RUTGERS UNIVERSITY, NEW BRUNSWICK

Probabilistic Hunting

Eshaan Gandhi

November 18, 2020

1 INTRODUCTION

Consider a landscape represented by a map of cells. The cells can be of various terrain types ('flat','hilly', 'forested', 'a complex maze of caves and tunnels') representing how difficult they are to search. (Just like minecraft.) Hidden somewhere in this landscape is a target. Initially, the target is equally likely to be anywhere in the landscape. Hence the initially:

$$P(Target\ in\ Cell_i) = \frac{1}{\#\ of\ cells}.$$

This is our prior belief in an event. We also have the ability to query or search a given cell. However the more difficult a terrain is, the harder it would be to search. There is a false negative rate that is associated with it. The false negative rate that were given to us are as follows:

$$P(\text{Target not found in Cell}_i\ |\text{Target not in Cell}_i) =$$

$$\begin{cases} 0.1 & \text{if Cell is flat} \\ 0.3 & \text{if Cell is hilly} \\ 0.7 & \text{if Cell is forested} \\ 0.9 & \text{if Cell is a maze of caves} \end{cases}$$

There is fortunately to false positive rate.

1.1 Implementation

The map is generated as a 25 by 25 grid and then every cell is randomly assigned a terrain type according to some initial probability. In my case: (flat with probability 0.2, hilly with probability 0.3, forested with probability 0.3, and caves with probability 0.2). We then select a cell to be the target. I chose a lower dimension to reduce the computation required, but my program can handle a 50 by 50 matrix and give the correct answer.

2 A STATIONARY TARGET

2.1 Computing belief

Given observations up to time t (Observations$_t$), and a failure searching Cell$_j$ (Observations$_t$+ 1 = Observations $t \wedge$ Failure in Cell$_j$), we use Bayes' theorem to efficiently update the belief state, i.e., compute:

$$P(\text{Target in Cell}_i \mid \text{ Observations} \wedge \text{Failure in Cell}_j)$$

So here we go:
We are trying to update the belief of a cell after an event. Now if cell j i searched and we get a success, we are done with the board. If we get a failure we have to compute all the new beliefs we have in the cells. There are two cases again.
If i = j i.e. we want to update the cell that was just searched and it failed. We can do so very efficiently and beautifully (bayes theorem is simply aesthetic) using bayes theorem.

$$P(Target\ in\ Cell_i \mid Failure\ in\ Cell_i) = \frac{P(Failure\ in\ Cell_i \mid Target\ in\ Cell_i)P(Target\ in\ Cell_i)}{P(Failure\ in\ Cell_i \mid Target\ in\ Cell_i)P(Target\ in\ Cell_i) + P(Failure\ in Cell_i \mid Target\ is\ not\ in\ Cell_i)P(Target\ is\ not\ in\ Cell_i)}$$

This boils down to:

$$\frac{P(Failure\ in\ Cell_i \mid Target\ in\ Cell_i)P(Target\ in\ Cell_i)}{P(Failure\ in\ Cell_i \mid Target\ in\ Cell_i)P(Target\ in\ Cell_i) + P(Target\ is\ not\ in\ Cell_i)}$$

The values of which are already known. We now come to the other case where i≠j

$$P(Target\ in\ Cell_j \mid Failure\ in\ Cell_i) = \frac{P(Failure\ in\ Cell_i \mid Target\ in\ Cell_j)P(Target\ in\ Cell_j)}{P(Failure in Cell_i)}$$

Which boils down to

$$\frac{P(Target\ in\ Cell_j)}{P(Failure\ in\ Cell_i \mid Target\ in\ Cell_i)P(Target\ in\ Cell_i) + P(Target\ is\ not\ in\ Cell_i)}$$

We now have all our update rules.

2.2 Computing belief in finding target in terrain

There are some terrains that have an exceptionally high false negative rate. To optimize and take into account for those we try to find the P(Target found in Cell$_i$ |Observations$_t$)
We go about this using marginalization:
P(Target found in Cell$_i$ |Observations$_t$) = P(found in Cell i $\wedge$ cell$_i$ contains target) + P(found in cell $_i \wedge$ cell$_i$ does not contain target)
The second party of is not possible because false positives cannot happen.
Hence the desired probability is P(found in Cell$_i$ $\wedge$ Cell$_i$ contains target)
That equals

$$P(\text{Found in cell}_i \mid \text{cell}_i \text{ contains target})P(\text{cell i contains target})$$

2.3 Rules

We now consider two decision rules to decide how we are going to explore the map.

- 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.
  To break the ties for either rule, we randomly choose the most viable option.

  Note: If a it takes more than 5000 searches to find a target, the search is called off

2.3.1 Rule 1

So I generated one map and then randomly allocated one cell as to be the target. I then let rule 1 try and find the target. The following the data that I found.
On average it takes 1188.63 many searches to find the target if found. I also thought that it would be interesting to see what the average number of searches required for each individual element. Here are my findings:

Terrain Type	Average Searches
Flat	731.48
Hilly	958.54
Forested	1526.13
Maze of caves	1795.25

2.3.2 Rule 2

So I generated one map and then randomly allocated one cell as to be the target. I then let rule 1 try and find the target. The following the data that I found.
On average it takes 1035.66 many searches to find the target if found. I also thought that it would be interesting to see what the average number of searches required for each individual element. Here are my findings

Terrain Type	Average Searches
Flat	217.55
Hilly	479.0
Forested	1156.03
Maze of caves	2470.32

2.3.3 Implications

We see that the two strategies perform pretty much the same when the map is pretty evenly distributed. I expect the 2nd rule to be better when the map is sparser and it is easier to get find the target if searched. The average searches definitely depends on the map. Rule 1 is more efficient when we have a map that is very hard to navigate and thus running away from "hard" areas would not be helpful.

2.4 How about we have to travel now

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.

- Basic Agent 1
  Simply travel to the nearest cell chosen by Rule 1, and search it.
  Here is the data that was collected. The average work that was done by the agent was 25,238 units. The data for the individual terrains is as follows:

Terrain Type	Average Work Done
Flat	16,653.75
Hilly	18636.81
Forested	27,434.35
Maze of caves	40,075.84

- Basic Agent 2
  Simply travel to the nearest cell chosen by Rule 2, and search it.
  Here is the data that was collected. The average work that was done by the agent was 17,751 units. The data for the individual terrains is as follows:

Terrain Type	Average Work Done
Flat	6714.62
Hilly	7128.91
Forested	24826.52
Maze of caves	41920.6875

- Basic Agent 3
  At every time, score each cell with(manhattan distance from current location)/(probability of finding target in that cell); identify the cell with minimal score, travel to it, and search it.
  This dramatically betters our score and results into a big decrease in work done. On average it takes only 3101.25 units of work. The by terrain data is as follows:

Terrain Type	Average Work Done
Flat	966.73
Hilly	2705.31
Forested	4288.64
Maze of caves	4750.19