Assignment 2
11th Sept, 2019

Reinforcement Learning

Yajur Ahuja
2016121

Q1

classmate
Date
Page

Assuming finding & not finding have equal prob of $\frac{1}{2}$ (searching & waiting)

Q1.

Ex 3.4 $p(s', r | s, a) = p(r | s, a, s') \times p(s' | s, a) > 0.$

$S \in \{high, low\}$. $A(s) \in \{\cancel{s, w}\}$. $A(high) \in \{s, w\}$

$s' \in \{high, low\}$. $\cancel{A(s') \in \{}$. $A(low) \in \{s, w, r\}$.

	s	a	s'	r	$p(r\|s,a,s')$	$p(s'\|s,a)$	$p(s',r\|a,s)$
1.	h	s	h	0	1/2	α	$\frac{1}{2}\alpha$
2.	h	s	h	1	1/2	α	$\frac{1}{2}\alpha$
3.	h	s	l	0	1/2	$(1-\alpha)$	$\frac{1}{2}(1-\alpha)$
4.	h	s	l	1	1/2	$(1-\alpha)$	$\frac{1}{2}(1-\alpha)$
5.	h	w	h	0	1/2	1.	~~1~~ 1/2
6.	h	w	h	1	~~1~~ 1/2	1.	1/2
7.	l	s	h	-3	1	$1-\beta$	$1-\beta$
8.	l	s	l	0	1/2	β	$1/2\ \beta$
9.	l	s	l	1	1/2	β	$1/2\ \beta$
10.	l	w	l	0	1/2	1	1/2
11.	l	w	l	1	1/2	1.	1/2
12	l	r	h	0	1	1.	1.

rest have $p(s', r | s, a) = 0$

Either ~~the~~ it is impossible to reach the states from a ~~cert~~ certain state and action taken i.e. $p(s' | s, a) = 0$ or if they did receive the state s', the reward is not ~~what~~ what they wanted i.e. $p(r | s, a, s') = 0$.

Scanned by CamScanner

Q2

Steps:

1) Created a probability Matrix : for all s, s' -> p_ss' = p(s'|s) (25 X 25)
2) Created an Expected reward Matrix: R (25 X 1)
3) gamma : discount factor

We know, that the Bellman equation is a set of linear equations

V (25 X 1) = R (25 X 1) + P (25 X 25) * V (25 X 1)

V = R + gamma *P*V

V - gamma*P *V = R

(I - gamma*P)V = R

V = (I - gamma*P)$^{-1}$ * R

We do the same and obtain the Values in the code

Output Values:

```
[[-16. ,   0.6,  -3.3,   5.8, -13.2],
 [-15.2,  -6.9,  -5.5,  -5.3, -13.7],
 [-17.1, -10.8,  -8.9, -10.2, -16.4],
 [-20.8, -14.9, -13.1, -14.7, -20.5],
 [-27.3, -21.6, -19.9, -21.5, -27.1]]
```

Q3

Exercise 15

Q3. Eq(3.8)

Exercise 15

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Adding C to all Rewards,

$$G_t' = (R_{t+1} + C) + (R_{t+2} + C) + \cdots = \sum_{k=0}^{\infty} \gamma^k (R_{t+k+1} + C)$$

$$G_t' = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} + \boxed{\sum_{k=0}^{\infty} \gamma^k \cdot C} = V_C$$

$$\boxed{G_t' = G_t + V_C}$$

$$G_t' = G_t + C\left(\frac{1}{1-\gamma}\right)$$

$$V_C = \begin{cases} \gamma = 1 & \infty \\ \gamma < 1 & \dfrac{C}{1-\gamma} \end{cases}$$

$V_\pi(s) = \mathbb{E}[G_t \mid S_t = s]$ (Bellman's equations

$= \cancel{\mathbb{E}}$

$$V_\pi'(s) = \mathbb{E}[G_t' \mid S_t = s] = \mathbb{E}[G_t + V_C \mid S_t = s]$$

$$= \mathbb{E}[G_t \mid S_t = s] + \mathbb{E}[V_C \mid S_t = s]$$

$$= \mathbb{E}[G_t \mid S_t = s] + V_C \quad (\text{constant})$$

$$V_C = \begin{cases} \infty & \gamma = 1 \\ \dfrac{C}{1-\gamma} & 0 < \gamma < 1. \\ C & \gamma = 0 \quad (\text{only for } R_{t+1}) \end{cases}$$

We see that the values for each state are not affected relatively under all policies. Therefore the sign's don't matter in the case ~~when~~ of continuous tasks.

Exercise 16

Ex 16. $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

$$G_t' = (R_{t+1}+c) + (R_{t+2}+c) + \cdots = \sum_{k=t+1}^{T} \gamma^{k-t-1}(R_k+c)$$

$$\Rightarrow G_t' = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k + c \sum_{k=t+1}^{T} \gamma^{k-t-1}$$

$$= G_t + c\left[\gamma^0 + \cdots + \gamma^{T-t-1}\right]$$

$$= G_t + \underbrace{c\left(\frac{1-\gamma^{T-t}}{1-\gamma}\right)}_{=V_c}$$

$$V_\pi'(s) = \mathbb{E}[G_t' \mid S_t = s]$$

$$= \mathbb{E}[G_t + V_c \mid S_t = s]$$

$$= \mathbb{E}[G_t \mid S_t = s] + \mathbb{E}[V_c \mid S_t = s]$$

$$= V_\pi(s) + \cancel{\mathbb{E}\left[c\left(\frac{1-\gamma^{T-t}}{1-\gamma}\right)\Big| S_t\right]} \quad \mathbb{E}\left[c\left(\frac{1-\gamma^{T-t}}{1-\gamma}\right)\Big| S_t = s\right]$$

this changes with time step & we see that the Value functions are ~~a~~ relatively ~~dif~~, ~~d~~ different.

Consider the case of Robot finding the fastest way to reach the terminal point. If the rewards ~~for going wrong~~ taking more distance is ~~+0. -1~~ ~~+1~~, negative, ~~it will~~ for using more distance is -1 for each extra (step/distance) maximizing it will lead to reach the end. But in case of +1 reward, the ~~max~~ maximizing the reward will not lead to the shortest path.

Q4 The Bellman's optimality Equation is non-linear. Solved it by using Policy iteration methods given in the code.

The Final Output for both the methods:

Policy Iteration:

Value matrix:

```
[[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]]
```

Policy:

```
['R', 'ULDR', 'L', 'ULDR', 'L', 'UR', 'U', 'UL', 'L', 'L', 'UR', 'U', 'UL', 'UL', 'UL', 'UR', 'U', 'UL', 'UL', 'UL',
'UR', 'U', 'UL', 'UL', 'UL']
```

Q6 Coded both policy iteration and value iteration and run all to get the output. We get the optimal policy in both the cases.

```
['', 'L', 'L', 'LD', 'U', 'UL', 'ULDR', 'D', 'U', 'ULDR', 'DR', 'D', 'UR', 'R', 'R', '']
```

Successive shots in Policy iteration

```
[[  0. -13. -19. -21.]
 [-13. -17. -19. -19.]
 [-19. -19. -17. -13.]
 [-21. -19. -13.   0.]]
['', 'L', 'L', 'L', 'U', 'UL', 'L', 'D', 'U', 'U', 'D', 'D', 'U', 'R', 'R', '']
=========
[[ 0.  0. -1. -2.]
 [ 0. -1. -2. -1.]
 [-1. -2. -1.  0.]
 [-2. -1.  0.  0.]]
['', 'L', 'L', 'LD', 'U', 'UL', 'ULDR', 'D', 'U', 'ULDR', 'DR', 'D', 'UR', 'R', 'R', '']
=========
[[ 0.  0. -1. -2.]
 [ 0. -1. -2. -1.]
 [-1. -2. -1.  0.]
 [-2. -1.  0.  0.]]
['', 'L', 'L', 'LD', 'U', 'UL', 'ULDR', 'D', 'U', 'ULDR', 'DR', 'D', 'UR', 'R', 'R', '']
=========
```

Successive shots in Value iteration

```
[[ 0.  0. -1. -1.]
 [ 0. -1. -1. -1.]
 [-1. -1. -1.  0.]
 [-1. -1.  0.  0.]]
['', 'L', 'L', 'ULDR', 'U', 'UL', 'ULDR', 'D', 'U', 'ULDR', 'DR', 'D', 'ULDR', 'R', 'R', '']
=========
[[ 0.   0.  -1.  -1.9]
 [ 0.  -1.  -1.9 -1. ]
 [-1.  -1.9 -1.   0. ]
 [-1.9 -1.   0.   0. ]]
['', 'L', 'L', 'LD', 'U', 'UL', 'ULDR', 'D', 'U', 'ULDR', 'DR', 'D', 'UR', 'R', 'R', '']
=========
[[ 0.   0.  -1.  -1.9]
 [ 0.  -1.  -1.9 -1. ]
 [-1.  -1.9 -1.   0. ]
 [-1.9 -1.   0.   0. ]]
['', 'L', 'L', 'LD', 'U', 'UL', 'ULDR', 'D', 'U', 'ULDR', 'DR', 'D', 'UR', 'R', 'R', '']
=========
```

We fix the bug by keeping a probability distribution for the actions taken in the policy. As more than one action at a step can have the optimal value, we will divide the probability to each of the actions as (1/ no of actions giving optimal value). Then we would not have a random selection of action when iterating through the policy iteration algorithm and the algorithm will converge.