CS440 Project 3: Search and destroy

Mustafa Sadiq (ms3035)

April, 2021

Introduction

We have four type of cells:

1 : flat

2 : hilly

3 : forested

4 : caves

An environment is constructed with a 50x50 array and a target location. Every cell type in the array is assigned randomly from above types. A random location is selected as the target and it is not accessible by the agent.
When an agent searches a cell, the environment can return a false negative with the following False Negative Rates (FNR):

1 : 0.1

2 : 0.3

3 : 0.7

4 : 0.9

The agent maintains a 50x50 array containing belief that cell has target. At time $t = 0$, we have

$$\text{Belief}[\text{Cell}_i] = P(\text{Target in Cell}_i|\text{Observations through time t}) = 1/2500$$

The agent is placed randomly on the map and can either search the current cell or move up down left right. A search is successful and map solved when the agent finds the target.

Problem 1

To efficienty update the belief system, i.e.:

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

We expand it using Bayes Theorem as:

$$\frac{\text{P(Failure in Cell}_j\text{ AND Observations}_t|\text{Target in Cell}_i\text{) P(Target in Cell}_i)}{\text{P(Failure in Cell}_j\text{ AND Observations}_t)}$$

Failure in Cell$_j$ and Observations$_t$ are independent,

$$= \frac{\text{P(Failure in Cell}_j|\text{Target in Cell}_i\text{) * P(Observations}_t|\text{Target in Cell}_i\text{) * P(Target in Cell}_i)}{\text{P(Failure in Cell)}j\text{) * P(Observations}_t)}$$

$$= \frac{\text{P(Failure in Cell}_j|\text{Target in Cell}_i\text{) * P(Target in Cell}_i|\text{Observations}_t)}{\text{P(Failure in Cell}_j))}$$

We first calculate Failure[Cell$_j$] as:

$$\text{P(Failure in Cell}_j\text{ AND Target in Cell}_j)$$

$$+ \text{ P(Failure in Cell}_j\text{ AND Target not in Cell}_j)$$

$$= \text{(FNR(Cell}_j\text{)*Belief[Cell}_j\text{]) + (1 - Belief[Cell}_j\text{])}$$

Then for every Cell$_i$:

- if $i = j$

$$\text{new belief Cell}_j = \frac{\text{P(Failure in Cell}_j|\text{Target in Cell}_j\text{) * P(Target in Cell}_j|\text{Observations}_t)}{\text{Failure in Cell}_j}$$

$$= \frac{\text{FNR(Cell}_j\text{)*Belief[Cell}_j]}{\text{Failure[Cell}_j]}$$

- if $i \neq j$

$$\text{new belief Cell}_i = \frac{\text{P(Failure in Cell}_j|\text{Target in Cell}_i\text{) * P(Target in Cell}_i|\text{Observations}_t)}{\text{Failure in Cell}_j}$$

$$= \frac{\text{Belief[Cell}_i]}{\text{Failure[Cell}_j]}$$

Problem 2

Given the current probabilty the target is in a given cell, we can find the probabilty that the target will be found in it if it is searched, i.e.:

$$\text{P(Target found in Cell}_i|\text{Observations}_t)$$

Expanding it as following formula:

$$\text{P(Target is in Cell}_i \text{ AND Success in Cell}_i|\text{Observations}_t)$$

which is:

$$\text{Belief[Cell}_i] * (1\text{-FNR(Cell}_i))$$

Problem 3

We implement two agents:

- Agent 1

  Iteratively travel to the cell with the highest probability of containing the target, search that cell. Repeat until target is found.

- Agent 2

  Iteratively travel to the cell with the highest probability of finding the target within that cell, search that cell. Repeat until the target is found.

For both agents, ties in probability between cells is broken based on shortest distance (minimal manhattan distance), and broken at random between cells with equal probability and equal shortest distance. The final performance of an agent is taken to be 'total distance traveled' + 'number of searches'.

We generate 10 maps, and play through each map (with random target location and initial agent location each time) 10 times with each agent and get the following performance (averaged and rounded up):

Agent 1 = **49001**

Agent 2 = **28819**

We can see that Agent 2 outperforms Agent 1 by a big difference. The reason why this happens is because Agent 2 uses the False Negative Rates in its belief system, searching cells that have a higher probability to find the target.

To further our analysis, we also look at target terrain type. We generate and solve maps until we have 10 maps and results for each target terrain type. Following are results averaged over the first 10 runs for each target terrain type (all results rounded up):

Agent 1

$$\begin{aligned} \text{flat} &= 3705 \\ \text{hilly} &= 23531 \\ \text{forested} &= 23691 \\ \text{caves} &= 52212 \end{aligned}$$

Agent 2

$$\begin{aligned} \text{flat} &= 536 \\ \text{hilly} &= 8340 \\ \text{forested} &= 6615 \\ \text{caves} &= 85178 \end{aligned}$$

Agent 2 outperforms Agent 1 as expected on all target terrain types except *caves*. The reason why this happens is because *caves* has a really high FNR, so Agent 2 really ignores *caves* for many searches as the probability to find target in *caves* is really low.

Problem 4

For our Improved agent we develop a product of engineering, Agent **FEAT** (Fuel Efficient All Terrain) on top of Agent 2. The motive behind this agent is to take into account the distance travelled between searches and try to minimize it. Agent FEAT calculates the probability target will be found in a cell and also account for the utility it will require using the belief map. Since we want to minimize the distance we divide the probability by the distance from the current cell plus one. The plus one accounts for the cost to search. Using the belief map that cell has target, we apply the following forumla:

$$\frac{\text{Belief[Cell}_i\text{] * (1-FNR(Cell}_i\text{))}}{\text{distance from current cell to Cell}_i + 1}$$

Like before, selecting a cell and score calculating remains the same.

We generate 10 maps, and play through each map (with random target location and initial agent location each time) 10 times with the Improved agent and get the following performance (averaged and rounded up):

Average score = 9695

We can see that our Improved agent beats both Agent 1 and Agent 2. This is because now our agent does not travel too much between searches and tries to minimize it, **saving fuel**. Relative to the basic agents, number of searches remains the same but now the agent travels conservatively.

To further our analysis, we also look at target terrain type. We generate and solve maps until we have 10 maps and results for each target terrain type. Following are results averaged over the first 10 runs for each target terrain type (all results rounded up):

$$\begin{aligned}\text{flat} &= 1390\\ \text{hilly} &= 3236\\ \text{forested} &= 5013\\ \text{caves} &= 29774\end{aligned}$$

We see that Improved agent beats both basic agents on **all terrains**, we can however note an increase in score for *flat* target terrain, since now we search closer cells for several iterations, before reaching the target cell.

Bonus: A Moving Target

The target is no longer stationary and can move to neighbouring cells when agent queries a cell and it returns failure. However, all is not lost and we are also given a new clue: whether the target is within Manhattan distance 5 of the current location.

To adapt our belief, we adapt it as follows:

- Target within Manhattan distance 5:

  Set probability of all cells greater than distance 5 as zero and normalize rest of the cells.

- Target not within Manhattan distance 5:

  Set probability of all cells less than equal to 5 as zero and normalize rest of the cells.

Implementing it, we encountered 'invalid value encountered error', since our belief map is not able to maintain high precision. What happens is that the sum of belief map goes towards non precise zero and division by it gives us NaN values. To deal with it we set probabilities to 0.0000000000000000001 instead of zero.
We generate 10 maps, and play through each map (with random target location and initial agent location each time) 10 times with all three agent and get the following performance (averaged and rounded up):

$$\begin{aligned} \text{Basic agent 1} &= 857 \\ \text{Basic agent 2} &= 636 \\ \text{Improved agent} &= 971 \end{aligned}$$

We observe that it takes a lot less iterations and it is faster to find the target, and scores are exponentially good compared to all three agents when target is stationary. All agents perform more or less the same however we can note that Improved agent does not beat Agent 1 and Agent 2 this time. The reason is that the target is more likely to be found at distance 5 and trying to search with less utility ignores it for several iterations as it is trying to minimize distance.

Addendum

I have read and abided by the rules laid out in the assignment prompt, I have not used anyone else's work for my project, my work is only mine.

Signed by: Mustafa Sadiq