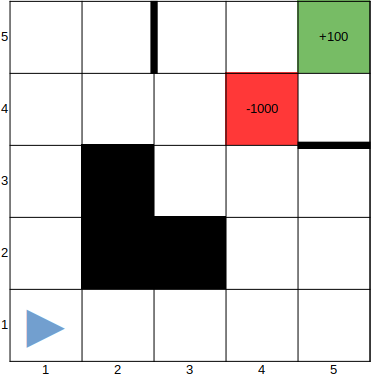
**CPSC 4420/6420: ARTIFICIAL INTELLIGENCE**

ASSIGNMENT 2 NAME:

Charanjit Singh

Consider the following puzzle. The green and red states are both terminal states, with the rewards as shown (so we can consider the green state the “goal”, and the red a “game over” state with a large negative reward). Thick borders between cells represent walls that the robot player cannot cross, and the black squares contain obstacles and cannot be entered. The robot player is represented by the blue triangle, and the direction the triangle points is the way the robot is facing.



Let’s represent a state with (x,y,d), where x and y represent the horizontal and vertical positions (i.e. location), and d represents the direction the robot is facing (1: up, 2: down, 3: left, and 4: right).

The robot can take the following actions:

A1: Move one cell forward in the direction it is facing. Cost: 1.5 A2: Move two cells forward in the direction it is facing. Cost: 2 A3: Turn to its left and stay in the same cell. Cost: 0.5

A4: Turn to its right and stay in the same cell. Cost: 0.5

Note that each action has a different cost value. This can also be considered an immediate negative reward. For example, we have R(s,A1,s’) = -1.5. The cost is evaluated on the current state, (the state the robot is in when it begins the action, not the one it lands on after

performing the action). In the same way, the value of state V(s) represents the value of the current state and you should initialize the algorithm with V1(5,5,x)=+100, V1(4,4,x)=-1000 (for x=1,2,3,4 representing the robot orientation/direction), and zero for all other states.

So, for example, if the robot is in state (4,1,4), it means that it is in location (4,1) and facing right. The result of possible actions for this state are as follows:

A1 (move 1 cell forward) --> (5,1,4)

A2 (move 2 cells forward) --> impossible remains in the current state (4,1,4) A3 (turn left) --> (4,1,1) : the robot stays in (4,1) but now faces up

A4 (turn right) --> (4,1,2) : the robot stays in (4,1) but now faces down

A move is impossible if it would result in landing on a blocked cell, like (2,2), (2,3), or (3,2), or if it would result in crossing a barrier, like moving from state (2,5) to (3,5), or (5,3) to (5,4). A move that would take the robot outside of our 5x5 grid is also impossible.

Note that we have more states than the number of cells, because the robot facing a different direction produces a new state, even if it does not change location. In the example above, if we move to (4,1,1), where the robot is facing up, this is a different state from the one we were in, (4,1,4), even though the robot has not moved cells.

1. If there is no living reward/penalty, no noise, and no discount (gamma = 1), use your common sense to find the best possible route from (1,1) to (5,5).

Ans) Considering no living reward/penalty, no noise and discount factor = 1, the best possible route will be the one which costs the least. So, the following path should be followed for most optimal results:

Chart

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Current State | Action | Resulting State | Cost |
| (1,1,4) | A3 | (1,1,1) | -0.5 |
| (1,1,1) | A1 | (1,2,1) | -1.5 |
| (1,2,1) | A2 | (1,4,1) | -2.0 |
| (1,4,1) | A4 | (1,4,4) | -0.5 |
| (1,4,4) | A2 | (3,4,4) | -2.0 |
| (3,4,4) | A3 | (3,4,1) | -0.5 |
| (3,4,1) | A1 | (3,5,1) | -1.5 |
| (3,5,1) | A4 | (3,5,4) | -0.5 |
| (3,5,4) | A2 | (5,5,4) | -2.0 |
|  |  | Total Cost | -11 |

1. With no discount (gamma = 1), no living reward, and no noise, use the Value Iteration Algorithm with 100 iterations to update the optimal values for each state and print the result [only for the first 10 iterations] in the following format:

|  |  |  |
| --- | --- | --- |
| iter 1:  state (1,1,1)  state (1,1,2)  … | V = (some value) V = (some value) | Best Action: Ai (where i is some number 1-4) Best Action: Aj |
| state (5,5,4) | V = (some value) |  |
| iter 2:  state (1,1,1) | V = (some value) | Best Action: Ai (where i is some number 1-4) |
| state (1,1,2)  …  state (5,5,4) | V = (some value)  V = (some value) | Best Action: Aj |

If two actions are tied for best, you can select one at random or always choose the one with the smallest index.

Ans) The result of first 10 value iterations is attached at the end.

Click on the following hyperlink for result:

Refer to HW2\_B file for python code.

Text

Description automatically generated

Figure : Screenshot of Value Iteration results for reference

The optimal policy recommended is shown in figure 2:

Text

Description automatically generated

Figure : Optimal path recommended by the policy from state (1,1,4) to terminal state

1. If you start from state (1,1,4) and follow the optimal policy you found in part B, does it follow the same path you proposed in part A?

Ans) **Yes**, it follows the same path as proposed in part A.

Text

Description automatically generated

Figure : Screenshot of path followed by optimal policy found in part B

1. Repeat part B with the same assumptions, except for gamma = 0.8 (discount factor). Compare the results with that from part B. Do they match?

Ans) The result of first 10 value iterations is attached at the end.

Click on the following hyperlink for result:

Refer to HW2\_D file for python code.

Comparing the results with part B, although the values of V\* are different for cases when discount factor = 1 (for Part B) or

discount factor = 0.8 (for Part D).

But, the best action obtained by following the path with maximum expected utility, i.e.

A picture containing text

Description automatically generated

is **same for both parts B and D**.

So, the action A to be followed in a particular state S, will be the same if we choose to follow obtained in part D or in part B.

Text

Description automatically generated

Figure : Screenshot of Value Iteration results for reference

A screenshot of a computer

Description automatically generated with medium confidence

Figure : Optimal Path for discount factor = 0.8

1. Repeat part B with the same assumptions, except for gamma = 0.2. Compare the results with that from parts B and D. Do they match?

Ans) The result of first 10 value iterations is attached at the end.

Click on the following hyperlink for result:

Refer to HW2\_E file for python code.

Comparing with results from Parts B and D, the results (optimal paths recommended) do **NOT** match.

As the value of discount factor (gamma) has decreased considerably, the utility of future rewards has reduced drastically.

This tends the agent more towards myopic behavior, i.e. the agent will consider the immediate rewards more strongly as compared to future rewards.

So, for the starting states (1,1,1), (1,1,2), (1,1,3) and (1,1,4), the best action recommended at V\* values convergence is A3 (which costs the least [-0.5]).

This makes the agent fall in an infinite loop of rotating left while in the cell (1,1), as can be seen in figure 7.

Text

Description automatically generated

Figure : Screenshot of Value Iteration results for reference

Text

Description automatically generated with low confidence

Figure : Path followed by agent for discount factor = 0.2 (infinite loop)

1. **(Optional for 4420)** Repeat part B, but this time with noise = 0.2, and gamma = 0.9 and no living reward. With a noise of 0.2, every time you take an action, the result will be the expected action with Probability 0.8 (80%), but 20% of the time, the robot will instead take a different action (taken randomly out of unexpected actions, with equal probability). If the action is impossible, it remains in the same cell.

For example, if we are in state (4,1,4), location (4,1) and facing right, and we take action A1 (moving one cell forward), the resulting state will be:

s’= (5,1,2) with probability 0.8 [because A1 is rendered]

s’= (4,1,4) with probability 0.2/3 [renders A2 which is impossible] s’= (4,1,1) with probability 0.2/3 [because A3 is rendered]

s’= (4,1,2) with probability 0.2/3 [because A4 is rendered] Compare the results with that of the previous parts and explain observations.

Ans) The result of first 10 value iterations is attached at the end.

Click on the following hyperlink for result:

Refer to HW2\_F file for python code.

In the previous parts, there was no noise which meant the actions were executed with a probability of 1. As in this case, there is a noise of 20%, there is a risk of falling in negative terminal state (4,4) which makes the agent skip the states (3,4,4) and (4,3,1) (which now have a negative V\*).

Considering initial state as (1,1,4), the agent adopts a different path, via (3,1,4).

It avoids going upwards as it will eventually come across (3,4,4).

It avoids going through (4,1,1) as it will eventually come across (4,3,1).

So, it follows the safest option available to it which is guaranteed to give maximum expected utility. The path is shown in figure 8.

Text

Description automatically generated

Figure : Optimal path followed for discount factor = 0.9 and noise = 0.2