CS520 Project 3 - Probabilistic Sensing

Arun Subbaiah • as3590

1.

Prior to any interaction with the environment, what is the probability of the target being in a given cell?

Prior to any interaction with the environment, we have no reason to believe that the probability of the target being in any particular cell is more likely than the other. Hence, the probability of the target being in a cell is uniformly distributed across the whole grid. Since the target can be in any cell in the grid,

$$Probability\ of\ target\ being\ in\ a\ cell\ = \frac{1}{Number\ of\ cells\ in\ the\ gridworld} = \frac{1}{dim * dim}$$

2.

Let $P_{i,j}(t)$ be the probability that cell (i, j) contains the target, given the observations collected up to time t. At time t + 1, suppose you learn new information about cell (x, y). Depending on what information you learn, the probability for each cell needs to be updated. What should the new $P_{i,j}(t + 1)$ be for each cell (i, j) under the following circumstances:

1. *At time t + 1 you attempt to enter (x, y) and find it is blocked?*
2. *At time t + 1 you attempt to enter (x, y), find it unblocked, and also learn its terrain type?*
3. *At time t + 1 you examine the cell (x, y) of terrain type flat, and fail to find the target?*
4. *At time t + 1 you examine the cell (x, y) of terrain type hilly, and fail to find the target?*
5. *At time t + 1 you examine the cell (x, y) of terrain type forest, and fail to find the target?*
6. *At time t + 1 you examine the cell (x, y) and find the target?*

You may take the density of blocked cells p to be known by the robot, as well as the frequencies of each terrain type, and the false-negativealse negative rates for each terrain type.

To answer these questions, let's get some notations down.

$B_{i,j}$ - Whether a cell (i, j) is blocked

- $B_{i,j} = True \rightarrow$ Cell (i, j) is blocked

- $B_{i,j} = False$ → Cell (i, j) is unblocked

$T_{i,j}$- Whether a cell (i, j) contains the target

- $T_{i,j} = True$ → Target is in the cell (i, j)
- $T_{i,j} = False$ → Target is not in the cell (i, j)

$F_{i,j}$- Whether the target is found after searching cell (i, j). Note that this is irrespective of whether the cell actually contains the target or not.

- $F_{i,j} = True$ → Target is found in the cell (i, j) after searching
- $F_{i,j} = False$ → Target is not found in the cell (i, j) after searching

$P_{i,j}$- Probability that a cell (i, j) contains the target

- $P_{i,j} = P(T_{i,j} = True)$

t - Observations or Information till time t

$P_{i,j}(t)$ - Probability that a cell (i, j) contains the target given observation till time t

- $P_{i,j}(t) = P(T_{i,j} = True \mid t)$

FNR_{tt} - False Negative Rate for terrain type tt. Probability that the target will be found after searching given the target is actually present in the cell.

- $FNR_{Flat} = P_{Flat}(F_{i,j} = False \mid T_{i,j} = False)$ = 0.2
- $FNR_{Hill} = P_{Hill}(F_{i,j} = False \mid T_{i,j} = False)$ = 0.5
- $FNR_{Forest} = P_{Forest}(F_{i,j} = False \mid T_{i,j} = False)$ = 0.8

2.1.

$P_{i,j}$ (t + 1), given that at time t + 1 you attempt to enter (x, y) and find it is blocked?

The new information, t + 1 we got is, $B_{x,y} = True$. This would mean that the target cannot be present in the cell (x, y). Therefore, $T_{x,y} = False$

$$P_{i,j}(t + 1) = P(T_{ij} = True \mid t + 1) = P(T_{i,j} = True \mid T_{x,y} = False)$$

Applying Bayes' rule,

$$P_{i,j}(t + 1) = \frac{P(T_{i,j} = True) * P(T_{x,y} = False \mid T_{i,j} = True)}{P(T_{x,y} = False)}$$

$P(T_{i,j} = True)$ is nothing but the prior probability till time t. Hence we can replace it with $P_{i,j}(t)$

$$P_{i,j}(t + 1) = \frac{P(T_{i,j} = True) * P(T_{x,y} = False \mid T_{i,j} = True)}{P(T_{x,y} = False)}$$

Since the target can be present in only one cell, we know that if the cell (i, j) contains the target, cell (x, y) does not contain it (as long as (x, y) and (i, j) does not refer to the same cell). Therefore, $P(T_{x,y} = False \mid T_{i,j} = True)$ = 1.

Substituting this in the above equation,

$$P_{i,j}(t + 1) = \frac{P(T_{i,j} = True)}{P(T_{x,y} = False)}$$

$$P_{i,j}(t + 1) = \frac{P(T_{i,j} = True)}{1 - P(T_{x,y} = True)}$$

$P(T_{i,j} = True)$ and $P(T_{x,y} = True)$ are nothing but the probabilities that the respective cells contain the target given the observations till time t. Hence we can replace it with prior notations.

$$P_{i,j}(t + 1) = \frac{P_{i,j}(t)}{1 - P_{x,y}(t)}$$

Hence, the new probability that the cell (i, j) contains the target given a cell (x, y) is found to be blocked is, { prior probability of cell (i, j) } divided by { 1 - prior probability of the cell (x, j) }

On the other hand, if both (x, y) and (i, j) refer to the same cell, then

$$P_{i,j}(t + 1) = 0$$

2.2.

$P_{i,j}$ (t + 1), given that at time t + 1 you attempt to enter (x, y), find it unblocked, and also learn its terrain type?

The new information, t + 1 we got is, $B_{x,y} = False$.

Based on our simplified assumption that finding a cell to be unblocked does not change the probability of that cell or other cell containing the target. Therefore,

$$P_{i,j}(t + 1) = P_{i,j}(t)$$

2.3.

$P_{i,j}$ (t + 1), given that at time t + 1 you examine the cell (x, y) of terrain type flat, and fail to find the target?

The new information, t + 1 we got is, $F_{x,y} = False$ for the terrain type *Flat*.

$$P_{i,j}(t + 1) = P(T_{ij} = True \mid t + 1) = P(T_{i,j} = True \mid F_{x,y} = False)$$

Applying Bayes' rule,

$$P_{i,j}(t + 1) = \frac{P\ (T_{i,j} = True) * P(F_{x,y} = False \mid T_{i,j} = True)}{P(F_{x,y} = False)}$$

$$P_{i,j}(t + 1) = \frac{P\ (T_{i,j} = True) * P(F_{x,y} = False \mid T_{i,j} = True)}{P(F_{x,y} = False \wedge T_{x,y} = True) + P(F_{x,y} = False \wedge T_{x,y} = False)}$$

$$P_{i,j}(t + 1) = \frac{P\ (T_{i,j} = True) * P(F_{x,y} = False \mid T_{i,j} = True)}{P(T_{x,y} = True) . P(F_{x,y} = False \mid T_{x,y} = True) + P(T_{x,y} = False) . P(F_{x,y} = False \mid T_{x,y} = False)}$$

$$P_{i,j}(t + 1) = \frac{P\ (T_{i,j} = True) * P(F_{x,y} = False \mid T_{i,j} = True)}{P(T_{x,y} = True) . FNR_{Flat} + P(T_{x,y} = False) .1}$$

Substituting $P(T_{x,y} = False) = 1 - P(T_{x,y} = True)$

$$P_{i,j}(t + 1) = \frac{P\ (T_{i,j} = True) * P(F_{x,y} = False \mid T_{i,j} = True)}{P(T_{x,y} = True) . FNR_{Flat} + 1 - P(T_{x,y} = True)}$$

$$P_{i,j}(t + 1) = \frac{P\ (T_{i,j} = True) * P(F_{x,y} = False \mid T_{i,j} = True)}{P(T_{x,y} = True) . FNR_{Flat} + 1 - P(T_{x,y} = True)}$$

$P\ (T_{i,j}\ =\ True)$ and $P\ (T_{x,y}\ =\ True)$ are nothing but the probabilities that the respective cells contain the target given the observations till time t. Hence we can replace it with prior notations.

$$P_{i,j}(t\ +\ 1)\ =\ \frac{P_{i,j}(t) * P(F_{x,y} = False\ |\ T_{i,j} = True)}{P_{x,y}(t)\ .\ FNR_{Flat} + 1 - P_{x,y}(t)}$$

$$P_{i,j}(t\ +\ 1)\ =\ \frac{P_{i,j}(t) * P(F_{x,y} = False\ |\ T_{i,j} = True)}{1 - P_{x,y}(t)\ [1 -\ FNR_{Flat}\ ]}$$

$P(F_{x,y}\ =\ False\ |\ T_{i,j}\ =\ True)\ =\ 1$ if (i, j) and (x, y) refer to different cells.

$P(F_{x,y}\ =\ False\ |\ T_{i,j}\ =\ True)\ =\ FNR_{Flat}$ if (i, j) and (x, y) refer to the same cell.

Therefore,

$$P_{i,j}(t\ +\ 1)\ =\ \frac{P_{i,j}(t)}{1 - P_{x,y}(t)\ [1 -\ FNR_{Flat}\ ]} \text{if (x, y) and (i, j) are different cells}$$

$$P_{i,j}(t\ +\ 1)\ =\ \frac{P_{i,j}(t) * FNR_{Flat}}{1 - P_{x,y}(t)\ [1 -\ FNR_{Flat}\ ]} \text{if (x, y) and (i, j) are same}$$

where $FNR_{Flat}\ =\ 0.2$

2.4.

$P_{i,j}$ (t + 1), given that at time t + 1 you examine the cell (x, y) of terrain type hilly, and fail to find the target?

The new information, t + 1 we got is, $F_{x,y}\ =\ False$ for the terrain type *Hilly*.

Using the result from the previous question,

$$P_{i,j}(t\ +\ 1)\ =\ \frac{P_{i,j}(t)}{1 - P_{x,y}(t)\ [1 -\ FNR_{Hill}\ ]} \text{if (x, y) and (i, j) are different cells}$$

$$P_{i,j}(t\ +\ 1)\ =\ \frac{P_{i,j}(t) * FNR_{Hill}}{1 - P_{x,y}(t)\ [1 -\ FNR_{Hill}\ ]} \text{if (x, y) and (i, j) are same}$$

where $FNR_{Hill} = 0.5$

2.5.

$P_{i,j}$ (t + 1), given that at time t + 1 you examine the cell (x, y) of terrain type forest, and fail to find the target?

The new information, t + 1 we got is, $F_{x,y} = False$ for the terrain type *Forest*.

Using the result from the previous question,

$$P_{i,j}(t + 1) = \frac{P_{i,j}(t)}{1 - P_{x,y}(t)\,[1 - FNR_{Forest}]} \text{if (x, y) and (i, j) are different cells}$$

$$P_{i,j}(t + 1) = \frac{P_{i,j}(t) * FNR_{Forest}}{1 - P_{x,y}(t)\,[1 - FNR_{Forest}]} \text{if (x, y) and (i, j) are same}$$

where $FNR_{Forest} = 0.8$

2.6.

$P_{i,j}$ (t + 1), given that at time t + 1 you examine the cell (x, y) and find the target?

The new information, t + 1 we got is, $T_{x,y} = True$

$$P_{i,j}(t + 1) = P(T_{ij} = True \mid t + 1) = P(T_{i,j} = True \mid T_{x,y} = True)$$

Applying Bayes' rule,

$$P_{i,j}(t + 1) = \frac{P\ (T_{i,j} = True) * P(T_{x,y} = True \mid T_{i,j} = True)}{P(T_{x,y} = True)}$$

$P(T_{x,y} = True \mid T_{i,j} = True) = 0$ if (i, j) and (x. y) refer to different cells.

$P(T_{x,y} = True \mid T_{i,j} = True) = 1$ if (i, j) and (x. y) refer to the same cell.

Therefore,

$P_{i,j}(t + 1) = 0$ if (i, j) and (x. y) are different cells

$$P_{i,j}(t+1) = 1 \text{ if (i, j) and (x. y) are the same}$$

3.

At time t, with probability $P_{i,j}(t)$ of the cell (i, j) containing the target, what is the probability of finding the target in cell (x, y):

1. *If (x, y) is hilly?*
2. *If (x, y) is flat?*
3. *If (x, y) is forest?*
4. *If (x, y) has never been visited?*

3.1.

Probability of finding the target in (x, y) if (x, y) is hilly

Since we will only be able to find the target if the target actually exists in the cell,

$$P\ (F_{x,y} = True\,|\,t) = P(F_{x,y} = True\ \wedge T_{x,y} = True\ |\,t)$$

$$P\ (F_{x,y} = True\,|\,t) = P(T_{x,y} = True\,|\,t)\,.\,P(F_{x,y} = True\,|\,T_{x,y} = True)$$

$$P\ (F_{x,y} = True\,|\,t) = P(T_{x,y} = True\,|\,t)\,.\,\left[1 - P(F_{x,y} = False\,|\,T_{x,y} = True)\right]$$

$$P\ (F_{x,y} = True\,|\,t) = P_{x,y}(t)\,.\left[1 - FNP_{Hill}\right]$$

Substituting FNP_{Hill},

$$P\ (F_{x,y} = True\,|\,t) = P_{x,y}(t)\,.\,[0.5]$$

3.2.

Probability of finding the target in (x, y) if (x, y) is flat

Following the same steps as the previous question,

$$P\ (F_{x,y} = True\,|\,t) = P_{x,y}(t)\,.\left[1 - FNP_{Flat}\right]$$

Substituting FNP_{Flat},

$$P\ (F_{x,y} = True \mid t)\ = P_{x,y}(t) \,.\, [0.8]$$

3.3.

Probability of finding the target in (x, y) if (x, y) is Forest

Following the same steps as the previous question,

$$P\ (F_{x,y} = True \mid t)\ = P_{x,y}(t) \,.\, \left[1\ -\ FNP_{Forest}\right]$$

Substituting FNP_{Forest},

$$P\ (F_{x,y} = True \mid t)\ = P_{x,y}(t) \,.\, [0.2]$$

3.4.

Probability of finding the target in (x, y) if (x, y) has never been visited

We know that the probability that a cell is blocked is 0.3. Therefore, $P(B_{x,y} = True)\ =\ 0.3$

We also know that if a cell is unblocked, it has an equal probability of being a Hilly, Flat or Forest terrain. Therefore,

$$\begin{aligned} &P(Terrain_{i,j} = Flat \mid B_{i,j}\ =\ False) \\ &\quad =\ P(Terrain_{i,j} = Hill \mid B_{i,j}\ =\ False) \\ &\quad =\ P(Terrain_{i,j} = Forest \mid B_{i,j}\ =\ False) \\ &\quad =\ 1/3 \end{aligned}$$

If a cell (x, y) has never been visited, then we do not know if the cell is blocked or unblocked. So we have to calculate the probability that the cell is unblocked and the probability that it has the target.

Plugging all the known probabilities, we get

$$P\ (F_{x,y} = True \mid t)\ =\ P(B_{x,y} = False)\,.\ 1/3\,.\ [0.5\ *\ P_{x,y}(t)\ +\ 0.8\ *\ P_{x,y}(t)\ +\ 0.2\ *\ P_{x,y}(t)\,]$$

$$P\ (F_{x,y} = True \mid t)\ =\ [1\ -\ P(B_{x,y} = True)]\,.\ 1/3\,.\ [1.5\ *\ P_{x,y}(t)]$$

$$P\ (F_{x,y} = True \mid t)\ =\ 0.7\ *\ 1/3\ *\ [1.5\ *\ P_{x,y}(t)]$$

$$P\ (F_{x,y} = True \mid t)\ =\ 0.35\,.\ P_{x,y}(t)$$

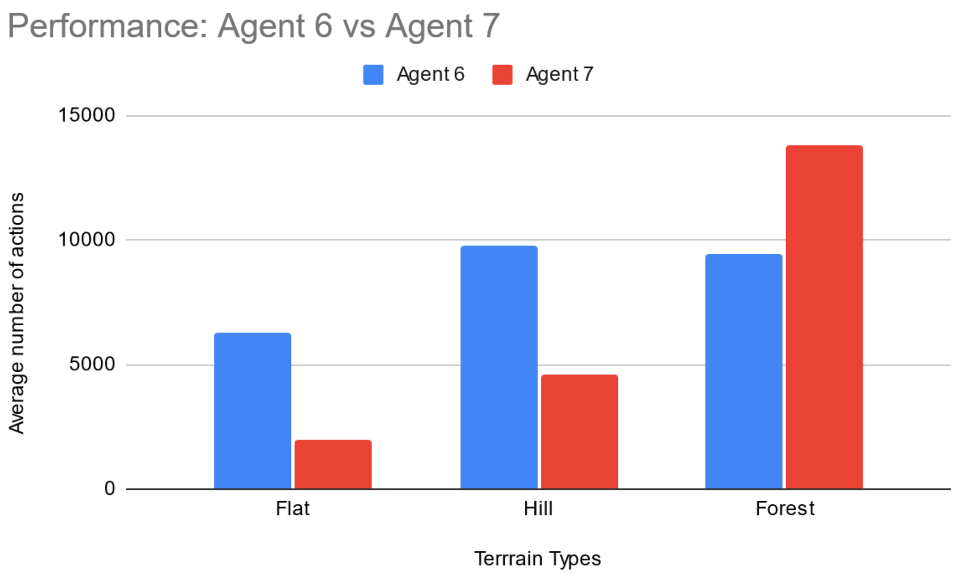
4.

Implement Agent 6 and 7. For both agents, repeatedly run each agent on a variety of randomly generated boards (at constant dimension) to estimate the number of actions (movement + examinations) each agent needs on average to find the target. You will need to collect enough data to determine which of these agents is superior. Do you notice anything about the movement/examinations distribution for each agent? Note, boards where the target is unreachable from the initial agent position should be discarded.

- For this experiment run, we choose the gridworld dimension to be 20.
- And the number of trials for each agent is chosen to be 500. This is to make sure that the effect of extreme outcomes does not affect the final results.

The following table summarizes the average number of actions taken by each agent to find the target across different terrain types.

Avg. no. of Actions	Agent 6	Agent 7
Flat	6264	1994
Hill	9803	4628
Forest	9433	13815



Observations:

- At first glance, it was observed that agent 7 took a lesser number of actions to find the target in comparison to agent 6 on average.
- But on closer inspection, it can be observed that the agent 7 performs far superior than agent 6 for the terrain types *Flat* and *Hill.* But when it comes to the Forest terrain, agent 6 finds the target in fewer steps than agent 7.

Conclusion:

- In conclusion, it can be said that agent 7 generally performs better than agent 6 in terms of the number of actions taken to find the target.
- But as the False Negative Rate (FNR) of the terrain increases (for example, *Forest* terrain), the performance of agent 7 deteriorates and agent 6 becomes more suited.
- And that makes sense. We know that the agent 7 prioritizes the cells with least probability of finding the target. And it's formula is given below.
$$P_{Terrain}(F_{x,y} = True \mid t) = [1 - FNR_{Terrain}] \,.\, P_{x,y}(t)$$
- Basically, the agent 7 will choose the cell that has maximum value for the above equation. As a consequence, the agent will often choose the cell with lesser FNR as it will give the higher value for the above equation. Because of that, the terrains with higher FNR like Forest terrain will be searched less often. Hence, if the target is in the cell with Forest terrain, the agent 7 will take a long time to find it.

5.

Describe your algorithm for agent 8, be explicit as to what decisions it is making, how, and why. How does the belief state $P_{i,j}$ (t) enter into the decision making? Do you need to calculate anything new that you didn't already have available?

- From the previous question, we saw that agent 7 significantly performed better than agent 6, except for the cases where the target was in the Forest terrain.
- Hence, our approach to building an agent 8 that's better than agent 6 and 7, is to take what we know to be a good approach so far and build on it (or improve it). From agent 7, we know that pursuing and examining the cells with the highest probability of finding the target is a good approach. At least, as long as FNR is on the lesser side.
- Hence, our new agent will still make use of each cell's probability of finding the target

- We know that we want to optimize for the metric of the number of actions the agent takes before finding the target. Two activities count as an action - 'moving' and 'searching'.
- We know the influence of 'search' is small on the final metric since we 'search' less often compared to 'moving'.
- Hence, it would be good to reduce the number of 'moving' our agent does.

- Since we don't want our agent to travel too long to search for the target, we can add distance from the agent as a factor in the metric our agent looks at when it tries to decide on the next cell to examine. That way our agent will deprioritize a cell if it has to travel longer to reach it even if our agent has a higher probability of finding the target there.

Implementation:

Let's get the notations down.

- D - Manhattan distance from the position of the agent to the cell (i, j)
- D_{Norm} - Normalized Manhattan Distance
- dim - Dimension of the grid
- α - Co-efficient / weight
- P_{calc} - Metric that our agent uses to decide which cell to examine or search

$$D_{Norm} = D / 2 * dim$$

$$P_{calc} = P\ (F_{x,y} = True)\ +\ \alpha\ *\ [1\ -\ D_{Norm}]$$

- Distance is normalized using the maximum value the distance can have - from one corner to the diagonally opposite corner which is two times the dimension of the grid.
- Since the agent will maximize the above metric P_{calc}, we subtract D_{Norm} from 1 before including it in the metric.
- Coefficient α will decide how much to penalize longer distances. For our implementation, a value of 0.0002 is used.

In conclusion, our new agent will tend to decide to search the cell with maximum probability of finding the target. But if this cell is too far from the agent, it might choose a nearer one with a high probability of finding the target. In effect, we have penalized the agent from travelling longer distances. And this will minimize the number of actions required to find the target.

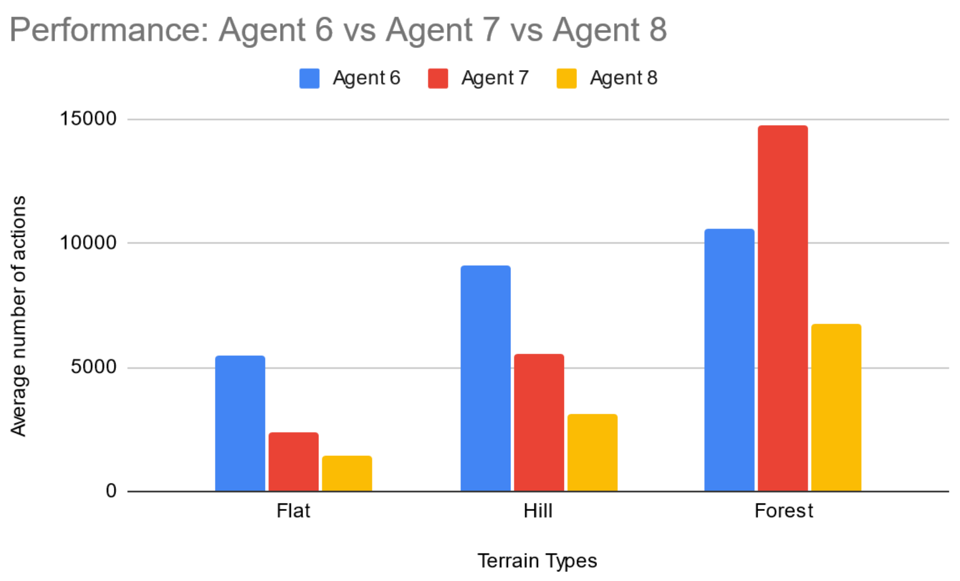
And to answer the question, belief state very much directly influences our algorithm's decision since $P\ (F_{x,y} = True)\ =\ P_{i,j}(t)\ *\ [1\ -\ FNR]$

6.

Implement Agent 8, run it sufficiently many times to give a valid comparison to Agents 6 and 7, and verify that Agent 8 is superior.

Similar to question 4, the number of trials run will be 500. And the dimension of the grid will be 20. We also use an alpha of 0.0002 in our implementation of agent 8.

Avg. Actions	Agent 6	Agent 7	Agent 8
Flat	5460	2352	1435
Hill	9083	5535	3089
Forest	10556	14750	6741



Observations:

- From the data above we can clearly see that our new agent performs significantly better than the other agents in all 3 terrains.

7.

How could you improve Agent 8 even further? Be explicit as to what you could do, how, and what you would need.

- Alpha value for the agent is chosen after manual trial and error. Hence, different alpha values can be experimented to identify which one works the best, the ideal weight with which we have to penalize distances.
- Also, keeping the idea of constraining the distance, we can come up with a better equation for the metric P_{calc}.

8.

For Agent 9, we let the target move each time step, but only to one of its immediate neighbors (uniformly at random among unblocked neighbors). The Agent can additionally sense in each cell whether the target is in one of its 8 immediate neighbors, but not which one. Build an Agent that adapts to this moving target and extra partial sensing. How does it decide where to go next, and what to do? How is the belief state updated? Implement, and generate enough data for a good comparison.

Since getting into equations can be really confusing for this question, the answer will deliberately be in plain text and will have very few equations/notations unlike previous questions.

Note: In this answer, whenever 'probability' is mentioned, it refers to the probability that the target is in the cell and not the probability of finding the target in the cell.

Belief state update for every time step:

At the end of each time step, the target moves to one of its unblocked neighbours. Hence, the agent has to account for this and update it's belief state after each time step based on what it knows about the gridworld so far.

To do that, after every time step, irrespective of what the agent has found/inferred in that time step, the following update is performed on all cells *i* of the gridworld (agent's belief state).

$$P_{t+1}(Target\ is\ in\ cell\ i) = \sum_{j\ in\ N} [\ P_t(Target\ in\ j)\ /\ Number\ of\ unblocked\ neighbours\ of\ j\ ]$$

N - set of all the neighbours of cell *i*

In other words, the probability that the target is in the cell *i* at time step *t+1* is equal to the sum of [Probability that the target is in j at time step *t* {divided by} Count of unblocked neighbours of cell *j*] over all the neighbours *j* of cell *i*.

<u>Explanation:</u>

- The probability that the target is in cell *i* at time step *t+1* is entirely dependent on its neighbours since it's a target that can move only to its neighbours.
- For example, a cell *i* can contain the target in time step (t + 1) only if the target was in one of its neighbours in the previous step.
- Hence, the new probability of cell *i* is calculated from

a. how likely is a neighbour *j* of the *i* to contain the target
b. given the target in this neighbour *j*, how likely is it to move to the cell *i* given it has n unblocked neighbours to which it can move to. Since the target moves randomly, this becomes 1/n

Belief state update after inference neighbours:

Once the agent arrives at a cell i, it senses its 8 immediate neighbours to see if the target is in one of these neighbours. There are two possible scenarios:

A. Target is in one of its neighbours
B. Target is not in one of its neighbours

<u>A. Belief state update if the target is in one of its 8 neighbours</u>

- In this case, since we know that the target is one of the 8 neighbouring cells, the probability of all the cells other than these 8 are set to 0 and these 8 are updated accordingly.
- For the latter, we make use of the probability update equation we use when we find that a cell is blocked.
- This is because knowing that a cell is blocked essentially gives us the information that it does not have the target. In this case, we just get the information from inferencing rather than from identifying the blocks.
- Hence, we just set the cell's probability to 0 and update the probability of other cells accordingly using the relation from the blocked cell scenario from 2.1.
- But in this case, we will have to do the probability update multiple times in a loop since we just found that the target is not in multiple cells.

<u>B. Belief state update if the target is not in one of its neighbours</u>

- This is very similar to the previous scenario. Except here, we get to know that the 8 neighbours don't have the target but one of the remaining cells does.
- Hence, the same probability update equation is used. But in reverse.
- Here, we use the probability update equation to update the probability of all the cells except the 8 neighbours using the probability values of the 8 neighbours rather than vice versa.

How the next target is decided?

Even though we know that the agent 8's way of deciding the next target is far superior, for the sake of simplicity, the agent 9 identifies the next target as the cell that has the highest probability of containing the target. Basically the agent 6's way

Execution flow:

1. Initially the agent’s belief state is made of empty gridworld with all the cells having equal probability of 1 / (dim * dim).
2. Agent identifies and updates its goal based on the current belief state. The goal is selected as the cell that has the highest probability of containing the target. This is always (0, 0) at the start.
3. Agent uses A* algorithm to devise a path to the set goal based on its belief state.
4. Agent starts traversing towards its goal.
5. Agent senses neighbours at every step. Based on whether a target is sensed or not, the belief state and the probabilities of all cells are updated. Check if the cell with the highest probability based on the latest belief state has changed from the agent's set goal. Then go to step 2.
6. If a block is identified, the belief state is updated. The probabilities of all cells are reevaluated. Goes to step 2
7. Once the agent’s goal and agent’s position matches, search for a target. If a target is found, exit. Else update the belief state and the probabilities and go to step 2.

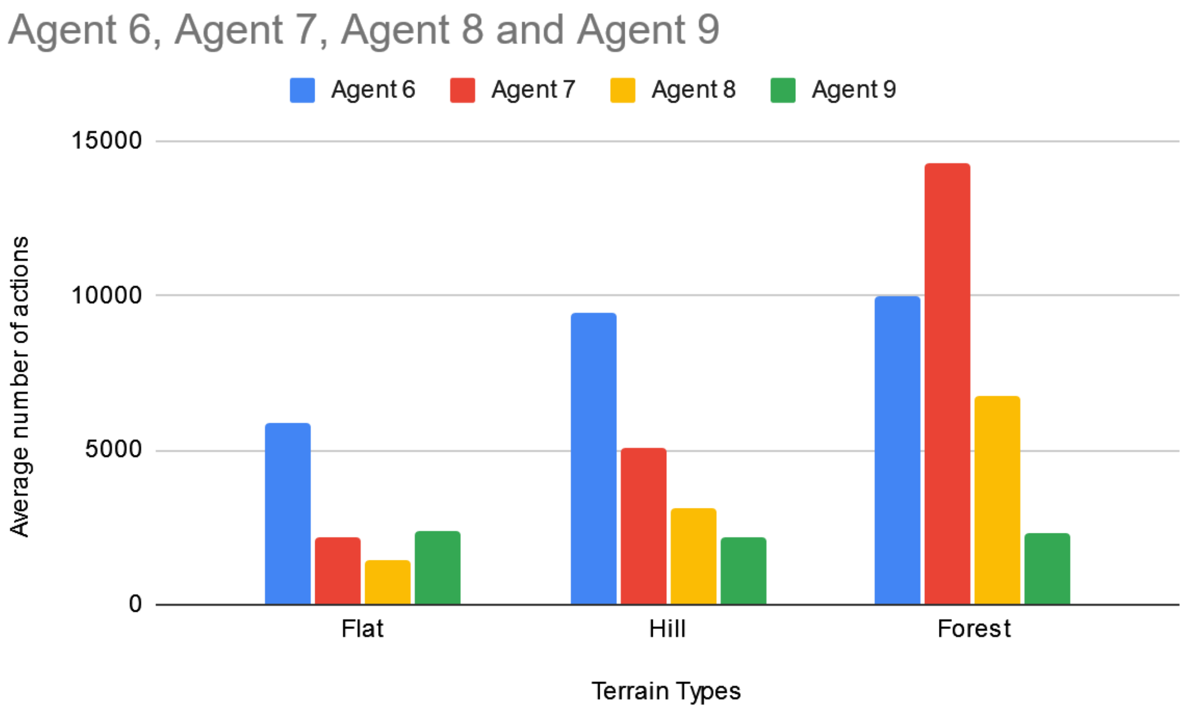
Implementation & Code:

Code for agent 9 is not available in GitHub but in this **Colab notebook** separately as I didn’t want to mess up previous agents.

Results:

As before, we run the new agent on randomly generated 500 grids and the average number of actions is calculated.

Avg. Actions	Agent 6	Agent 7	Agent 8	Agent 9
Flat	5862	2173	1435	2406
Hill	9443	5081	3089	2193
Forest	9994	14282	6741	2347



- We can see from the above plot the agent 9 performs a lot better than other 3 agents overall.
- Interestingly, agent 9's performance doesn't vary much across different terrains.