CS 440: Assignment 3 - Probabilistic Search (and Destroy) 16:198:440

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 0.25, hilly with probability 0.25, forested with probability 0.25, and caves with probability 0.25). Select a cell uniformly at random, and set this as the location of the target. Note, this information represents the **environment**. The location of the target is hidden from your agent, but the agent knows each cell type and can query the environment about a requested cell.

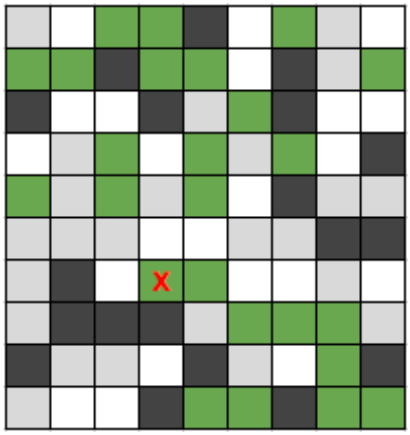

Figure 1: A simple 10 x 10 map with indicated target.

For your **agent**, construct a 50 x 50 array of values representing the agent's current **belief state**, the probability *given everything 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**, the agent must determine a cell to check by some rule and **search it**. If the cell does not contain the target, the environment will return **failure**. If the environment 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' Theorem). If the search returns success, return the total number of searches taken to locate the target.

Problems

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 the following situation. Your agent is dropped into the map at a random location and wants to find the target as quickly as possible. At every time step, the agent can either a) search the current cell the agent is in, or b) move to one of the immediate neighbors (up/down/left/right). We can consider the following basic agents:

   - Basic Agent 1: Iteratively travel to the cell with the highest probability of containing the target, search that cell. Repeat until target is found.
   - Basic 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 should be 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', and we want this number to be as small as possible.

   Generate 10 maps, and play through each map (with random target location and initial agent location each time) 10 times with each agent. Which agent is better, on average?

4) Design and implement an improved agent and show that it beats both basic agents. Describe your algorithm, and why it is more effective than the other two. Given world enough, and time, how would you make your agent even better?

Bonus: A Moving Target

In this section, the target is no longer stationary, and can move between neighboring cells (up/down/left/right). 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.

- Adapt Basic Agents 1 and 2 to this new situation and extra information. What modifications to your probabilities did this require? Are they still able to find the target? Which one is better?
- Adapt your Improved Agent to this new situation and extra information. What modifications does your algorithm require? Is it still able to find the target? Does it still beat Basic Agents 1 and 2?