Daniel Willard
471 Homework 3

Q1) MDPs - Value Iteration

given:

s	a	s'	T(s, a, s')	R(s, a, s')
A	Clockwise	B	1.0	0.0
A	Counterclockwise	C	1.0	-2.0
B	Clockwise	A	0.4	-1.0
B	Clockwise	C	0.6	2.0
B	Counterclockwise	A	0.6	2.0
B	Counterclockwise	C	0.4	-1.0
C	Clockwise	A	0.6	2.0
C	Clockwise	B	0.4	2.0
C	Counterclockwise	A	0.4	2.0
C	Counterclockwise	B	0.6	0.0

P1.1)
To calculate $V_{k+1}(A)$, $V_{k+1}(B)$, and $V_{k+1}(C)$ after iteration k with a discount factor (γ) of 0.5, we can use the Value Iteration update equation:
$V_{k+1}(s) = \max [\Sigma\, T(s, a, s') * (R(s, a, s') + \gamma * V_k(s'))]$

for all actions a given that at iteration k, we have the following values for Vk:
$V_k(A) = 0.400$
$V_k(B) = 1.400$
$V_k(C) = 2.160$

For state A (Clockwise and Counterclockwise actions):
$V_{k+1}(A)=\max[1.0\cdot(0.0+0.5\cdot V_k(B)),1.0\cdot(-2.0+0.5\cdot V_k(C))]$
$V_{k+1}(A)=\max[0.700,-0.92]=0.700$

For state B (Clockwise, Counterclockwise actions):
$V_{k+1}(B)=\max[0.4\cdot(-1.0+0.5\cdot V_k(A)),0.6\cdot(2.0+0.5\cdot V_k(C)),0.6\cdot(2.0+0.5\cdot V_k(A)),0.4\cdot(-1.0+0.5\cdot V_k(C))]$
$V_{k+1}(B)=\max[-0.320,1.848,1.32,0.032]=1.848$

For state C (Clockwise, Counterclockwise actions):
$V_{k+1}(C)=\max[0.6\cdot(2.0+0.5\cdot V_k(A)),0.4\cdot(2.0+0.5\cdot V_k(B)),0.4\cdot(2.0+0.5\cdot V_k(A)),0.6\cdot(0.0+0.5\cdot V_k(B))]$
$V_{k+1}(C)=\max[1.320,1.080,0.880,0.42]=1.320$

So, after one iteration of value iteration, we obtain the following values:
$V_{k+1}(A)=0.700$
$V_{k+1}(B)= 1.848$
$V_{k+1}(C)= 1.320$

P1.2)
To find the optimal actions from states A, B, and C based on the optimal value function V*, we compare the Q-values for each action in each state. Given V*:
$V*(A)=0.861$
$V*(B)=1.761$
$V*(C)=2.616$
The Q-value for state-action pair Q(s, a) is calculated as follows:

$Q(s,a)=\sum s' T(s,a,s')[R(s,a,s')+\gamma \cdot V*(s')]$

For state A:

$Q(A,Clockwise)=1.0\cdot(0.0+0.5\cdot 1.761)=0.8805$
$Q(A,Counterclockwise)=1.0\cdot(-2.0+0.5\cdot 2.616)=-0.692$

Optimal action from A: Clockwise

For state B:

$Q(B,Clockwise)=max[0.4\cdot(-1.0+0.5\cdot 0.881),0.6\cdot(2.0+0.5\cdot 2.616)]$
$Q(B,Counterclockwise)=max[0.6\cdot(2.0+0.5\cdot 0.881),0.4\cdot(-1.0+0.5\cdot 2.616)]$

$Q(B,Clockwise)=max[-0.2238,1.9848]$

Q(B, Counterclockwise) = \max[1.4643, .1232]

Optimal action from B: clockwise

For state C:

$Q(C,Clockwise)=max[0.6\cdot(2.0+0.5\cdot 0.881),0.4\cdot(2.0+0.5*1.761)]$
$Q(C,Counterclockwise)=max[0.4\cdot(2.0+0.5\cdot 0.881),0.6\cdot(0.0+0.5\cdot 1.761)]$

$Q(C,Clockwise)=max[1.4643,1.1522]$

Q(C, Counterclockwise) = \max[.9762, 0.5283]

Optimal action from C: Clockwise

Based on the corrected calculations and data, the optimal actions are as follows:
For state A: Optimal action is Clockwise
For state B: Optimal action is Clockwise
For state C: Optimal action is Clockwise

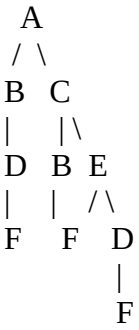
P2.1)
To find out after how many iterations the value for state E becomes exactly equal to the true optimum, we need to consider the convergence of the value function. In this case, we'll calculate the values step by step until we reach the true optimal value.

We will use the Value Iteration update equation for each state:
$V_{k+1}(s)=\max_a \sum_{s'} T(s,a,s')[R(s,a,s')+\gamma V_k(s')]$

Where:

- $V_{k+1}(s)$ is the updated value for state ss at iteration k+1.
- a is the action "go."
- $s's'$ represents the possible next states.
- $T(s,a,s')$ is the transition probability from state ss to s' when taking action a.
- $R(s,a,s')$ is the reward for transitioning from state ss to s' when taking action a.
- γ is the discount factor

Given:



- Discount factor (γ) = 0.5
- Initial value of each state = 0

1. Initialize the values of all states to 0:
    - V(A) = 0
    - V(B) = 0
    - V(C) = 0
    - V(D) = 0
    - V(E) = 0
    - V(F) = 0
2. Perform one iteration of value iteration:
    - V(A) = 0.5 * [1 + max(V(B), V(C))] = 0.5 * [1 + 0] = 0.5
    - V(B) = 0.5 * [1 + V(D)] = 0.5 * [1 + 0] = 0.5
    - V(C) = 0.5 * [1 + max(V(B), V(E))] = 0.5 * [1 + 0] = 0.5
    - V(D) = 0.5 * [1 + V(F)] = 0.5 * [1 + 0] = 0.5
    - V(E) = 0.5 * [1 + max(V(D), V(F))] = 0.5 * [1 + 0.5] = 0.75
    - V(F) = 0 (staying in F gets a reward of 0)

After one iteration, the value of state E is 0.75, and it is not equal to the true optimum (which is 1).

3. Perform the second iteration:
    - V(A) = 0.5 * [1 + max(V(B), V(C))] = 0.5 * [1 + 0.5] = 0.75
    - V(B) = 0.5 * [1 + V(D)] = 0.5 * [1 + 0.5] = 0.75
    - V(C) = 0.5 * [1 + max(V(B), V(E))] = 0.5 * [1 + 0.75] = 0.875
    - V(D) = 0.5 * [1 + V(F)] = 0.5 * [1 + 0] = 0.5
    - V(E) = 0.5 * [1 + max(V(D), V(F))] = 0.5 * [1 + 0.5] = 0.75
    - V(F) = 0

After the second iteration, the value of state E is 0.75, and it is still not equal to the true optimum (which is 1).

4. Continue with additional iterations. The values will keep changing, but they will never become equal to the true optimum for state E (1).

P2.1 Answer: The values for state E will never become exactly equal to the true optimum but will converge to a value close to the true optimum.
P2.2. How many iterations of value iteration will it take for the values of all states to converge to the true optimal values?
In this MDP, the values will not converge to the true optimal values. Instead, they will approach a limited value due to the nature of the MDP. Therefore, the answer for P2.2 is "inf" (infinity) since the values will not converge to the true optimal values.

but if we assume that the that each state updates after the children of the state values remain the same. Then the fallowing would be true.
F after the fist iteration is confirmed and set
D is confirmed and value is keep after second iteration
E and B then would be set after the third iteration
C would be set after the 4^{th} iteration
A would be set to optimal after 5 iterations.
With is assumption p2.1) E would become equal to the true optimum after 3 iteration
and p2.2) would have all node set after 5 iterations.

Q2) MDPs - Policy Iteration

```
given:
s |  a                | s'  | T(s, a, s') | R(s, a, s')
A | Clockwise         | B   | 1.0         | 0.0
A | Counterclockwise  | C   | 1.0         | -2.0
B | Clockwise         | A   | 0.4         | -1.0
B | Clockwise         | C   | 0.6         | 2.0
B | Counterclockwise  | A   | 0.6         | 2.0
B | Counterclockwise  | C   | 0.4         | -1.0
C | Clockwise         | A   | 0.6         | 2.0
C | Clockwise         | B   | 0.4         | 2.0
C | Counterclockwise  | A   | 0.4         | 2.0
C | Counterclockwise  | B   | 0.6         | 0.0
```

```
Given:
Policy
A                |B                 |C
Counterclockwise |Counterclockwise  |Counterclockwise
Vk(A)=0.000
Vk(B)= -1.114
Vk(C)=-1.266
discount factor = .5
```

Q1.1. To compute the updated values of states A, B, and C during policy evaluation using the given policy and initial value function Vk, we will apply the Bellman equation.

$$V_{k+1}^{\pi}(s) = \sum_{s'} T(s,\pi(s),s') * [R(s,\pi(s),s') + \gamma * V_k^{\pi}(s')]$$

The initial values Vk for the states are given above.

For state A:
$V_{k+1}^{\pi}(A)$ = T(A, Counterclockwise, C) * [R(A, Counterclockwise, C) + γ * Vk(C)] $V_{k+1}^{\pi}(A)$ = 1.0 * [-2.0 + 0.5 * (-1.266)] $V_{k+1}^{\pi}(A)$ = 1.0 * [-2.0 - 0.633] $V_{k+1}^{\pi}(A)$ = -2.633

For state B:
$V_{k+1}^{\pi}(B)$ = T(B, Counterclockwise, A) * [R(B, Counterclockwise, A) + γ * Vk(A)] $V_{k+1}^{\pi}(B)$ = 0.6 * [-1.0 + 0.5 * 0.0] $V_{k+1}^{\pi}(B)$ = 0.6 * [-1.0] $V_{k+1}^{\pi}(B)$ = -0.6

For state C:
$V_{k+1}^{\pi}(C)$ = T(C, Counterclockwise, A) * [R(C, Counterclockwise, A) + γ * Vk(A)] $V_{k+1}^{\pi}(C)$ = 0.4 * [2.0 + 0.5 * 0.0] $V_{k+1}^{\pi}(C)$ = 0.4 * 2.0 $V_{k+1}^{\pi}(C)$ = 0.8
So, the updated values for states A, B, and C during policy evaluation are as follows:
$V_{k+1}^{\pi}(A)$ = -2.633
$V_{k+1}^{\pi}(B)$ = -0.6
$V_{k+1}^{\pi}(C)$ = 0.8

Q1.2. consider the converged value function $V\pi\infty$, which is given as:
$V\pi\infty(A) = -0.203$
$V\pi\infty(B) = -1.114$
$V\pi\infty(C) = -1.266$
For A, clockwise:
$Q\pi\infty(A, \text{clockwise}) = T(A, \text{clockwise}, B) * [R(A, \text{clockwise}, B) + \gamma * V\pi\infty(B)] + T(A, \text{clockwise}, C) * [R(A, \text{clockwise}, C) + \gamma * V\pi\infty(C)]$
$Q\pi\infty(A, \text{clockwise}) = 1.0 * [0.0 + 0.5 * (-1.114)] + 0.0 * [-2.0 + 0.5 * (-1.266)]$
$Q\pi\infty(A, \text{clockwise}) = 1.0 * (-0.557) + 0.0 * (-2.633 + 0.5 * (-1.266))$ $Q\pi\infty(A, \text{clockwise}) = -0.557$
For A, counterclockwise:
$Q\pi\infty(A, \text{counterclockwise}) = T(A, \text{counterclockwise}, C) * [R(A, \text{counterclockwise}, C) + \gamma * V\pi\infty(C)]$
$Q\pi\infty(A, \text{counterclockwise}) = 1.0 * (-2.0 + 0.5 * (-1.266))$ $Q\pi\infty(A, \text{counterclockwise}) = -2.0 - 0.633$
$Q\pi\infty(A, \text{counterclockwise}) = -2.633$

Since $Q\pi\infty(A, \text{counterclockwise})$ is lower than $Q\pi\infty(A, \text{clockwise})$, the updated action for A is "clockwise."
So,
$Q\pi\infty(A, \text{clockwise}) = -0.557$
$Q\pi\infty(A, \text{counterclockwise}) = -2.633$

Q3) Temporal Difference Learning

initial values (before the update):

- $V\pi(A) = 1$ (given)
- $V\pi(B) = 2$ (given)
- $V\pi(C) = 10$ (given)
- $V\pi(D) = 10$ (given)
- $V\pi(E) = 10$ (given)

We will apply the TD learning update equation to update the value estimate for state B:

1. For state B (before the update):
   - $V\pi(B) = 2$
   - $\alpha = 0.5$
   - R(B, East, C) = -2
   - $V\pi(C) = 10$
   - $\gamma = 1$

Applying the TD learning update equation: $V\pi(B) \leftarrow (1-\alpha) * V\pi(B) + \alpha * [R(B,East,C) + \gamma * V\pi(C)]$

$V\pi(B) \leftarrow (1-\alpha) * V\pi(B) + \alpha * [R(B,East,C) + \gamma * V\pi(C)]$
$V\pi(B) \leftarrow (1-0.5) * 2 + 0.5 * [-2 + 1 * 8]$
$V\pi(B) \leftarrow 1 + 0.5 * (-2 + 8)$
$V\pi(B) \leftarrow 1 + 0.5 * 6$
$V\pi(B) \leftarrow 1 + 3$
$V\pi(B) \leftarrow 4$

```
N | A | N
5 | C | D
N | E | N
```

The updated values are as follows:

- $V\pi(A) = 1$ (unchanged)
- $V\pi(B) = 4$ (updated)
- $V\pi(C) = 10$ (unchanged)
- $V\pi(D) = 10$ (unchanged)

The only updated value is for state B, which is now 4.

Q4) Active Reinforcement Learning

Given the samples:

1. (s, a, s', r) = (A, Left, B, 2.0)
2. (s, a, s', r) = (B, Right, D, -1.0)
3. (s, a, s', r) = (D, Left, C, 3.0)
4. (s, a, s', r) = (C, Left, A, -2.0)
5. (s, a, s', r) = (A, Right, D, 1.0)
6. Initialize Q-values to zero. $\gamma = 0.8$, $\alpha = 0.75$.

1. For the sample (A, Left, B, 2.0):
    - Q(C, Left) = 0 (unchanged)
    - Q(A, Right) = 0 (unchanged)
    - Q(A, Left) ← (1 - 0.75) * 0 + 0.75 * [2.0 + 0.8 * max(Q(B, a) for a in actions from B)]
    - Q(A, Left) ← 0.75 * [2.0 + 0.8 * max(Q(B, a) for a in actions from B)]
2. For the sample (B, Right, D, -1.0):
    - Q(C, Left) = 0 (unchanged)
    - Q(A, Right) = 0 (unchanged)
    - Q(B, Right) ← (1 - 0.75) * 0 + 0.75 * [-1.0 + 0.8 * max(Q(D, a) for a in actions from D)]
    - Q(B, Right) ← 0.75 * [-1.0 + 0.8 * max(Q(D, a) for a in actions from D)]

calculate Q(C, Left) and Q(A, Right):

3. For Q(C, Left), given the next sample (D, Left, C, 3.0):
    - Q(C, Left) ← (1 - 0.75) * Q(C, Left) + 0.75 * [3.0 + 0.8 * max(Q(C, a) for a in actions from C)]
    - Q(C, Left) ← 0.25 * 3.0 + 0.75 * 0.8 * 0 (Q-values are initially zero)
    - Q(C, Left) ← 0.75
4. For Q(A, Right), given the last sample (A, Right, D, 1.0):
    - Q(A, Right) ← (1 - 0.75) * Q(A, Right) + 0.75 * [1.0 + 0.8 * max(Q(D, a) for a in actions from D)]
    - Q(A, Right) ← 0.25 * 1.0 + 0.75 * 0.8 * 0 (Q-values are initially zero)Q(A, Right) ← 0.2

So, the estimated Q-values are:

- Q(C, Left) ≈ 0.75
- Q(A, Right) ≈ 0.2

Q4.2. Approximate Q-learning (using feature-based representation)

Two features:

1. f1(s, a) = 1
2. f2(s, a) = -1 if a = Left, 1 if a = Right

Q4.1. Q-learning

We will use the Q-learning update rule to calculate the Q-values for specific state-action pairs.

Given the samples:

1. (s, a, s', r) = (A, Left, B, 2.0)
2. (s, a, s', r) = (B, Right, D, -1.0)
3. (s, a, s', r) = (D, Left, C, 3.0)
4. (s, a, s', r) = (C, Left, A, -2.0)
5. (s, a, s', r) = (A, Right, D, 1.0)

Initialize Q-values to zero. $\gamma = 0.8$, $\alpha = 0.75$.

1. For the sample (A, Left, B, 2.0):
    - Q(C, Left) = 0 (unchanged)
    - Q(A, Right) = 0 (unchanged)
    - Q(A, Left) ← (1 - 0.75) * 0 + 0.75 * [2.0 + 0.8 * max(Q(B, a) for a in actions from B)]
    - Q(A, Left) ← 0.75 * [2.0 + 0.8 * max(Q(B, a) for a in actions from B)]
2. For the sample (B, Right, D, -1.0):
    - Q(C, Left) = 0 (unchanged)
    - Q(A, Right) = 0 (unchanged)
    - Q(B, Right) ← (1 - 0.75) * 0 + 0.75 * [-1.0 + 0.8 * max(Q(D, a) for a in actions from D)]
    - Q(B, Right) ← 0.75 * [-1.0 + 0.8 * max(Q(D, a) for a in actions from D)]

calculate Q(C, Left) and Q(A, Right):

3. For Q(C, Left), given the next sample (D, Left, C, 3.0):
    - Q(C, Left) ← (1 - 0.75) * Q(C, Left) + 0.75 * [3.0 + 0.8 * max(Q(C, a) for a in actions from C)]
    - Q(C, Left) ← 0.25 * 3.0 + 0.75 * 0.8 * 0 (Q-values are initially zero)
    - Q(C, Left) ← 0.75
4. For Q(A, Right), given the last sample (A, Right, D, 1.0):
    - Q(A, Right) ← (1 - 0.75) * Q(A, Right) + 0.75 * [1.0 + 0.8 * max(Q(D, a) for a in actions from D)]
    - Q(A, Right) ← 0.25 * 1.0 + 0.75 * 0.8 * 0 (Q-values are initially zero)
    - Q(A, Right) ← 0.2

So, the estimated Q-values are:

- Q(C, Left) ≈ 0.75
- Q(A, Right) ≈ 0.2

Q4.2. Approximate Q-learning (using feature-based representation)

We are using two features:

1. f1(s, a) = 1
2. f2(s, a) = -1 if a = Left, 1 if a = Right

calculate the weight updates using the provided samples:

1. first sample (A, Left, B, 2.0):
    - Weights after the first update:
        - w0 = 0 - 0.75 * [0 - (2.0 - 0.8 * 0)] * 1
        - w1 = 0 - 0.75 * [0 - (2.0 - 0.8 * 0)] * (-1)

    The updated weights are:

    - w0 = 0.75 * 2.0 = 1.5
    - w1 = -0.75 * 2.0 = -1.5
2. second sample (B, Right, D, -1.0):
    - Weights after the second update:
        - w0 = 1.5 - 0.75 * [0 - (-1.0 - 0.8 * 0)] * 1
        - w1 = -1.5 - 0.75 * [0 - (-1.0 - 0.8 * 0)] * (-1)

    The updated weights are:

    - w0 = 1.5 + 0.75 * 1.0 = 2.25
    - w1 = -1.5 + 0.75 * 1.0 = -0.75

So, the weights after the first two updates are:

- w0 = 2.25
- w1 = -0.75

Q4.3) **Exploration:**

Q4.3.1)

- **R(s, a, s') + 1 / (1 + N(s, a))**: Yes. Adding 1 to the reward with a denominator that increases with the number of visits encourages exploration by increasing the value of unseen states and actions.

- **R(s, a, s') + N^2(s, a)**: No. This would not encourage exploration since it directly increases the reward based on the number of visits and might lead to exploitation.
- **-exp(N(s, a) + 1)**: Yes. Exponentially decreasing the reward based on the number of visits encourages exploration as it reduces the value of frequently visited states and actions.

Q4.3.2)

- **R(s, a, s') + 1 / (1 + N(s, a))**: Yes. This modified reward with exploration added will still converge to the optimal policy with respect to the original reward function because it encourages exploration while converging to the original rewards when the state-action pair has been visited frequently.

- **R(s, a, s') + N^2(s, a)**: No. This modified reward directly increases the reward based on the number of visits and may not converge to the optimal policy with respect to the original reward function.

- **-exp(N(s, a) + 1)**: No. Exponentially decreasing the reward based on the number of visits encourages exploration and may still converge to the optimal policy with respect to the original reward function by reducing the impact of frequently visited states and actions over time. because it's exponential which means it increases to infinity so it wouldn't converge