CS 440: Introduction to Artificial Intelligence **Rutgers: Spring 2021**

Project #3: Probabilistic Search (and Destroy)

April 9, 2021

Names: Aamna Farooq (af704), Nada Elshamaa(nhe12), and Asma Makhdoom(aam355)

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:

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

Solution. Bayes' theorem can be used to efficiently update the belief state by deriving an equation using Bayes that we would then use to update the belief state at each time step. From the given probability, using Bayes' we knew:

P(Target in Cell$_i$ | Observations$_t$$\wedge$Failure in Cell$_j$).

=

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

=

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

=

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

P(Failure in Cell$_j$ | Target in Cell$_i$)
if j!= i: P(Failure in Cell$_j$ | Target in Cell$_i$) = 1
if j=i: P(Failure in Cell$_j$ | Target in Cell$_i$) = false negative rate

from this we are able to derive the following:
P(Target in Cell$_i$ | Observations$_t$) = Current Probability in Belief State
P(Failure in Cell$_j$ | Target in Cell$_i$) = False Negative Rate
P(Failure in Cell$_j$ | Observations$_t$) = Normalizing Factor

which gives us the equation to be able to update the belief state:
Belief[i][j] =

$$\frac{\text{Belief[i][j] * False Negative Rate}}{\text{Normalizing Factor(New Total Probability of Belief State)}}$$

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:

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

Solution. P(Target in Cell_i | Observations_t)

= P(Target in Cell_i $\wedge$ Search of i is successful | Observations_t)

= (1 - False Negative Rate of Cell_i) * Belief[Cell_i]

For each cell in the grid, we calculate the above formula using the belief state we derived in question 1. We maintain a running maximum for each cell and ultimately choose to search the cell with that maximum value. If there are any ties in maximum value we break those ties using minimum Manhattan distance and if there ties with distance, we choose randomly.

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, searchthat 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?

Solution. On average, after running the program with an average of 10 with 10 different 50 by 50 mazes, we found that agent 2 was better than agent 1.
The improved had the least score:
Basic Agent 1: 53255.61
Basic Agent 2: 36542.92

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?

Solution.

We modeled our improved agent similar to Agent 2 but we added an extra component of distance into our algorithm. This way, instead of solely using probability, we are balancing probability and distance to decide which cell to search next. For each cell we calculate the probability using the formula from Agent 2 and divide that by the Manhattan distance of the current cell to that particular cell. We keep a running maximum of this value and choose to search the cell that has the highest value.

This has proved to be highly effective compared to Agent 1 and Agent 2. On average, after running the program with an average of 10 with 10 different 50 by 50 mazes, we found that the improved agent performed the best out of the 3 algorithms.

Basic Agent 1: 53255.61
Basic Agent 2: 36542.92
Improved Agent: 14104.75

Our improved Agent is more effective than the other Agents because instead of solely using probability, we are balancing probability and distance to decide which cell to search next.

If we had enough time and resources, we would implement an improved Agent that also looks further time steps into the future. This would make the algorithm more effective because it could anticipate future steps and reduce overall travel distance since distance usually costs more than searches.

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?

In order to implement the moving target, depending on whether or not the target is within Manhattan distance 5 of the current location, we change how we pick the maximum cell. If we know that the target is also within Manhattan distance 5 of the current cell we take the maximum probability of cells that are within Manhattan distance 5. Otherwise, we only consider cells have a Manhattan distance that is greater than 5. We do this because we are guaranteed that the location of the target is either within Manhattan distance of 5 or not. We do the same for Agent 2 and Improved Agent as well. In all cases, the agents are always able to find the target. The improved Agent performs the best out of the 3. For example, we ran the 3 (moving) agents on the same maze of size 10 by 10:
Agent 1: 10582
Agent 2: 6284
Agent 3: 1071

Acronym

H.A.S.
Hide And Seek

This acronym relates to our project because we are in essence playing hide and seek with the target where we are trying to locate it but even when locating it, it may not be visible to us because of the terrain type which enables the target to "hide" better as we are trying to "seek" it.

Work Distribution
The work is our own and not copied or taken from any other students.

To work on this project we would meet up over video calls daily and discuss problems and our solutions. One person would then screen share and code while the others would contribute and also assist in coding using the request remote control feature in zoom. We would alternate in screen sharing and upload to git for version control.

The report was done similarly. We met to complete it as a group over video call.