CS440 Final exam: Question 1

Mustafa Sadiq (ms3035)

May 10, 2021

Introduction

Given the following landscape:

```
[['S' ' ' ' ' ' ' ' ' ']
[' ' ' ' ' ' ' ' ' ']
['X' ' ' ' ' ' ' ' ']
['X' ' ' ' ' ' ' ' ']
['X' ' ' ' ' ' ' ' ']
['X' ' ' ' ' ' ' ' ']
['X' ' ' ' ' ' ' 'X']
[' ' ' ' ' ' ' ' 'X']
[' ' ' ' ' ' ' ' 'X']
[' ' ' ' ' ' ' ' 'X']
['X' ' ' ' ' ' ' 'X']
['X' ' ' ' ' ' ' ' ']
['X' ' ' ' ' ' ' ' ']
['X' ' ' ' ' ' ' ' ']
['X' ' ' ' ' ' ' ' ']
[' ' ' ' ' ' ' ' ']
['G' ' ' ' ' ' ' ']]
```

Figure 1: Landscape, G=goal, X=ravine

Whenever we move in a given direction there is a 0.1 probability that we can end up in one of the orthogonal directions. Moreover, each step costs 1, and walking into a ravine has a reward -1000 (and termination). We begin at S and our goal is to reach G (termination). We set the reward at G as 0, the maximum possible reward. C(x) for any x, is the expected reward.

Answer to part 1

$C(S) = max\{(-1+0.9*C(1,0)+0.1*C(0,1)), (-1+0.9*C(0,1)+0.1*C(1,0))\}$

Answer to part 2

$C(1,0) = max\{(-1+0.9*C(S)+0.1*C(1,1)), (0.8*C(1,1)+0.1*C(S)-101)\}$

Answer to part 3

We define the following:

> $A(x)$ being the all the possible actions that can be taken from x.
>
> x' being the position after taking action a at position x
>
> x'' or (x'' and x''') being the position/s we can slip into after taking action a at position x depending on if the orthogonal direction is blocked by a wall or not.

At any given x, depending on if the orthogonal direction is blocked:

> If orthogonal direction blocked by wall:
>
> $$C(x) = max_{a \in A(x)}[-1 + 0.9 * C(x') + 0.1 * C(x'')]$$
>
> If orthogonal direction not blocked by wall:
>
> $$C(x) = max_{a \in A(x)}[-1 + 0.8 * C(x') + 0.1 * C(x'') + 0.1 * C(x''')]$$

Answer to part 4

If we had the value of $C(x)$ for each cell, at next step we would move to the neighbour with the highest $C(x)$, or arbitrarily if we have more than one neighbour with the highest $C(x)$. So the best direction to move in cell S will be either (1, 0) or (0, 1) depending on which one has the higher $C(x)$.

Answer to part 5

To find C(x) for each x, we use our equation depending on if diagonal is blocked from x and the value iteration method, and get:

```
[[  -28.45308   -27.25716   -26.27614   -26.55648   -26.99125]
[  -29.2164    -26.08628   -24.86847   -25.28718   -25.92845]
[-1000.        -24.65781   -23.41391   -24.00936   -24.88859]
[-1000.        -23.18055   -21.93399   -22.72389   -23.87517]
[-1000.        -21.6757    -20.42944   -21.42872   -22.89197]
[-1000.        -20.14097   -18.89874   -20.12072   -23.61485]
[-1000.        -18.54402   -17.34072   -18.58853 -1000.     ]
[  -17.72472   -16.57351   -15.78433   -17.03887 -1000.     ]
[  -16.74152   -15.27826   -14.27886   -15.52552 -1000.     ]
[  -17.46439   -13.97027   -12.74811   -13.98539 -1000.     ]
[-1000.        -12.43963   -11.19068   -12.34356 -1000.     ]
[-1000.        -10.90065    -9.64045    -9.92477   -10.18272]
[-1000.         -9.44331    -8.19738    -8.67806    -9.10027]
[-1000.         -7.95337    -6.73155    -7.43537    -8.03607]
[-1000.         -6.23798    -5.24085    -6.19826    -6.99171]
[   -1.24996    -2.49962    -3.74653    -4.96875    -5.96875]
[    0.         -1.24996    -2.49962    -3.74653    -4.96875]]
```

Figure 2: C(x) for each x, -1000 = ravine cells

We can see that **C(S) = -28.45308**.

Answer to part 6

To determine best direction from each cell, we find the neighbour with the highest C(x). If more than one neighbour have a highest C(x), we move to any one of them arbitrarily:

```
[['>' 'V' 'V' 'V' 'V']
['>' 'V' 'V' 'V' 'V']
['X' 'V' 'V' 'V' 'V']
['X' 'V' 'V' 'V' '<']
['X' 'V' 'V' 'V' '<']
['X' 'V' 'V' 'V' '<']
['X' 'V' 'V' 'V' 'X']
['>' 'V' 'V' 'V' 'X']
['>' 'V' 'V' 'V' 'X']
['>' 'V' 'V' 'V' 'X']
['X' 'V' 'V' 'V' 'X']
['X' 'V' 'V' 'V' 'V']
['X' 'V' 'V' 'V' 'V']
['X' 'V' 'V' 'V' 'V']
['X' 'V' 'V' 'V' 'V']
['V' 'V' 'V' 'V' 'V']
['G' '<' '<' '<' '<']]
```

Figure 3: Policy, G=goal, X=ravine

Answer to bonus

We can find the cost threshold by finding the policy which is not similar to the one before, while decreasing ravine reward:

policy = policy when ravine reward is 0

for ravine_reward in range(-1, -10000, -1):

new policy = policy when ravine reward is ravine_reward

if new policy is equal to policy, answer is ravine_reward

else policy = new policy

Using this algorithm we find that the ravine reward threshold is -15.

The best directions to move when below this threshold are:

```
[['V' '<' '<' '<' '<']
['V' '<' '<' '<' 'V']
['X' '<' '<' '<' 'V']
['X' '<' '<' '<' 'V']
['X' '<' '<' 'V' 'V']
['X' '<' 'V' 'V' 'V']
['X' '<' 'V' 'V' 'X']
['V' 'V' 'V' 'V' 'X']
['>' 'V' 'V' 'V' 'X']
['>' 'V' 'V' 'V' 'X']
['X' 'V' 'V' 'V' 'X']
['X' 'V' 'V' 'V' 'V']
['X' 'V' 'V' 'V' 'V']
['X' 'V' 'V' 'V' 'V']
['X' 'V' 'V' 'V' 'V']
['V' 'V' 'V' 'V' 'V']
['G' '<' '<' '<' '<']]
```

Figure 4: Policy when below reward threshold, G=goal, X=ravine

Addendum

I have read and abided by the rules laid out in the assignment prompt, I have not used anyone else's work for my project, my work is only mine.

Signed by: Mustafa Sadiq