CS440

Search & Destroy

By Maitri Shah, Kajol Bhat & Ashni Patel

INTRO TO AI SPRING 2021

Contents

Part One

1. Questions and Write Up

1.1 Problem 1

1) Calculating:

$$P(Target\ in\ Cell\ i |\ Observations\ t\ \wedge\ Failure\ in\ Cell\ j) \tag{1.1}$$

Solution:
The Bayes theorem states that:

$$P(H|D) = \frac{P(D|H)P(H))}{P(D)} \tag{1.2}$$

Assuming that the target is in cell i, we can express our belief state as:

$$B(i) = P(target\ is\ in\ i) \tag{1.3}$$

where B(i) is the probability of target being in cell i.
We can then draw the inference that the remaining cells j, are cells where the target doesn't exist and that they would return fail to return the target when searched.
Giving us the Probability:

$$P(target\ is\ in\ i\ |\ Fail\ searching\ j) \tag{1.4}$$

Re-writing in terms of Bayes Theorem:

$$\frac{P(Fail\ searching\ j|Target\ is\ in\ i) * P(Target\ is\ in\ i)}{P(Fail\ searching\ j)} \tag{1.5}$$

We can replace P(target is in i) with our belief state, as follows:

$$\frac{P(Fail\ searching\ j|Target\ is\ in\ i) * B(i)}{P(Fail\ searching\ j)} \tag{1.6}$$

We can infer the following for:

P(Fail searching j | Target is in i)

If i != j, i.e if every cell in j is a failure cell despite the cell's false negative rates then, P(Fail searching j | Target is in i) = 1 where no cell in j is mistaken for the target cell and i is indeed where the target is.
If (i ==j) then P(Fail searching j | Target is in i) would be the false negative rate of i. Since we assumed the target cell was a failure cell instead.
We can infer the following for:

P(Fail searching j)

Re-writing the above equation in terms of j, being in belief(j) and not being in belief(j)
we get:

P(Fail j $\wedge$ in j) + P(Fail j $\wedge$ not in j)

Simplifying further:

P (in j) * P(Fail j | in j) + P(not in j) * P(Fail j | in j)

In terms of belief(j)

belief(j) * (False negative rates of j) + (1 - belief(j)) * 1

With the above information we can re-write equation 6, as follow which updates our new belief(i) state:

$$\frac{P(Fail\ searching\ j|Target\ is\ in\ i) * B(i)}{belief(j) * (False\ negative\ rates\ of\ j) + (1 - belief(j)) * 1} \tag{1.7}$$

1.2 Problem 2

Calculating:

$$P(Target\ found\ in\ Cell\ i \mid Observations) \tag{1.8}$$

Solution:
The Bayes theorem states that:

$$P(H|D) = \frac{P(D|H)P(H))}{P(D)} \tag{1.9}$$

Using Bayes' theorem, we can simplify this.

P(D|H) in this situation would be the probability that the observations can be used to find the target. This can be rewritten as:

$$P(Target\ is\ in\ Cell \mid Observations) \tag{1.10}$$

P(H) in this situation would be P(Target found). This means that the target would have to be both in the cell and actually found despite the false negative rate. Therefore this can be written as :

$$P(Target\ in\ Cell\ AND\ Target\ found\ in\ Cell) \tag{1.11}$$

P(D) would be the total probability of the target being found based off of the observations. This would mean the probability that the target would be in the cell and found, plus the probability that the target is in the cell and not found.

$$P(Target\ in\ Cell\ AND\ Target\ found\ in\ Cell) + P(Target\ in\ Cell\ AND\ Target\ NOT\ found\ in\ Cell) \tag{1.12}$$

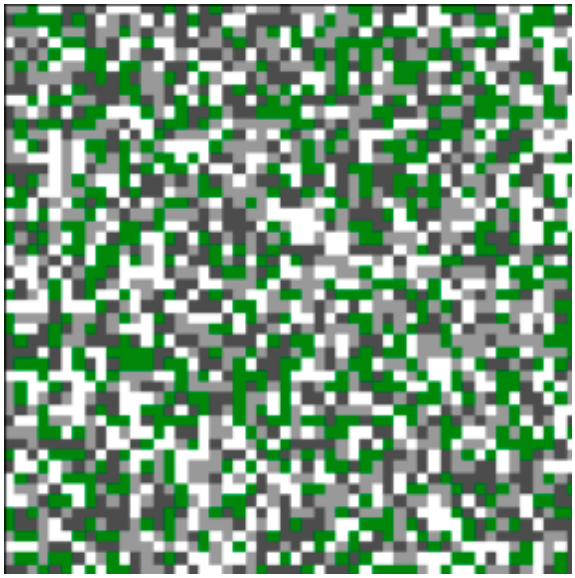
Suppose the probability of the value being in the cell, α and it's false negative rate is .9. Then $P(B) = .9 * \alpha + (1 - .9)(\alpha) = 1$.
Thus, in every situation P(D) would be equal to 1.
Therefore, the equation for using Bayes Theorem would be as follows :

$$P(H|D) = P(Target\ in\ Cell \mid Observations) * P(Target\ in\ Cell\ AND\ Target\ Found) \tag{1.13}$$

The probability that the target will be found in Cell i if it is searched, can be calculated using the equation above.

1.3 Environment

Using a 50x50 grid, we randomly assigned a terrain type to each cell and distributed these terrain types with an equal probability of 0.25. This was done using a numpy matrix, matplotlib, and a colormap library we created to assign each terrain its respective color.



1.4 Basic Agent 1

In basic agent 1, the decision to search the cell's highest probability of containing the target is made based on the agent's understanding of the probability of each cell after each iteration. Updating the probability matrix is essentially estimating the likelihood of a target in a new cell based on the previous cell's failures and previous observations. The agent assigns equal probabilities to each cell in its initial state, when no cells have been explored or searched by the agent. Since there are no observations or failures at this stage (as seen in the probabilistic equation in 1.1), the agent will being its search by choosing a random cell.

The agent will continue to search for the goal until it is found. The agent spends the bulk of her time searching for the target. The algorithm will check the neighbors of the current cell it is at and go to the cell with the highest probability of containing the target. Any ties in probability between cells are broken based on the lowest Manhattan distance and is implemented in the code using a lambda expression. After each move, assuming that the target is not found, the probability of the entire matrix is normalized and recalculated based on the previous observations and failures. If the goal is found, the algorithm will come to a halt and return the sum of the total number of cells and total distance traversed. The total distance traveled is determined by summing the Manhattan distances of each move taken by the agent. In the instance that the target is not present in a given cell, the agent will update its probability matrix accordingly. The agent would lower the probability of the current cell having the target by incorporating the false-negative rate of the terrain. After each move the agent makes, the agent must normalize all the cells to ensure that the sum of all cell probabilities will be 1.

False Negatives

Once again, in order to factor the false negative rate into the probability matrix, the agent will reduce the probability by some variation of the false negative rate. In doing so, the agent will prioritize flat lands over caves because caves have a higher change of producing false negatives. In terrains where it is difficult to find the target, our algorithm often needs to check the same spot because it misses the target the first time. For example, caves have a 90% chance of false negatives. Once a target is found, a random number from 0 to 1 is chosen. If the number random number chosen is lower than the false negative rate, the agent has not found the target. In the event that the random number chosen is higher than the false negative rate, the agent has found the target successfully. The agent finding a target successfully is much more likely in flat lands than a cave because in flat lands, there is only a 10% chance of false negatives.

1.5 Basic Agent 2

In basic agent 2, the agent will iteratively travel to the cell with the highest probability of finding the target within that cell and search that cell. The algorithm will repeat this process until the target is found. Similar to the logic that we used in basic agent 1, we used our probability matrix to implement this and decided which cell had the highest probability of being the target. However, in addition to this, we immediately also used the complement of the false-negative rate ($1 - rate_{FalseNegative}$) after this to accurately guess where the target is likely to be and be actually found. Generally, agent 2 is more efficient than agent 1 as it can find the target by visiting a smaller number of cells and using a lower overall Manhattan distance. However, sometimes basic agent 2 would result in a higher overall score with the cave or forest terrains. This is can be attributed to the fact that the low false-negative rate made agent 2 less confident that the target would actually be found when going to the target location and thus would travel to other cells first and ultimately increase the score. Since there is a lower chance of finding the target in forest and cave regions, Basic Agent 2 tends to do better in Flat and Hilly regions because these regions have a lower false

negative rate. The algorithm will search the flat and hilly cells first because of that. This can be seen in the graph below as due to one exceptionally high value, the forest score for agent 2 is higher than that of agent 1 as its false-negative rate is .7. Still, in all other situations, agent 1 is generally worse than agent 2.

We have graphed for basic agent 1 and basic agent 2 the average performance, where each run results in the total distance + number of searches. We ran the data with 50x50 maps until we were able to get about 10 maps for each terrain type. On average, agent 2 performed better, and when we break this down by terrain type we can see that this is especially the case for the hilly and flat terrains.

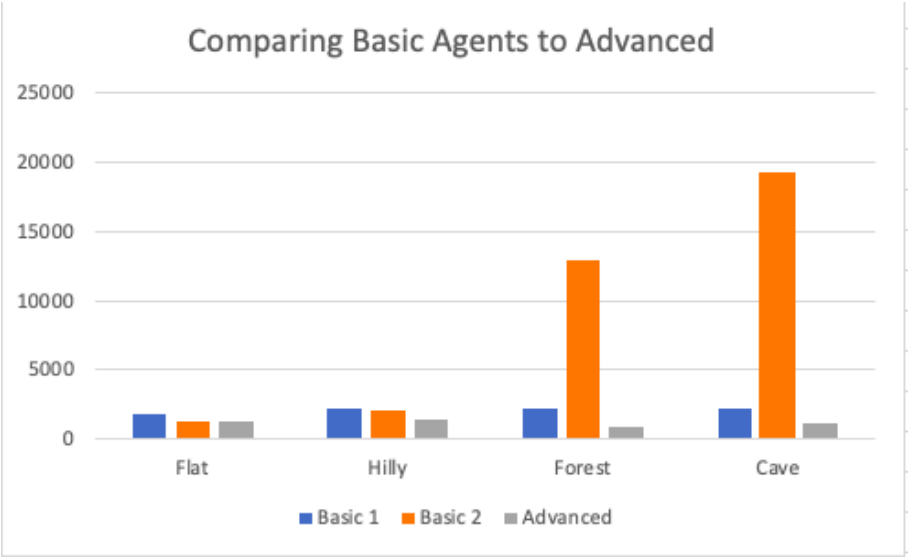
1.6 Advanced Agent

In order to improve our basic agents, we created an advanced agent. Generally, our basic agent 2 performed better than our basic agent 1, but there were some situations where this was not the case. Basic agent 2 would be worse or perform the same as agent 1 in a few situations with forest or cave terrains as these two terrains have very high false negative rates, and thus the agent 2 is less confident that the target will actually be found in those locations. Agent 2 will prioritize hilly and flat regions because those have a better chance of the target being found. After all, when the agent gets to forest or cave regions, it will likely not find the target due to false negatives. Thus, Basic Agent 2 will not go to cave or forest regions as soon as Basic Agent 1 would. To move around this, we decided to implement an agent that would go to the cave and forest terrains twice to improve the likelihood of the target being found at the target location. Additionally, to generally improve the efficiency of the code we decided to implement the function so that the neighbors of the neighbors of the initial cell would be checked for probability and Manhattan distance, rather than just the immediate neighbors like with the basic agent. This allowed us to have a larger number of cells to look at when deciding which cells would be optimal, and this combined with the multiple searches made the advanced agent much more efficient, as we can see with the chart below.

Due to our computer's capacity, for the advanced agent we tested using a $30x30$ map and compared it to basic agents 1 and 2 results that were also with a $30x30$ map. Our computer was having a hard time handling the $50x50$ dimension so we decided to lower the dimensions only for this problem to be able to get enough results to draw conclusions after receiving permission from Professor Cowan.

Given enough space and time, to make our agent even better we would run the repeatedly searched cells for forest and cave even more times, possibly 10, and then mark these off to not

repeat further so that we do not unnecessarily check the same cell too many times. In addition, if possible to do so check the neighbors of multiple cells at once and go to the most ideal probability and Manhattan distance based off of that.



2. Bonuses

2.1 LaTeXed Report

This entire report was written using LaTeX

2.2 Clever Acronym

PORSCHE - Probability of Reaching Specific Cells Here Efficiently (Advanced Agent)
TICUP - Target is Contained Using Probability (Basic Agent 1)
PROF - Probability Realizing Observable Failures (Basic Agent 2)

2.3 Moving Target

In order to try to solve the question with the addition of a moving target, after changing our code implement this addition, we also tried to factor it into our logic system. Since we were told whether the new location of the target was within a 5 value Manhattan distance of our agent, we created a loop in which we appended every value within this circular radius onto a list. Next, since we knew the target was one of those values, we changed the selection process of the next cell to query. Originally, we looked at the entire map and decided which cell had the best likelihood of being the target, but now, we just decided this based off of the values in the new list of values that matched the Manhattan distance criteria. Furthermore, in the instance that the target is not within a distance of 5 of the agent, we just run the code normally based on the probability. We additionally have a counter sot that if the agent is a distance of 5 away from the target more than 10 times, we exit the program as it is unlikely to catch up with it at this point. This same logic was used and implemented into the basic agents as well as the advanced agent.

Ultimately, we were unsuccessful when we tried to implement this for our basic agent when we tried to implement it with maps larger than 10x10, but we were successful when we added this to our advanced agent. However, we were not able to make too many informed decisions based off of this as implementing this was extremely time consuming as our computers could not handle the time complexity of the algorithm. The result of the algorithm for our advanced agent did relatively well and was able to find targets in a under 100 performance score. Based on our understanding, if

we were able to have more computational power, we believe that the result of this would be that this would ultimately end up being able to find most targets in a smaller overall score. This is as typically, we have no real idea about where the target is location-wise and we are depending mostly on probabilities, but with this since we know that it's in a radius of 5, we can use our logic in a more contained amount of space and thus likely find the result more quickly and less computationally.

2.4 Division of Work

All of the work was done over Zoom calls as a group so all of the work was split evenly (Acknowledged by Maitri Shah, Kajol Bhat, and Ashni Patel)

2.5 Honor Statements

Maitri Shah (ms2804): On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.

Kajol Bhat (kkb56): On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.

Ashni Patel (agp96): On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of the code written is my own.