Reinforcement Learning - Assignment 2

Q1

Ex 3.4

s	a	s'	r	P(s',r|s,a)
high	search	high	r_{search}	α
high	search	low	r_{search}	$1-\alpha$
~~to~~ low	search	high	-3	$1-\beta$
low	search	low	r_{search}	β
high	wait	high	r_{wait}	1
~~high~~ low	wait	low	r_{wait}	1
low	recharge	high	0	1

The table contains rows for all cases where $p(s', r \mid s, a) > 0$

This table is based on Eg 3.3 in the book. We have assumed that ~~the~~ :

$$P[\cancel{high}\ S_{t+1} = high \mid S_t = high, A_t = search] = \alpha$$
$$P[S_{t+1} = low \mid S_t = low, A_t = search] = \beta$$

Average reward for search $= r_{search}$
Average reward for wait $= r_{wait}$
~~Cost o~~ Reward on completely running out of battery $= -3$
Reward for recharge $= 0$

Q2

For this question refer to the file **GridWorld.ipynb**. The file contains solutions to this problem in 2 approaches :

1. By solving simultaneous linear equations
2. Using policy evaluation

Two approaches have been used as an experiment to compare solutions obtained using iterative means as compared to the exact solution.
Note: The values achieved using policy evaluation have been rounded off to 1 decimal place

Optimal State Value Function :

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Q3

Ex 3.15

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

If we add a constant 'c' to each reward then new return G_t' is,

$$G_t' = R_{t+1} + c + \gamma (R_{t+2} + c) + \dots$$

$$= c + \gamma c + \gamma^2 c + \dots + R_{t+1} + \gamma R_{t+2} + \dots$$

$$\Rightarrow G_t' = \frac{c}{1-\gamma} + G_t$$

Now,

$$V_\pi(s) = E\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \mid S_t = s\right]$$

$$V_\pi'(s) = E\left[\sum_{t=1}^{\infty} \gamma^{t-1} (R_t + c) \mid S_t = s\right]$$

$$= E\left[\sum_{t=1}^{\infty} \gamma^{t-1} (R_t) \mid S_t = s\right] + E\left[\sum_{t=1}^{\infty} \gamma^{t-1} c \mid S_t = s\right]$$

$$\Rightarrow V_\pi'(s) = V_\pi(s) + \frac{c}{1-\gamma}$$

$$\Rightarrow V_c = c/(1-\gamma) \equiv \text{constant}$$

Thus the sign of the reward does not matter, only the interval between them

Ex 3.16

Adding a constant 'c' to all rewards would can effect unlike in continuous tasks.

$$v'_\pi(s) = E\left[R_{t+1}+c+\gamma(R_{t+2}+c)+\dots+\gamma^{K-t-1}(R_K+c)\right]$$

$$= E\left[R_{t+1}+\gamma R_{t+2}+\dots+\gamma^{K-t-1}R_K\right]$$

$$+$$

$$E\left[c+\gamma c+\dots+\gamma^{K-t-1}c\right]$$

$$v'_\pi(s) = v_\pi(s) + c\,\frac{\gamma^{K-t}-1}{\gamma-1}$$

Here 'K' is a random number

thus $c\left(\frac{\gamma^{K-t}-1}{\gamma-1}\right)$ is not a constant

as episodes can end after

random finite time.

Q4

For this question refer to the file **GridWorld.ipynb.**

Optimal State Value Function

```
[[22.   24.4  22.   19.4  17.5]
 [19.8  22.   19.8  17.8  16. ]
 [17.8  19.8  17.8  16.   14.4]
 [16.   17.8  16.   14.4  13. ]
 [14.4  16.   14.4  13.   11.7]]
```

Optimal Policy

```
[['  →    ' '→ ↓ ← ↑' '  ←   ' '→ ↓ ← ↑' '  ←   ']
 [' → ↑  ' '  ↑   ' ' ← ↑ ' '  ←   ' '  ←   ']
 [' → ↑  ' '  ↑   ' ' ← ↑ ' ' ← ↑ ' ' ← ↑ ']
 [' → ↑  ' '  ↑   ' ' ← ↑ ' ' ← ↑ ' ' ← ↑ ']
 [' → ↑  ' '  ↑   ' ' ← ↑ ' ' ← ↑ ' ' ← ↑ ']]
```

Q5

Q5

$\cancel{v_*(s) = \max_a q_*(s,a)}$

$$v_*(s) = \max_{a\in A(s)} q_*(s,a)$$

Q6

For this question refer to the file **DP_GridWorld.ipynb**.

For **policy iteration** we can see that in the first two iteration the value function has increased for each state with respect to its value in the previous state. In the next iteration it converges to the optimal policy wherein the the value function remains the same as the previous state. Thus policy iteration is demonstrated to improve value function at each iteration.

In the below diagram we also see how the policy is changing on each iteration.

To correct the bug mentioned in Ex 4.4, we could use two possible fixes. This is observing the fact that multiple optimal policies would arise from multiple valid actions in a state have the same action-value function value --> Q(s,a1) = Q(s,a2).
Thus the two possible fixes are:
1. In case of multiple optimal actions pick either the first or the last action, by maintaining the same action ordering across states
2. We could have a stochastic policy where for a given state the any of the optimal actions is equiprobable to occur while the non optimal actions occur with a zero probability.

```
[[  0. -14. -20. -22.]
 [-14. -18. -20. -20.]
 [-20. -20. -18. -14.]
 [-22. -20. -14.   0.]]

[['      ' '  ←   ' '  ←   ' '  ←   ']
 ['  ↑   ' ' ← ↑  ' '  ←   ' '  ↓   ']
 ['  ↑   ' '  ↑   ' '  ↓   ' '  ↓   ']
 ['  ↑   ' '  →   ' '  →   ' '      ']]

[[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]

[['      ' '  ←   ' '  ←   ' ' ↓ ←  ']
 ['  ↑   ' ' ← ↑  ' '→ ↓ ← ↑' '  ↓   ']
 ['  ↑   ' '→ ↓ ← ↑' ' → ↓  ' '  ↓   ']
 [' → ↑  ' '  →   ' '  →   ' '      ']]

[[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]

[['      ' '  ←   ' '  ←   ' ' ↓ ←  ']
 ['  ↑   ' ' ← ↑  ' '→ ↓ ← ↑' '  ↓   ']
 ['  ↑   ' '→ ↓ ← ↑' ' → ↓  ' '  ↓   ']
 [' → ↑  ' '  →   ' '  →   ' '      ']]

Dispalying Optimal Policy and Value Function

[[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]

[['      ' '  ←   ' '  ←   ' ' ↓ ←  ']
 ['  ↑   ' ' ← ↑  ' '→ ↓ ← ↑' '  ↓   ']
 ['  ↑   ' '→ ↓ ← ↑' ' → ↓  ' '  ↓   ']
 [' → ↑  ' '  →   ' '  →   ' '      ']]
```

Below are the value functions after every iteration of **Value Iteration**. At the end you can see the optimal value function and policy. Notice that it matches the one generated by policy iteration.

```
[[ 0. -1. -1. -1.]
 [-1. -1. -1. -1.]
 [-1. -1. -1. -1.]
 [-1. -1. -1.  0.]]


[[ 0. -1. -2. -2.]
 [-1. -2. -2. -2.]
 [-2. -2. -2. -1.]
 [-2. -2. -1.  0.]]


[[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]


[[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]

 Optimal Policy and Value Function

 [[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]

 [['        ' '   ←    ' '   ←    ' '  ↓ ←   ']
 ['   ↑    ' '  ← ↑   ' '→ ↓ ← ↑ ' '   ↓    ']
 ['   ↑    ' '→ ↓ ← ↑ ' '  → ↓   ' '   ↓    ']
 ['  → ↑   ' '   →    ' '   →    ' '        ']]
```

Q7

Refer to the file **Jacks_Car_Rental.ipynb** for the solution.

Below are the heatmaps for the policies after every iteration until the converge. This heatmap is for the modified Jack's Car Rental decribed in Ex4.7. Prior to running for the Ex 4.7 I have also verified the correctness of the code by running on the original Jack's Car Rental Example. Necessary comments are placed in the code to adapt current code to run the original Jack's Car Rental problem