CS 520: Assignment 3 - Probabilistic Search (and Destroy)

16:198:520

The purpose of this assignment is to model your knowledge/belief about a system probabilistically, and use this belief state to efficiently direct future action.

The Problem: 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. Hidden somewhere in this landscape is a **Target**. Initially, the target is equally likely to be anywhere in the landscape, hence starting out:

$$\mathbb{P}(\text{Target in Cell}_i) = \frac{1}{\#\text{ of cells}}. \tag{1}$$

This represents your **prior belief** about where the target is. You have the ability to **search** a given cell, to determine whether or not the target is there. However, the more difficult the terrain, the harder is can be to search - the more likely it is that even if the target is there, you may not find it (**a false negative**). We may summarize this in terms of the false negative rates:

$$\mathbb{P}\left(\text{Target not found in Cell}_i | \text{Target is in Cell}_i\right) = \begin{cases} 0.1 & \text{if Cell}_i \text{ is } \textbf{flat} \\ 0.3 & \text{if Cell}_i \text{ is } \textbf{hilly} \\ 0.7 & \text{if Cell}_i \text{ is } \textbf{forested} \\ 0.9 & \text{if Cell}_i \text{ is } \textbf{a maze of caves} \end{cases} \tag{2}$$

The false positive rate for any cell is taken to be 0, i.e., $\mathbb{P}\left(\text{Target found in Cell}_i | \text{Target not in Cell}_i\right) = 0$. Whatever the result of a search, however, it does provide **some** useful information about the location of the target, lowering the likelihood of the target being in the searched cell and raising the likelihood of it being in others.

If you find the target, you are done. The goal is to locate the target in **as few searches as possible** by utilizing the results of the observations collected.

Implementation: Maps may be generated in the following way: for a 50 by 50 grid, randomly assign each cell a terrain type, (flat with probability 0.2, hilly with probability 0.3, forested with probability 0.3, and caves with probability 0.2). Select a cell uniformly at random, and identify this as the location of the target. Note, this information is going to be stored and available in your program, but you **cannot** use it to determine which cell to check at any time.

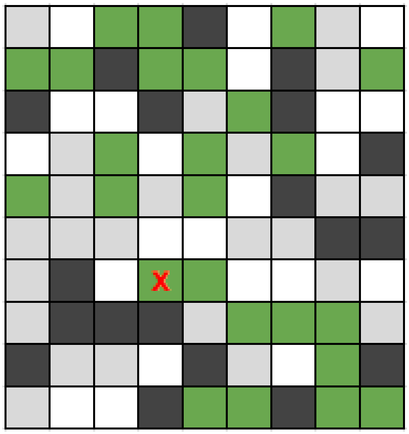

Figure 1: A simple 10 x 10 map with indicated target.

Separately, construct a 50 x 50 array of values representing your current **belief state**, the probability *given everything you've observed so far* that the target is in a given cell, i.e., at time t

$$\text{Belief}[\text{Cell}_i] = \mathbb{P}\left(\text{Target in Cell}_i | \text{ Observations through time } t\right). \tag{3}$$

Note, at time $t = 0$, we have $\text{Belief}[\text{Cell}_i] = 1/2500$.

At any time t, given the current state of **Belief**, determine a cell to check by some rule and **search it**. If the cell does not contain the target, the search will return **failure**. If the search does contain the target, the search will return **failure** or **success** with the appropriate probabilities, based on the terrain type. If the search returns failure, for whatever reason, use this observation about the selected cell to update your belief state for *all* cells (using Bayes). If the search returns success, return the total number of searches taken to locate the target.

Part 1: A Stationary Target

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

$$\mathbb{P}\left(\text{Target in Cell}_i | \text{Observations}_t \wedge \text{Failure in Cell}_j\right). \tag{4}$$

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:

$$\mathbb{P}\left(\text{Target found in Cell}_i | \text{Observations}_t\right)? \tag{5}$$

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?

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? Implement and compare the following agents:

   – Basic Agent 1: Simply travel to the nearest cell chosen by Rule 1, and search it.
   – Basic Agent 2: Simply travel to the nearest cell chosen by Rule 2, and search it.
   – 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.
   – Improved Agent: Your own strategy, that tries to beat the above three agents.

Part 2: A Moving Target

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 - every time you search a cell, you are now given two pieces of information instead of one: first, you are told whether or not the search was successful (same false negative rates as before); and if the search was unsuccessful, you are told whether or not the target is within Manhattan distance 5 of your current location.

- Factor this additional information into your search strategy in the case that you are able to search any point at any time. How does this additional information affect the number of searches needed to find the target? Compare to Rule 1 and Rule 2.
- Factor this additional information into your search strategy in the case that you are only able to move to your immediate neighbors in each time step. How does this additional information affect the number of actions (searches + motion) needed to find the target? Compare to Basic Agent 3 and your improved agent.