1.          1.        ]]

Iteration 50 (Final Iteration)
[[-5.285664   -4.30351027 -3.05351027]
 [-4.15843322  0.         -1.64726027]
 [-2.90843322 -1.57534247 -0.39726027]
 [-2.32315925 -1.          1.        ]]
```