Name: Sidharth Banerjee

ID: 1001622703

CSE 4309: Introduction to Machine Learning

Assignment 7

Task 1

python value_iteration.py environment2.txt -0.04 1 20

0.812, 0.868, 0.918, 1.000

0.762, 0.000, 0.660,-1.000

0.705, 0.655, 0.611, 0.387

python value_iteration.py environment2.txt -0.04 0.9 20

0.509, 0.650, 0.795, 1.000

0.399, 0.000, 0.486,-1.000

0.296, 0.254, 0.345, 0.130

Task 2

python q_learning.py environment2.txt -0.04 1 1000 20

python q_learning.py environment2.txt -0.04 0.9 1000 20

Task 3

I would assign a reward of -0.04 for non-terminal states. This reward value is low enough for the algorithm to encourage exploration. If the reward value is less negative than this, there will be too much exploration which help the algorithm learn a lot of the different kind of moves but will never prioritize exploiting the data to win. At the same time, if the reward is too negative, it will discourage exploration and may take very long to discover that some moves are better than the ones it already knows.

I would use a gamma value of 0.3. Since we only know the reward of a state when we reach that state, we should prioritize immediate rewards (for example taking the opponent's Queen). Since we know that

having more pieces will have a positive impact on the algorithm winning the game, immediate rewards maybe more important.

Task 4

Part a:

Utility(UP) = 0.8*1 + 0.2*-0.04 = 0.792

Utility(DOWN) = 0.8*-1 + 0.2*-0.04 = -0.808

Utility(LEFT) = 0.8*-0.04 + 0.1*1 + 0.1*-1 = -0.032

Utility(RIGHT) = 0.8*-0.04 + 0.1*1 + 0.1*-1 = -0.032

Max(0.792, -0.808, -0.032, -0.032) = 0.798

Utility(2, 2) = -0.04 + 0.9*0.798 = 0.68

Part b:

Utility(UP) = 0.8*1 + 0.2*r

Utility(LEFT) = 0.8*r

For UP not to be optimal, any other action should be more optimal. In this case, we find r values for which LEFT is optimal.

$0.8r > 0.8 + 0.2r$

$0.6r > 0.8$

$r > 0.8/0.6$

$r > 1.33$