# Assignment 3
# Search & Destroy

**Contributors:**
Bhavya Tiwari - bt290
Himanshu Jain - hj288
Nidhi Arun Harwani - nh417
Shishir Umesh - su90

## Part 1

Q1) Given observations up to time t Observations$_t$, and a failure searching Cell j, how can Bayes' theorem be used to efficiently update the belief state, i.e., compute: *P( Target in Cell$_i$ | Observations$_t$ ⋀ Failure in Cell$_j$ )* .

We are given that there are 4 types of terrains and the probability of a false negative to occur in each terrain is also given:

| | | |
|---|---|---|
| *P( Failure in Cell[j] \| Target in Cell [j] ) =* | *0.1* | *if Cell[j] is flat* |
| | *0.3* | *if Cell[j] is hilly* |
| | *0.7* | *if Cell[j] is forest* |
| | *0.9* | *if Cell[j] is cave* |

Initially, we have no clue as to where the target is. So, at this period of time, the possibility of each cell having the target is the same i.e. 1/(number of cells).
Therefore, initial *Belief[i] = 1/2500* (for a 50x50 grid)

If there is a target in searched cell[j] , the program returns success and we stop, otherwise we need to continue our search by first updating the Belief[] array which is done by finding

*P(target in Cell[i] | Observations till now ^ Failure in Cell[j])* for every cell 'i' in the 2D array when we search any cell j which is equal to:
*P(target in Cell[i] | Observations till now ^ Failure in Cell[j])*
*= P( Observations till now) P(target in Cell[i] | Observations till now)*
** P(Failure in Cell[j]/target in Cell[i],Observations till now | P( Observations till now ^ Failure in Cell[j])* **(1)**

Since the past observations have no bearing on the reported failure probability of a cell, we observe that :

*P(Failure in Cell[j]/target in Cell[i], Observations till now)*
*= P(Failure in Cell[j]/target in Cell[i]))*

And **(1)** reduces to :
*P(target in Cell[i] | Observations till now ^ Failure in Cell[j])*
**=** *P( Observations till now) P(target in Cell[i] | Observations till now) **
*( P(Failure in Cell[j]/target in Cell[i]) | P( Observations till now ^ Failure in Cell[j])*

$P(\,Target\ in\ cell[i]/\,Past\ Observations\,)$ is nothing but *Belief[i]* ,

We also know that:
*P(Failure in Cell[j]/target in Cell[i])*
*= { 0.1 or 0.3 or 0.7 or 0.9 if i=j ,*
*1 otherwise }*

There are two terms which are unknown but as they are constants we can rewrite **(1)** with a constant k as :
*P(target in Cell[i] | Observations till now ^ Failure in Cell[j])*
*= K * Belief[i] * P(Failure in Cell[j] | target in Cell[i])*

Note that:
*Σ P(Target in Cell[i] | Failure in Cell[j] ^ Observations till now) = 1*

We can use this to find the Normalized values as :
*P(target in Cell[i] | Observations till now ^ Failure in Cell[j])*
*= Belief [i] * P(Failure in Cell[j] | target in Cell[i])*
*/ Σ(Belief [i] * P(Failure in Cell[j] | target in cell[i]))*

Q2) 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 (Target found in Cell i | Observations)*?

We know that according to the type of terrain of the cell, there is a possibility that while searching for the target, we may not find the target even though the target is present in that cell. This is a false negative case. While calculating the probability of finding the target in a cell if the target is present in that cell, we need to consider this false negative possibility.

The probability that the target will be found in Cell[i] if it is searched can be determined as follows :

*P( Target found in cell[i] | Observations )*

*= P(Target found in cell[i] ^ Target in cell[i])*

*= P( Target found in cell[i] | Target in cell[i])* P(Target in cell[i] | Observations)*

*= ( 1 - P( Failure in Cell[i] | Target in Cell [i] ) )* P(Target in cell[i] | Observations)*

We know that,

*P(Target found in cell[i] | Target in Cell[i] ) + P(Failure in Cell[i] | Target in Cell[i] ) = 1*

This has been used in the above derivation as we have been given the values of *P(Failure in Cell[i] | Target in Cell[i] )*

3) 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?

According to rule 1, we select the cell that has the highest probability of containing the target every time we need to search a new cell. This means that we need to calculate the probability of each cell to contain the target and then we select the one with the highest probability. Here, we do not have to concern ourselves if we actually find the target in that cell.

i<sup>th</sup> Cell is chosen to be searched next using the following rule:

*Max ( Belief[i]$_t$) where Belief[i]$_t$*

*= P(target in cell[i] | Observations till now ^ Failure in cell[j])*

*= Belief [i]$_{t-1}$ P(Failure in cell[j] | target in cell[i]) / Σ (Belief$_{t-1}$ [i] P(Failure in Cell[j] | target in cell[i]))*

Rule 2 says that we need to select the cell with the highest probability of finding the target every time we select a new cell to search for the target. The probability of finding the target can be found as shown in question 2. Thus, we update the Belief array for each cell as shown in the equation below and then select the cell which has the highest probability:

*P( Target found in cell[i] | Observations )*

*= ( 1 - P( Failure in Cell[i] | Target in Cell [i] ) ) * P(Target in cell[i] | Observations)*

**Performance:** Experiments were performed on 1000 landscapes of 50x50 size with approximately equal number of flat, hilly, forested, cave terrains. From the experiments, two sets of data was recorded:

i) Average number of searches for Rule 1 and Rule 2. These tests clearly prove that Rule 2, which used an average of 4486 searches to find the target, is superior to Rule 1, which used 6665 searches in comparison (This data is also tabulated in table 1). The general increase in performance can be attributed to the fact that rule 2 makes more informed decisions using extra evidence, which is the probability of encountering false negatives.

ii) Average number of searches for Rule 1 and Rule 2 for each terrain type. The results observed here show something interesting. Although rule 2 is superior in a general case. In the case where the target is located in a cave, i.e. with a very high probability of encountering a false positive, rule 2 proves inferior. This may be attributed to the fact that an algorithm that uses rule 2 does not search caves since they have a low *P(target found in cave).* The results are depicted in Fig. 2.

The answer to the question “Does the superiority of one rule over the other hold across multiple maps” is, therefore, “No”. While rule 2 is generally superior, it is inferior in the cases where the target is located in a cave.

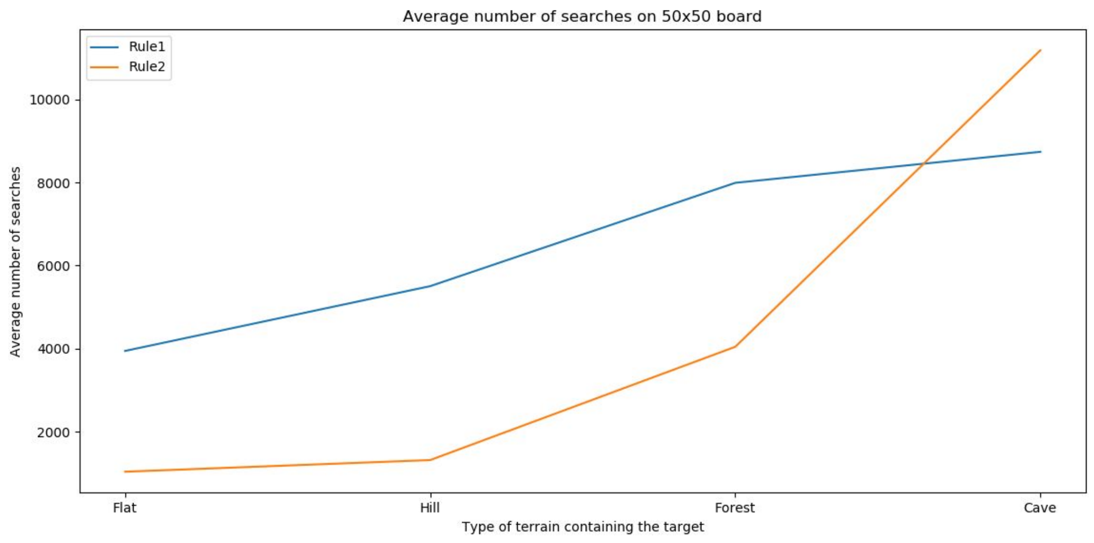

Fig 1. Average number of searches depending on the type of the terrain on which the target is located.

Q4) 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. Discuss.

We decided to go with a greedy approach where the next cell to be searched is the one with the highest value of the measure $P(Target\ found\ in\ Cell_{ij} \mid Observations_t) / d$ where d is the Manhattan distance from the current cell to the $cell_{i,j}$. Taking such a measure, while not optimal, finds a balance between the current belief state and the total cost of travel. This can be observed in the way that according to our measure, the higher the probability of finding the target in the cell and the closer it is, the more likely it is that it will be searched next. And, the farther it is from the current location and the lower the probability of it containing the target, the less likely it is that it will be searched next. The decision rule g(x) which states the action to be taken when at cell x can, then, be written as:

$$g(x) = \begin{cases} \text{search } x, & \text{if } x \text{ is the cell with the highest } val_x \\ \text{move to neighbor closest to } cell_x \text{ with highest } val_x, & \text{otherwise} \end{cases}$$

where $val_x = P(target\ found\ in\ cell_x \mid Observations_t) / d$

**Performance:** Tests were performed on 1000 50x50 landscapes. The observed results indicated very poor performance of the devised procedure, especially when compared to Rule 1 and Rule 2. The average number of actions ( no. of moves + no. of searches ) was 11,183. The comparatively poor performance is expected since the search is far more restricted.

| Rule 1<br>(No. of searches) | Rule 2<br>(No. of searches) | Restricted search (No. of moves + No, of searches) |
|---|---|---|
| 6665.21 | 4486.67 | 11183.44 |

Table 1. Average no. of actions required to find a stationary target

Q5) 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.

We have observed that if our program decides to search a cell based on the 'ease of finding' the target in that particular terrain, it performs better only for situations where the target is actually in one of the easier terrains . For instance, if the target is in a hilly terrain then Rule 2 does better at finding it on average. However, if the target is in the cave, Rule 1 now performs better. This is because Rule 2 will spend more time searching in the easier terrain even though the target is actually not there. The drunk man is essentially following Rule 2 by looking for his keys in the lighted portion where it would be easier to locate them. However, the target i.e. keys are actually in the tougher terrain ( dark area ) and so he has adopted a bad strategy.

# Part 2

In this case, we start we an initial belief of 1/n^2 where n is the dimension of the landscape. We start the search for the target at random. At any point during the search, a new evidence is received of the Type1 X Type2. We update the belief state in the following way:

i) Multiply the belief of the current cell c by a factor of *P(target not found in c | target in c)*.
ii) Initiate the new belief of all cells as 0.
iii) For each cell c, check if the cell is either Type1 or Type2. If it is say Type1, then you look at its horizontal and vertical neighbors which are of Type2. Let the number of such neighbors be n. The new belief of each of the n neighbors is incremented by *belief[c]/n* where belief[c] is the current belief of cell c.
iv) After all the cells have been updated, normalize the individual beliefs such that their sum is equal to 1.

Notice that, at any point in the fictitious time t, only the neighbors of cells that had a non-zero belief at time t-1 can have a non-zero belief at t. In this way, the number of cells where a target may exist and that, consequently, need to be searched is brought down drastically.

**Moving target with restricted search:** For the restricted search case, we used the same methodology as the one used in the case of the stationary target. The methodology was based on the rule: 'The next cell to be searched is the cell c with the highest value of *P(target for in cell c | Observations upto time t ) / Manhattan distance between current cell, cell c*.

## Performance:

Compared to the case where the object was stationary, the moving target can be found is much fewer moves. This can be attributed to the extra evidence of the form Type1 x Type2 we receive at every point, which in turn leads to more informed decisions. Moreover, the performance in the case of restricted search on a moving target, the algorithm proved comparable to the rule 1 and rule 2, despite being restricted and the estimate being pessimistic (i.e. by considering both the number of searches and the number of moves). When compared with the case of a stationary target, the increase in performance of the restricted search algorithm is hard to explain. It may be due to the fact that the number of cells where the search can take place in the case of a moving target is quite low, which is caused by the belief of most cells being equal to 0. The observed metrics are shown in table 2.

|  | Rule 1<br>(No. of searches) | Rule 2<br>(No. of searches) | Restricted search<br>(No. of moves + No, of searches) |
|---|---|---|---|
| Stationary target | 6665.21 | 4486.67 | 11183.44 |
| Moving target | **49.041** | **45.158** | **60.029** |

Table 2. Average number of searches over 1000 trials on a 50x50 board