Assignment 3
Introduction to Artificial Intelligence

BY

Kartik Rattan, 187001624
Mareesh Kumar Issar, 186000297

Date: November 12, 2019

Ques1. Given observations up to time t ($Observations_t$), and a failure searching Cell j ($Observations_{t+1}$ = $Observations_t \wedge$ Failure in $Cell_j$), how can Bayes' theorem be used to efficiently update the belief state, i.e., compute:

$$P(\text{Target in } Cell_i \mid Observations_t \wedge \text{Failure in } Cell_j) .$$

Solution.

Using Bayes theorem, we have

$$\mathbf{P(A \wedge B \mid C) = P(A \wedge B \wedge C) \div P(C) = (\, P(A \mid B \wedge C) .\ P(B \mid C) .\ \cancel{P(C)}\,) \div \cancel{P(C)}}$$

Therefore, $P(A \wedge B \mid C) = P(A \mid B \wedge C) .\ P(B \mid C)$

$$\mathbf{P(\, failiure_j \mid target\ in\ cell_j\,) = False\ negative\ in\ cell_j = P(j)}$$
$$\mathbf{P(target\ in\ cell_i \mid obs_{t+1}) = F(i,t+1)}$$

$$\mathbf{P(target\ in\ cell_i \mid obs_{t+1}) = P(target\ in\ cell_i \mid obs_t \wedge failiure_j)}$$
$$\mathbf{= P(target\ in\ cell_i \wedge obs_t \wedge failiure_j) \div P(\ obs_t \wedge failiure_j\ )}$$
$$\mathbf{= \cancel{P(\ obs_t)}\ .\ P(target\ in\ cell_i \mid obs_t)\ .\ P(failiure_j \mid target\ in\ cell_i \wedge obs_t) \div (\ \cancel{P(\ obs_t)}\ .\ P(\ failiure_j \mid obs_t)\ )}$$
$$\mathbf{= P(target\ in\ cell_i \mid obs_t)\ .\ P(failiure_j \mid target\ in\ cell_i \wedge obs_t) \div P(\ failiure_j \mid obs_t)}$$

$$\mathbf{Now,\ P(\ failiure_j \mid obs_t) = \sum_i P(\ failiure_j \wedge target\ in\ cell_i \mid obs_t)}$$
$$\mathbf{= \sum_i P(\ failiure_j \mid target\ in\ cell_i \wedge obs_t).\ P\ (\ target\ in\ cell_i \mid obs_t)}$$
$$\mathbf{= P(\ failiure_j \mid target\ in\ cell_j\ )\ .\ P(\ target\ in\ cell_j \mid obs_t) + \sum_{i \neq j} P(\ failiure_j \mid target\ in\ cell_i \wedge obs_t).\ P\ (\ target\ in\ cell_i \mid obs_t)}$$
$$\mathbf{= P(j)\ .\ F(j,t) + (\ 1 - F(j,t)\ )}$$

$$\mathbf{Hence,\ P(target\ in\ cell_i \mid obs_{t+1}) = F(i,t+1)}$$
$$\mathbf{P(target\ in\ cell_i \mid obs_{t+1}) = P(target\ in\ cell_i \mid obs_t \wedge failiure_j)}$$

$$\mathbf{= P(j)\ .\ F(j,t) \div (\ P(j)\ .\ F(j,t) + (\ 1 - F(j,t)\ )\ )} \qquad \mathbf{if\ (\ i == j\ )}$$
$$\mathbf{= F(i,t) \div (\ P(j)\ .\ F(j,t) + (\ 1 - F(j,t)\ )\ )} \qquad \mathbf{if\ (\ i \neq j\ )}$$

Ques 2. Given the observations up to time t, the belief state captures the current probability the target is in a given cell. What is the probability that the target will be found in $Cell_i$ if it is searched?
$P(\text{Target found in } Cell_i \mid Observations_t)$?

Solution.

From the previous question we know that, $P(A \wedge B \mid C) = P(A \mid B \wedge C) .\ P(B \mid C)$

$$P(\text{Target found in } Cell_i \mid Obs_t) = P(\text{Target found in } Cell_i \wedge \text{target in } cell_i \mid Obs_t) + P(\text{Target found in } Cell_i \wedge \sim \text{target in } cell_i \mid Obs_t)$$
$$= P(\text{Target found in } Cell_i \wedge \text{target in } cell_i \wedge Obs_t) \div (\ P(\ obs_t\ )\ )$$
$$= \cancel{P(\ obs_t)}\ .\ P(target\ in\ cell_i \mid obs_t)\ .\ (P(\text{Target found in } Cell_i \mid \text{target in } cell_i \wedge Obs_t) \div \cancel{P(\ obs_t)}$$
$$= F(i,t)\ .\ P(\text{Target found in } Cell_i \mid \text{target in } cell_i \wedge Obs_t)$$

Where, $F(i,t) = P(\ target\ in\ cell_i \mid obs_t)$ calculated in the previous question and

$$P(\text{Target found in } Cell_i \mid \text{target in } cell_i \wedge Obs_t) = P(\text{Target found in } Cell_i \mid \text{target in } cell_i)$$
$$= 1 - \text{false negative for } Cell_i$$
$$\approx [0.9,\ 0.7,\ 0.3,\ 0.1] \text{ for flat, hilly, forested, maze of caves}$$

Ques3. Consider comparing the following two decision rules:

– Rule 1: At any time, search the cell with the highest probability of containing the target.

– Rule 2: At any time, search the cell with the highest probability of finding the target.

For either rule, in the case of ties between cells, consider breaking ties arbitrarily. How can these rules be interpreted / implemented in terms of the known probabilities and belief states?

For a fixed map, consider repeatedly using each rule to locate the target (replacing the target at a new, uniformly chosen location each time it is discovered). On average, which performs better (i.e., requires less searches), Rule 1 or Rule 2? Why do you think that is? Does that hold across multiple maps?

Solution. For a fixed map, we searched the target with rule 1 and rule 2 to search 100 times, among which, the target falls

The average number of steps for searching the target with Rule 1 = **7263.37**
The average number of steps for searching the target with Rule 2 = **7024.66**

We observe that Rule 2 has a better performance then rule 1 when both the rules are applied to a single map. Rule 2 considers in addition to the probability where the target is, the probability that the target can be found given that target is present there.

$$P(\text{Target found in Cell}_i | \text{Obs}_t) = F(i, t) \cdot P(\text{Target found in Cell}_i \mid \text{target in cell}_i \wedge \text{Obs}_t)$$

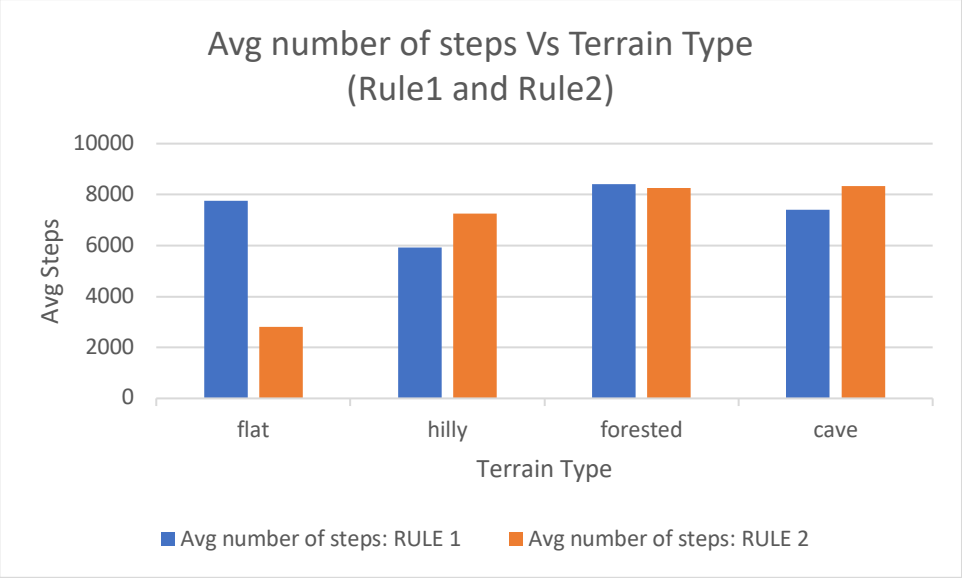
RULE 2 = RULE 1 . (1 - False Negative)

```
Average number of steps used for Rule 1 100 iterations:7263.37
Average number of steps used for Rule 2 100 iterations:7024.66
RULE 1 Analysis:
0 : 17 Avg Steps: 7766
1 : 33 Avg Steps: 5916
2 : 29 Avg Steps: 8402
3 : 21 Avg Steps: 7399
RULE 2 Analysis:
0 : 17 Avg Steps: 2801
1 : 33 Avg Steps: 7265.33
2 : 29 Avg Steps: 8269.79
3 : 21 Avg Steps: 8346.14
```

Where 0: flat, 1: hilly, 2:forest, and 3:cave.
Where,
0 : 17 Avg Steps: 7766 states: The target was found in flat areas 17 times for which the average steps is 7766.33

We can see from the table that rule 2 has a better performance in terrain type flat, hilly and forested and performs poorly in when the target is present in caves. We believe that rule 1 performs better than rule 2 when the target is in a cave is that the conditional probability to find a target in Caves when the target is indeed in the cave is too small (0.1). Therefore, if the target is in a cave, for rule 2, the probability to find target in that cell is still too small compared with other cells, and thus the cell is not selected.

Rule 2 gives a better estimate for finding the targets in different map. This is because it additionally takes into account P (Target found in Cell $_i$ | target in cell$_i$) = 1 - false negative for Cell$_i$. For flat areas, the probability is 0.9, which enhances the belief that the target is in flat areas and hence least number of steps.



Ques 4. Consider modifying the problem in the following way: at any time, you may only search the cell at your current location, or move to a neighboring cell (up/down, left/right). Search or motion each constitute a single `action'. In this case, the `best' cell to search by the previous rules may be out of reach and require travel. One possibility is to simply move to the cell indicated by the previous rules and search it, but this may incur a large cost in terms of required travel. How can you use the belief state and your current location to determine whether to search or move (and where to move), and minimize the total number of actions required? Derive a decision rule based on the current belief state and current location, and compare its performance to the rule of simply always traveling to the next cell indicated by Rule 1 or Rule 2. How does Utility apply here? Discuss.

Solution. In the previous questions, we used Rule 1 and Rule 2 to find the target (by searching the cell that has the maximum probability (utility) of the target):

Rule 1: P (Target in Cell$_i$ |Observations$_t$ ∧ Failure in Cell$_j$)

Rule 2: P (Target found in Cell$_i$|Observations$_t$)

For this question, we know that after opening each cell and incurring a failure, the agent is faced with a choice of whether to search the current cell again or move to a neighbor cell each of which constitute an action. So, here we propose the following rule for finding the target.

Rule_distance: P (Target in Cell$_i$ |Observations$_t$ ∧ Failure in Cell$_j$) * P(1÷ (manhattan distance +1))

Using the above rule we want to minimize the total number of actions. In order to travel towards a cell suggested by either Rule 1 or Rule 2, the agent performs actions/moves equal to the manhattan distance between the current cell and the suggested cell. So, in order to minimize the total number of actions taken by the agent, we have to focus on reducing the cost of moving between cells, or we have to create a metric that takes into account the distance of the suggested cell from the current cell. One such metric is the Manhattan distance between the cells.

Initially, we update the belief of each cell (after a failure) using rule 1. After this, we scale all the belief's by a scaling factor **(1÷(manhattan_distance(current cell where failure occured, suggested cell)+1)**, this step is analogous to applying a filter on all our observations. After scaling the beliefs we move to the cell with maximum belief of having the target and search it.

Consider the following example where the beliefs of the cells in the path from current cell to the suggested cell is given as:

Value of beliefs before scaling: [0.2,0.35,0.1,0.4]

Value of beliefs after scaling: [0.1, 0.1167, 0.025, 0.08]

Instead of going to the suggested cell, the intermediate cell with prob 0.35 is also worth searching (and in turn might reduce the number of actions taken by the agent). This functionality is not present in Rule 1, which is captured by Rule_distance.

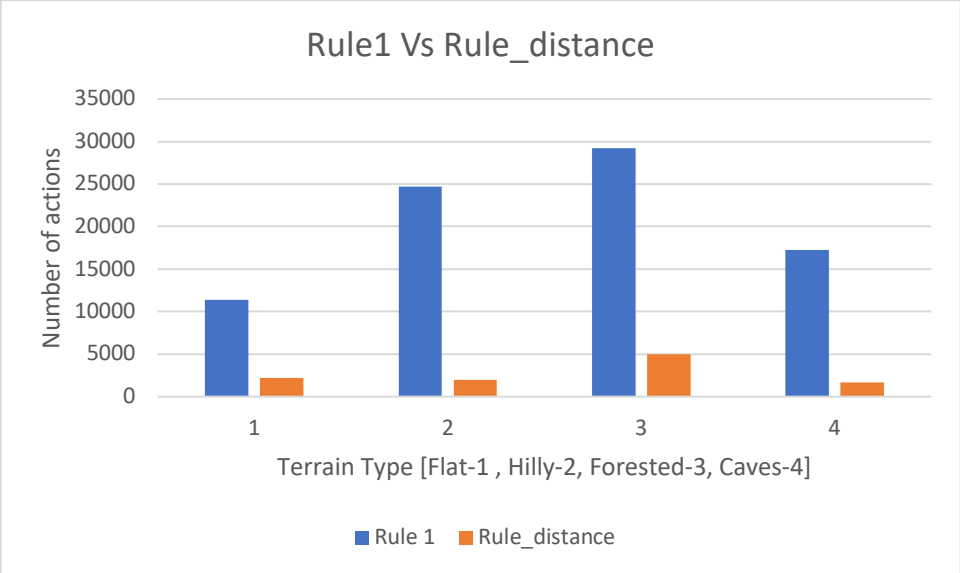
Also, after comparing the performance of Rule_distance with Rule 1 we find that Rule_distance moves to a cell only once, and moves to the next better cell when it is not able to find the target. Whereas if we use Rule 1, the agent can return back to a previously visited cell and thus takes more number of actions to find the target as compared to Rule_distance.

Comparison of performance of Rule 1 and Rule_distance (for 100 iterations):

Average number of actions using Rule 1- 82533.2
Average number of actions using Rule_distance- 10900.7

Terrain Type T	Number of times target in T	Avg steps to locate T using Rule 1	Avg steps to locate T using Rule_distance
FLAT	23	11363.8	2229.15
HILLY	21	24680	1982.97
FOREST	39	29212.6	5003.18
CAVES	17	17276.7	1685.36

Thus, Rule_distance performs better as compared to Rule 1.



Ques 5. An old joke goes something like the following:

A policeman sees a drunk man searching for something under a streetlight and asks what the drunk has lost. He says he lost his keys and they both look under the streetlight together. After a few minutes the policeman asks if he is sure he lost them here, and the drunk replies, no, and that he lost them in the park. The policeman asks why he is searching here, and the drunk replies, "the light is better here".

In light of the results of this project, discuss.

Solution. Making an analogy from the above-mentioned joke we have

The drunk man is searching a place where the light is better, or this refers to the cell with the highest probability of revealing the target or the terrain with the least false negative. In Rule 2, the prob of finding a target is also enhanced with a factor of (1 – false-negative). Thus, this is also observed in our analysis as the number of steps to find the target in flat terrain is always less than any other terrain. Also, the number of steps required to search for the target in flat terrain using rule 2 is much less than rule1.

$P(\text{Target found in Cell}_i \mid \text{target in cell}_i \wedge \text{Obs}_t)$ for each terrain [Flat, Hill, Forest, Cave] is [0.9, 0.7, 0.3, 0.1] respectively. Now the drunk man (or our algorithm) is inclined towards the flat terrain more because a $P(\text{Target found in Cell}_i \mid \text{target in cell}_i \wedge \text{Obs}_t)$ is 0.9 or the light is the brightest in flat terrain.

Ques6 (BONUS). In this section, the target is no longer stationary, and can move between neighboring cells. Each time you perform a search, if you fail to find the target, the target will move to a neighboring cell (with uniform probability for each). However! All is not lost; your target has a tracker that sends you information about their position. The tracker is meant to send back the type of terrain the target is located; however, it is broken, and each time step it reports a terrain type where the target is not. (i.e., if the tracker reports 'hilly', you know the target is located in a non-hilly location.)

Implement this functionality in your code. How can you update your search to make use of this extra information? How does your belief state change with these additional observations? Update your search accordingly.

Re-do question 4) above in this new environment with the moving target and extra information.

Solution.

We implement the above information using a separate transition model for target position.

State i: Target in i | $obs_{t+1} = X_{i,t+1}$

Let Y_i be the observation, or the information provided by the tracker at time i.

$$\text{Let } P(X_{i,t+1} = x \mid X_0, X_1, X_2, \ldots\ldots, Y_t, X_{i,t}) = P(X_{i,t+1} \mid X_{i,t})$$

When we marginalize for $X_{i,t}$, for all values of x', where x' = all states that the target can be in the obs_t or all states other than the one provided by the tracker. The tracker provides the information that the target would not be in a particular state x'. Thus, x' = all the states other than the one stated by the tracker.

If a cell is surrounded by the terrain returned by the tracker, the target can never be in that cell in t+1 since the target moves to adjoining cells only. For instance, the value returned by tracker at time t is hilly. If a cell is surrounded by hilly terrain, the target will never be in cell i.

hilly	hilly	hilly
hilly	Cell i	hilly
hilly	hilly	hilly

So, for each of the cell, there should be an increase in the probability of the cell with respect to the number of neighbors not equal to tracker value. Or we can provide a disadvantage to all those cells with the terrain type returned by the tracker.

This can be achieved as follows:

While we search for the next cell, we do not search for the cells with terrains returned by the tracker value. Thus, this reduces our search space. We implement this as follows:

When we query the algorithm to get the next cell for search, we select a cell with the maximum belief. If the cell with maximum belief has a terrain returned by the tracker, we ignore it. Thus, we search only those cells with a terrain not returned by the tracker.

We observe that when we run the algorithm in question 4 for the updated environment, the number of steps to reach the target decreases.