1) Given observations up to time t (Observationst), and a failure searching Cell j (Observationst+1 = Observationst^

Failure in Cellj ), how can Bayes' theorem be used to efficiently to update the belief state, i.e., compute:

P (Target in CellijObservationst ^ Failure in Cellj) :

The formula to apply here would be P(target is in Cell I | observations t)\* P(target not found in Cell I | target is in Cell i). So basically the Probability that the target is in the cell is the Probability it is in cell I given the observations at t multiplied by the failure rate give that the target is in Cell i.

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:

P (Target found in Cellij | Observationst)?

This is P(belief state t | observations t)\* (1-P(target not found in Cell I | the target is in Cell i)).

So first we have some belief state Probability give our observations at t. when checking which cells to pick this Probability usually the highest in the grid. Even thought it is the highest we still may not find the target so then we multiply by the failure rate given that the target actually is in the cell.

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?

2

The way we implemented this in our program is to first find the current belief state that is the highest in the entire grid say P. then search that cell and take P \*( failure rate of that terrain)= P’ which is the new belief state for that cell . Then take P\*( the success rate of that terrain)/ 2499 = P’’. now take P’’ and add this Probability to all other cells ( excluding cell i) to update the current belief state.

The average searches per 500 runs was (Rule1):5407.378

Now for Rule 2 the only difference between the two is that first when searching for the large current belief state given observations at t you multiply (but don’t update) each cell with the Probability

Of success of finding the target. Then continue as normal in Rule 1.

The average searches per 500 runs was(Rule 2):9625.234

Surprisingly Rule one on average and for fixed maps would have a shorter search time. At first glance it does make sense that you should only care about where the target would actually be as that is our goal. Searching the cell that you are mostly likely to find the target isn’t actually the goal of the agent.

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? Derive a decision rule based on the current belief state and current location, and compare its performance to the rule of simply always traveling to the next cell indicated by Rule 1 or Rule 2. Discuss.

The decision rule we choose to implement was simply taking the largest belief state of the neighbors. One reason for this is just because a cell has a higher belief state (using either Rule 1 or 2)

Still does not necessarily mean that you are better off searching there. Using our rule the searching is more efficient given that the cost is always 0 of searching but does overlook the bigger picture we choose to ignore this for reasons stated above.

As was expected being restricted to only be able to search neighboring cells was extremely detrimental to our efficiency.

The average searches per 500 runs was(Rule 1):6548.944

The average searches per 500 runs was(Rule 2):8304.302

5) An old joke goes something like the following:

A policeman sees a drunk man searching for something under a streetlight and asks what the drunk has lost. He says he lost his keys and they both look under the streetlight together. After a few minutes the policeman asks if he is sure he lost them here, and the drunk replies, no, and that he lost them in the park. The policeman asks why he is searching here, and the drunk replies, "the light is better here".

In light of the results of this project, discuss.

The parallel of this joke to the Project is that where he lost his keys could have been in a cave of mazes where the Probability to actually find the target give that it is in the cell is .1 which is horrible. The drunk man first decides to check the fields first even though he is certain he the target isn’t there, he still searches where the Probability of finding the target give its in fields is pretty low. The key difference between the project and this joke is that, the drunk man is certain he didn’t lose his keys in the light. We are actually never certain where the target is or is not so, if the Probabilities between cells are fairly close but one is a field and the others a cave of mazes it would be beneficial to search the field first. Our implementation did not account for Probability that where relatively close and the terrain difference between them.

After implementing the moving target for part two of the project, we let the search make use of

this extra information by simply scanning the grid and improving the probability of finding the

target in the two cells that are of Type1xType2 that the moving target function reports back.

Specifically after we move the target and get the report of the Type1xType2 back, we have a

function called update that will scan the grid and count the number of matches the scan finds and

then change the probability of those cells to 1.0/#ofMatches. That also has to do with our belief

state, because it will help observe other parts of the grid that the search didn’t get to yet and also

allows for the search to narrow down where the target may be, especially after a number of failed

searches. The probabilities of the cells that could possibly contain the target goes up so a general

part of the grid will have a raised probability. This also can be explained by the general

application of Bayes Theorem. After every try to search for the target, we can update the

probabilities based on what gets reported back and if the target moves maybe 50 times, again, a

general focused part of the gridworld will have a higher probability then the rest of the grid.

|  |
| --- |
| Rule 1: Rule 2: MovingTarget: |
| Counts: 4143 12984 2164 |
| 1975 6444 2330 |
| 6610 310 3103 |
| 37798 18053 2646 |
| 2144 253 4219 |
| 11726 2102 5152 |
| 6470 7324 7048 |
| 9536 134260 1643 |
| 571 29236 144 |
| 5986 39580 573 |
| Average count: 8695.9 25054.6 2902.2 |

As you can see, compared to Rules 1 & 2, the MovingTarget one is generally more tight in terms

of its count while the other two search methods seem almost random. When we use the rules to

search for the target, it just looks at the probabilities of whether it may be what our belief state

considers to have the highest probability of containing the target or finding the target. However,

with the MovingTarget rule use, we do not search to update our belief state, we search based on

our belief state. What I mean by that is that in the explanation above I explained how the new

rule changes how we view the belief state, so it actually narrows the cells to search down, while

the other two rules simply look and hope that the highest probability has the target.

Type1xType2 > Rule1 > Rule2.

If we are comparing question 4 between the Rule 1 and Rule 2 and the MovingTarget rule, even

then the MovingTarget rule is going to be faster in the average case. The extra information of

what Type1xType2 cells contain the target allows for the “agent” to update it’s belief state if the

first search is a failure and then it can then travel to the closest Type1xType2 matches and then

update it’s belief state if it fails there. Again, as said before, after a couple of failed searches

there will be a general part of the grid that is “focused” with a higher probability then the rest of

the grid. The “agent” can prioritize that area of cells and use less “actions” on randomly traveling

to high probability cells that may hold or contain the target. Especially with a moving target, the

probabilities in our belief state of what cell may hold a target is always going to be a state behind

or off, considering the target moves one cell randomly after each failed search.

5) “A policeman sees a drunk man searching for something under a streetlight and asks what the

drunk has lost. He says he lost his keys and they both look under the streetlight together. After a

few minutes the policeman asks if he is sure he lost them here, and the drunk replies, no, and that

he lost them in the park. The policeman asks why he is searching here, and the drunk replies,

”the light is better here”.”

In the light of this project, the joke or short story was originally told to explain what the

streetlight effect is. That is the phenomena that people search for things that they are looking for

only where it is easiest to look. In our project, in the search Rule 2 is a perfect example of what

the drunkard did in the joke above. He looks under the streetlight where it is well lit because that

light gives him the highest chance of finding his keys, even though he had the knowledge that his

keys were not there. This is just like how the Rule 2 tells the search to look for the target in the

places with the highest probability to find the target.