Anthony Triolo
Section 3
NetID: ajt182

Biyun Wu
Section 5
NetID: bw366

CS440 Project 3: Probabilistic Search (and Destroy)

2021-04-07

Problem 1

Given observations up to time $t(Observations_t)$, and a failure searching $Cell_j(Observations_{t+1} = Observations_t \wedge$ Failure in $Cell_j)$, how can Bayes' theorem be used to efficiently update the belief state? i.e., compute: $P($Target in $cell_i | Observations_t \wedge$ Failure in $cell_j)$.

First, let's begin with the formal proof of the formulas we ended up using to update the belief state of each cell. Note that $B^{t+1}_{cell_i}$ represents the belief state at time $t+1$ of $cell_i$:

- If $cell_i$ is not searched at time $T = t+1$:

$$
\begin{aligned}
B^{t+1}_{cell_i} &= P(\text{Target in } cell_i | Observations_t \wedge \text{Failure in } cell_j) \\
&= \frac{P(Observations_t \wedge \text{Failure in } cell_j | \text{T in } cell_i) \times P(\text{T in } cell_i)}{P(Observations_t \wedge \text{Failure in } cell_j)} \\
&= \frac{P(Observations_t | \text{T in } cell_i) \times P(\text{Failure in } cell_j | \text{T in } cell_i) \times P(\text{T in } cell_i)}{P(Observations_t) \times P(\text{Failure in } cell_j)} \\
&= \frac{P(\text{T in } cell_i | Observations_t) \times P(\text{Failure in } cell_j | \text{T in } cell_i)}{P(\text{Failure in } cell_j)} \\
&= \frac{B^t_{cell_i} \times P(\text{Failure in } cell_j | \text{T in } cell_i)}{P(\text{Failure in } cell_j)} \\
&= \frac{B^t_{cell_i} \times 1}{\sum_n P(\text{Failure in } cell_j | \text{T in } cell_n) \times P(\text{T in } cell_n | Obs_t)} \\
&= \frac{B^t_{cell_i} \times 1}{P(\text{T in } cell_j | Obs_t) P(\text{Failure in } cell_j | \text{T in } cell_j) + P(\text{T not in } cell_j | Obs_t) P(\text{Failure in } cell_j | \text{T not in } cell_j)} \\
&= \frac{B^t_{cell_i}}{B^t_{cell_j} \times FN_j + 1 - B^t_{cell_j}}
\end{aligned}
$$

- If $cell_i$ is searched at time $T = t+1$:

$$
\begin{aligned}
B^{t+1}_{cell_i} &= P(\text{Target in } Cell_i | Observations_t \wedge \text{Failure in } cell_i) \\
&= \frac{B^t_{cell_i} \times FN_i}{B^t_{cell_i} \times FN_i + 1 - B^t_{cell_i}}
\end{aligned}
$$

We can see that we are able to change the initial probability into a form that uses Bayes' Theorem. We then made the observation that all observations are independent from each other, since one observation does not affect another. This is applied to both the numerator and denominator. We can then see that some of the components from the numerator and denominator can be simplified to represent $P(\text{T in } cell_i | Observations_t)$, which is just the belief state up to time t. When $cell_i$ is not the one searched at time $t+1$, the probability that there's a failure at the cell that was searched given that the target is in $cell_i$ is always 1, because this will always occur. If $cell_i$ is the one searched, then we multiply be the false negative probability provided by the problem. In the denominator, we look at all of the possible ways that there could be a failure in $cell_j$, and the probabilities of doing so. For all cells except $cell_j$, this probability is equal to it's current belief state, while for $cell_j$, it's the same as the value in the numerator. This denominator value is the same across the belief of all cells.

Problem 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:

We know that in order to calculate the probability of finding the target in a cell, two things need to happen. The first is that the target is located in the cell, and the second is that the target is found when it is searched in the cell that contains the target. The agent doesn't know where the target is located but it can postulate about where it is. As shown in the previous problem, the probability of the target being in the cell given the observations made so far is simply the belief state up to that point. The probability of finding the target in a cell when that cell is searched is related to the false negative probabilities provided by the problem. We know that these probabilities represent the probability of not finding the target when searching in the cell that contains it. So, 1 minus this probability is the probability of finding it given it's located in that cell. By combining these two probabilities, we get the final probability:

$$P(\text{Target found in } Cell_i | Observations_t) = B^t_{cell_i} \times (1 - FN_i)$$

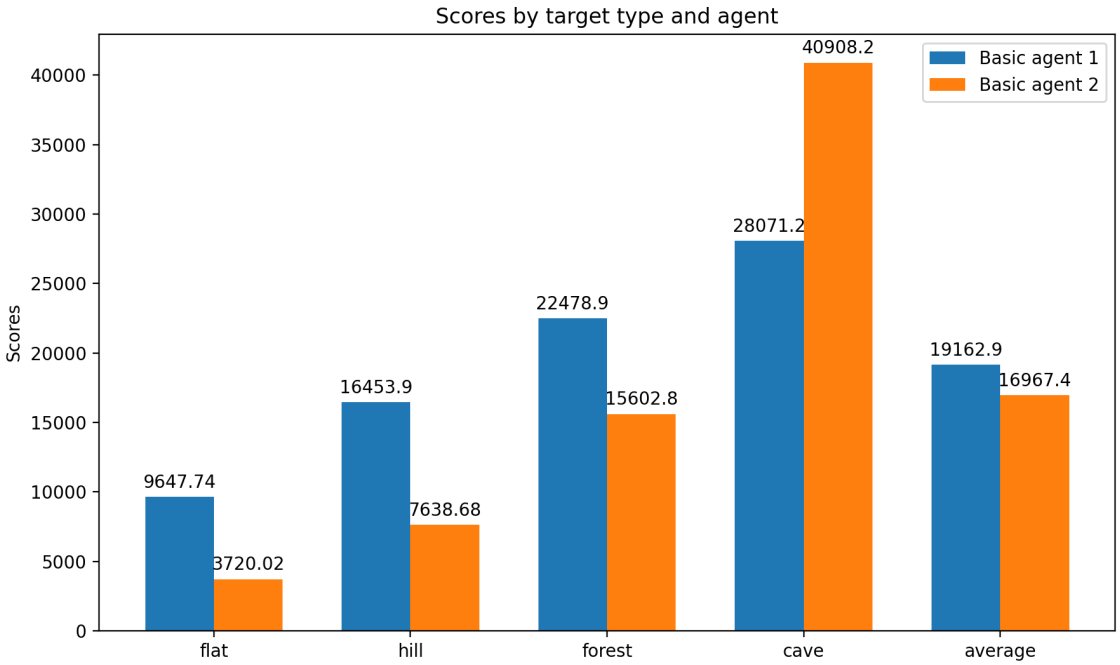
Problem 3

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?

When looking at the graph below, we can see that on average, basic agent 2 has a slightly better score. However, we can see that basic agent 1 was significantly better when it came to searching for the target in cave cells. This is due to the fact that these types of cells end up getting prioritized based on the belief state because the probability of them being located in these types of cells don't decrease as much as the others when a failure occurs. Because of this, the belief state at cave cells stay relatively high, and so basic agent 1 ends up visiting these pretty frequently. The problem that basic agent 2 has with these cells is that because $1 - FN_i$ is only 0.1, it's probability ends up getting scaled down significantly, so basic agent 2 does not search these types of cells as often. This factor, paired with the fact that it's already difficult to find the target in these types of cells, leads to a bad score. However, at the cells with lower false negative values, basic agent 2 is much better than basic agent 1 because it searches these types of cells more often, and so if it happens to miss finding the target cell, it comes back much more quickly to these types of cells than basic agent 1 since the probability is not that low compared to that of cave cells like they are in basic agent 1. This is the biggest reason basic agent 1 is so bad at finding targets in flat and hilly cells. If it misses it on the first pass, it takes a long time to come back since it ends up searching all of the other cells, and then highest probabilities left are in cave and forest cells, and so it can take a while to revisit a flat or hilly cell.



Problem 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?

In order to mitigate the cost of travel distance, the strategy for the improved agent is conducting multiple observations on the same cell before updating the belief state and making decisions based on the belief state or the confidence state (which is $belief \times (1 - \text{false negative rate})$) according to certain threshold.

Based on the false negative rates of each kind of target, the agent searches flat, hill, forest and hill targets 2, 4, 6, and 8 times, respectively. In this way, the false negative rates are reduced for each update:

Target type	False negative rate after designated number of searches
Flat	0.01
Hill	0.0081
Forest	0.117649
Cave	0.43046721

However, the false negative rate for the cave target is still not ideal even though it is searched 8 times. Its value is still relatively high. Considering the fact that basic agent 1 performs much better than the basic agent 2 on cave targets, the improved agent initially makes the decision of which cell to explore based on $belief_i \times (1 - \text{false negative rate}_i)$, which is $P(\text{Target found in } Cell_i | Observations_t)$, when the total score is less than $threshold = 10000$. This threshold was decided based on testing of various values of the threshold. By doing this, the agent ends up prioritizing flat, hilly, and sometimes forest, cells in the same way as basic agent 2 and also searches these cells many times. Once it gets to the threshold, it's fairly likely that the target is not in one of these types of cells since the false negative probabilities get so low. Once the threshold is reached, the improved agent determines its next destination based on the belief state about where the cell is located. This is similar to that of basic agent 1. In this way we take the strengths of both basic agents and implement them into the improved agent to get an overall improvement. By not only prioritizing cave cells, similar to basic agent 1, we also search them many times to decrease the likelihood of missing them.

Formula to update the belief state:

- To update the belief for $cell_i$ that are not searched based on several searches on $cell_j$ at time $T = t + 1$:

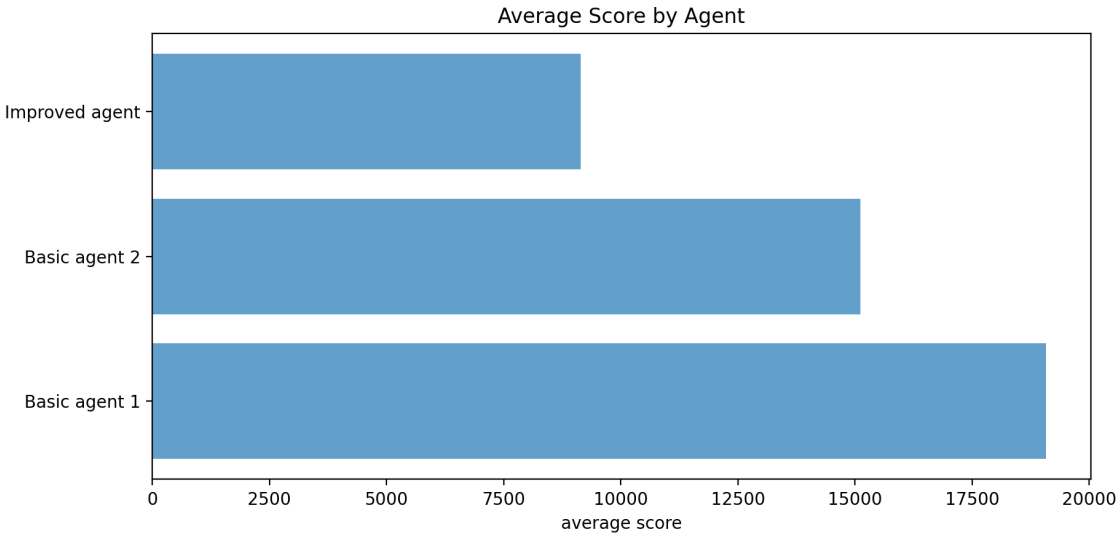
$$B^{t+1}_{cell_i} = \frac{B^t_{cell_i}}{B^t_{cell_j}(FN_j)^n + 1 - B^t_{cell_j}}$$

- To update the belief for $cell_i$ that is searched n times at time $T = t + 1$:

$$B^{t+1}_{cell_i} = \frac{B^t_{cell_i} \times (FN_i)^n}{B^t_{cell_i}(FN_i)^n + 1 - B^t_{cell_i}}$$

These formulas are similar to those shown in problem 1, but we raise the false negative rate to the number of times the specified cell was searched since this is the probability of that false negative instance occurring that many times.

Next, we can look at the comparison between the improved agents and the basic agents by average score:



We can clearly see that the test shows that the improved agent costs 52% and 40% less than basic agent 1 and basic agent 2 respectively.

If we had enough time to try something different, we might have tried a method of trying to sort of ”predict the future”. By this we mean that we would theorize what would happen if the next cell we wanted to search failed, without searching it, and check whether or not there would end up being a cell with a high probability on that next step that is closer to the cell we are at currently. This would potentially limit the number of steps that the agent has to take, since it might not have to go out of its way to then end up coming back. The only potential downside of this is that it skips the cell that it was supposed to go to and the target was there. However, since the agent was already planning on searching this cell, there is no additional cost as compared to just going to that cell in the first place.

Honor Pledge

We, Anthony and Biyun, certify that all work is our own, and that no work is copied or taken from online or any other student’s work

Contributions

Anthony and Biyun came up with different formulas which generated the same result. It was decided that Biyun's formula was more straight forward and is used in this project.

Anthony polished the proof for the formulas, implemented the environment and basic agents, as well as proposed the strategy of searching a cell multiple times for the improved agent.

Biyun implemented the improved agent by introducing the threshold to make the improved agent cost less to find the cave target. Also, Biyun handled the testing and plotproduction.

Both Anthony and Biyun contirbuted to writing the final report