Report

- Tim (Kyoung Tae) Kim
- COSC 76
- PA 6
- Nov 12, 2021
- '22

Description

(a) Description: How do your implemented algorithms work? What design decisions did you make? How you laid out the problems? In particular, for this specific assignment, explain your model precisely, and explain exactly how you compute each new distribution of possible states.

For this problem, I used a matrix (row, col) convention as the probabilities were created with numpy arrays. Thus, for this problem, I assume that the top left corner of the maze is (0, 0) and all robot positions and movements follow this assumption. As with previous problems, the colors are assigned integer values and converted back to strings for printing.

Once the maze has been loaded and each cell is randomly assigned a color, I create a uniform distribution of the probabilities given the maze dimension and number of walls. Then, the `move_robot` function moves the robot in random directions, staying in place if there is an obstacle or is out of bounds. The sensor measurements are also recorded with the provided accuracy of 0.88. This color sequence is provided as the parameter for the filtering algorithm.

The filtering algorithm computes the probability distribution for each iteration as follows. First, I compute the probabilities based on the sensor measurements. Given a color, the probability of being a particular cell is 0.88 if the sensor measurement matches the true color of the cell and 0.04 otherwise (as the sensor could be wrong). Thus, the prediction matrix is a 4 x 4 matrix comprised of 0.88 and 0.04. Then for each cell of the maze, I compute the transition matrix by adding 0.25 to the neighboring cells if the robot can move. This means that 0.25 is added to the current cell if the robot is at a corner or next to the wall. Thus, the transition matrix is a 4 x 4 matrix comprised of 0.25 and 0.5. The transition matrix for each cell is multiplied by the probability of being at the current cell, and the sum of these probabilites for each cell is stored in an intermediate matrix. Finally, we multiply this intermediate matrix to

the prediction matrix to calculate the new probability distribution, which is then normalized before being used in the next iteration.

Evaluation

(b) Evaluation: Do your implemented algorithms actually work? How well? If it doesn't work, can you tell why not? What partial successes did you have that deserve partial credit?

As shown below, the filtering algorithm works as intended. To test the filtering algorithm, I added several print statements for each iteration. First, the probability distribution and difference compared to the previous distribution can be examined to see how the probabilities change with each step. After each iteration, the probabilities accumulate over time in certain locations of the maze, although it never reaches certainty.

By displaying the actual robot location, we can see that areas of the highest probability correspond to the actual location of the robot. To make this observation clear, I added an additional display of the maze with `$` placed in cells that have probabilities equal to or geater than the third highest probability value in the cell.

Finally, the sensed and true color value of the current cell, as well as the current path in terms of colors, have been added to show that the robot is in one of the cells in the highest probabilities except for cases when the sensor has made an erroneous measurement. In the case of a wrong sensor reading, it may mean that the robot to not be in one of the cells marked by `$` . However, this is corrected after a few steps, provided that the subsequent measurements are correct. Thus, although the probability accumulates in certain locations of the maze after each iteration, it may not be as high as previous iterations because it has just received a wrong sensor reading.

The addition of walls in the maze does help the probilities converge on fewer points with higher values. This is demonstrated in the output below on maze4.maz. Here, we see that since there was no wrong sensor measurement, we are able to state with quite high probability that the robot is in cell (2, 0).

Output

```
------------------------------------------------------------------------
-------
Hidden Markov Model Filtering:
------------------------------------------------------------------------
-------
Intial distribution:
[[0.06666667 0.06666667 0.06666667 0.06666667]
 [0.06666667 0.06666667 0.06666667 0.06666667]
```

```
 [0.06666667 0.06666667 0.         0.06666667]
 [0.06666667 0.06666667 0.06666667 0.06666667]]
....
.R..
..#.
....

Initial Location
(r, c) = (1, 1)
------------------------------------------------------------------------
-------
Prob Distribution after 1 x filter
[[0.01  0.01  0.222 0.01 ]
 [0.01  0.01  0.222 0.222]
 [0.01  0.222 0.    0.01 ]
 [0.01  0.01  0.01  0.01 ]]
Difference w/ previous distriubtion:
[[-0.057 -0.057  0.156 -0.057]
 [-0.057 -0.057  0.156  0.156]
 [-0.057  0.156  0.    -0.057]
 [-0.057 -0.057 -0.057 -0.057]]
Robot location in maze
....
....
.R#.
....

Locations w/ high prob
..$.
..$$
.$..
....

location:  (2, 1)
Sensed color: Red // Actual color: Red
Current path:  ['Red']
[['G' 'Y' 'R' 'G']
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
------------------------------------------------------------------------
-------
Prob Distribution after 2 x filter
[[0.003 0.359 0.03  0.03 ]
 [0.003 0.03  0.044 0.03 ]
 [0.359 0.016 0.    0.016]
 [0.003 0.016 0.003 0.058]]
Difference w/ previous distriubtion:
```

```
[[-0.007  0.349 -0.192  0.02 ]
 [-0.007  0.02  -0.178 -0.192]
 [ 0.349 -0.206  0.     0.006]
 [-0.007  0.006 -0.007  0.047]]
Robot location in maze
....
....
R.#.
....

Locations w/ high prob
.$..
....
$...
...$

location:  (2, 0)
Sensed color: Yellow // Actual color: Yellow
Current path:  ['Red', 'Yellow']
[['G' 'Y' 'R' 'G']
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
--------------------------------------------------------------------------
-------
Prob Distribution after 3 x filter
[[0.013 0.015 0.365 0.004]
 [0.014 0.015 0.106 0.095]
 [0.014 0.332 0.    0.004]
 [0.014 0.001 0.003 0.005]]
Difference w/ previous distriubtion:
[[ 0.011 -0.344  0.335 -0.026]
 [ 0.012 -0.015  0.062  0.065]
 [-0.346  0.316  0.    -0.012]
 [ 0.011 -0.015  0.    -0.053]]
Robot location in maze
....
....
.R#.
....

Locations w/ high prob
..$.
..$.
.$..
....

location:  (2, 1)
```

```
Sensed color: Red // Actual color: Red
Current path:  ['Red', 'Yellow', 'Red']
[['G' 'Y' 'R' 'G']
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
--------------------------------------------------------------------------
-------
Prob Distribution after 4 x filter
[[0.05  0.017 0.02  0.422]
 [0.05  0.019 0.024 0.009]
 [0.015 0.015 0.    0.004]
 [0.038 0.316 0.    0.001]]
Difference w/ previous distriubtion:
[[ 0.037  0.002 -0.345  0.418]
 [ 0.036  0.004 -0.082 -0.086]
 [ 0.002 -0.318  0.     0.   ]
 [ 0.024  0.314 -0.002 -0.004]]
Robot location in maze
....
....
..#.
.R..

Locations w/ high prob
...$
$...
....
.$..

location:  (3, 1)
Sensed color: Green // Actual color: Green
Current path:  ['Red', 'Yellow', 'Red', 'Green']
[['G' 'Y' 'R' 'G']
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
--------------------------------------------------------------------------
-------
Prob Distribution after 5 x filter
[[0.013 0.008 0.036 0.066]
 [0.01  0.175 0.005 0.035]
 [0.009 0.028 0.    0.03 ]
 [0.031 0.028 0.526 0.   ]]
Difference w/ previous distriubtion:
[[-0.037 -0.009  0.016 -0.356]
 [-0.04   0.156 -0.018  0.026]
 [-0.006  0.013  0.     0.026]
```

```
 [-0.007 -0.288  0.526 -0.   ]]
Robot location in maze
....
....
..#.
..R.

Locations w/ high prob
...$
.$..
....
..$.

location:  (3, 2)
Sensed color: Blue // Actual color: Blue
Current path:  ['Red', 'Yellow', 'Red', 'Green', 'Blue']
[['G' 'Y' 'R' 'G']
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
------------------------------------------------------------------------
-------
Prob Distribution after 6 x filter
[[0.002 0.23  0.005 0.009]
 [0.009 0.002 0.011 0.006]
 [0.077 0.011 0.    0.004]
 [0.004 0.028 0.049 0.552]]
Difference w/ previous distriubtion:
[[-0.011  0.222 -0.031 -0.057]
 [-0.001 -0.173  0.006 -0.028]
 [ 0.068 -0.017  0.    -0.026]
 [-0.026 -0.    -0.478  0.552]]
Robot location in maze
....
....
..#.
...R

Locations w/ high prob
.$..
....
$...
...$

location:  (3, 3)
Sensed color: Yellow // Actual color: Yellow
Current path:  ['Red', 'Yellow', 'Red', 'Green', 'Blue', 'Yellow']
[['G' 'Y' 'R' 'G']
```

```
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
------------------------------------------------------------------------
-------
Prob Distribution after 7 x filter
[[0.007 0.007 0.007 0.001]
 [0.003 0.162 0.001 0.001]
 [0.003 0.003 0.    0.35 ]
 [0.003 0.003 0.418 0.032]]
Difference w/ previous distriubtion:
[[ 0.005 -0.224  0.002 -0.008]
 [-0.007  0.159 -0.011 -0.005]
 [-0.074 -0.008  0.     0.346]
 [-0.001 -0.025  0.369 -0.52 ]]
Robot location in maze
....
....
..#.
..R.

Locations w/ high prob
....
.$..
...$
..$.

location:  (3, 2)
Sensed color: Blue // Actual color: Blue
Current path:  ['Red', 'Yellow', 'Red', 'Green', 'Blue', 'Yellow', 'Bl
ue']
[['G' 'Y' 'R' 'G']
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
------------------------------------------------------------------------
-------
Prob Distribution after 8 x filter
[[0.029 0.01  0.001 0.012]
 [0.218 0.001 0.01  0.02 ]
 [0.001 0.01  0.    0.042]
 [0.015 0.535 0.05  0.047]]
Difference w/ previous distriubtion:
[[ 0.022  0.004 -0.006  0.011]
 [ 0.215 -0.161  0.009  0.019]
 [-0.002  0.006  0.    -0.308]
 [ 0.012  0.533 -0.369  0.015]]
Robot location in maze
```

```
....
....
..#.
.R..

Locations w/ high prob
....
$...
....
.$$.

location:  (3, 1)
Sensed color: Green // Actual color: Green
Current path:  ['Red', 'Yellow', 'Red', 'Green', 'Blue', 'Yellow', 'Bl
ue', 'Green']
[['G' 'Y' 'R' 'G']
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
--------------------------------------------------------------------
-------
Prob Distribution after 9 x filter
[[0.154 0.001 0.001 0.024]
 [0.134 0.006 0.001 0.002]
 [0.006 0.013 0.    0.004]
 [0.305 0.328 0.017 0.005]]
Difference w/ previous distriubtion:
[[ 0.125 -0.009 -0.     0.012]
 [-0.084  0.005 -0.009 -0.018]
 [ 0.005  0.004  0.    -0.038]
 [ 0.29  -0.207 -0.033 -0.043]]
Robot location in maze
....
....
..#.
R...

Locations w/ high prob
$...
....
....
$$..

location:  (3, 0)
Sensed color: Green // Actual color: Green
Current path:  ['Red', 'Yellow', 'Red', 'Green', 'Blue', 'Yellow', 'Bl
ue', 'Green', 'Green']
[['G' 'Y' 'R' 'G']
```

```
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
------------------------------------------------------------------------
-------
Prob Distribution after 10 x filter
[[0.025 0.202 0.002 0.003]
 [0.017 0.008 0.001 0.002]
 [0.571 0.02  0.    0.001]
 [0.054 0.038 0.021 0.037]]
Difference w/ previous distriubtion:
[[-0.129  0.201  0.001 -0.021]
 [-0.117  0.002 -0.    -0.   ]
 [ 0.565  0.007  0.    -0.003]
 [-0.251 -0.291  0.004  0.032]]
Robot location in maze
....
....
R.#.
....

Locations w/ high prob
.$..
....
$...
$...

location:  (2, 0)
Sensed color: Yellow // Actual color: Yellow
Current path:  ['Red', 'Yellow', 'Red', 'Green', 'Blue', 'Yellow', 'Bl
ue', 'Green', 'Green', 'Yellow']
[['G' 'Y' 'R' 'G']
 ['G' 'B' 'R' 'R']
 ['Y' 'R' '#' 'B']
 ['G' 'G' 'B' 'Y']]
------------------------------------------------------------------------
-------
Colors sequence [0, 2, 0, 1, 3, 2, 3, 1, 1, 2]
Robot positions [(1, 1), (2, 1), (2, 0), (2, 1), (3, 1), (3, 2), (3, 3
), (3, 2), (3, 1), (3, 0), (2, 0)]
```