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

Raj Sundhar Ravichandran, Justin Chou, Vatsal Parikh

November 20, 2017

Source Code and Contributions

Git Repository: https://github.com/rsundhar247/AI-BayesianSearch

1 A Stationary Target

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:**

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

   In the initial grid, the probability that each cell contains the target is equally likely, i.e., 1/(total number of cells). In the case of the 50x50 grid, our initial probability for each cell containing the target is (1/2500). When we search a cell for the target and the search fails, we update our belief system to reflect the new probability for that cell to contain the target to be equal to the probability of the cell containing the target multiplied by the probability of failing a search given the target is in the cell.

   Current P(Target in Cell) * P(Target not found in cell | Target in Cell)

   Additionally, we have to distribute the reduced probability proportionately to other cells based on their current probabilities. We are distributing the below probability to other cells.

   Current P(Target in Cell) - [Current P(Target in Cell) * P(Target not found in cell | Target in Cell)]

   If we lower the probability for the target to be in the cell we searched, then the probabilities for the other cells increase proportionately. In particular, we scale up each other cell's probability to contain the target by

   $$1 + \frac{\text{P(target in searched cell) * P(Target not found in cell | Target in Cell)}}{\text{1 - P(Initial target in Searched Cell)}}$$

   This ends up as,

   $$\text{P(Target in current cell) * } \left(1 + \frac{\text{P(target in searched cell) * P(Target not found in cell | Target in Cell)}}{\text{1 - P(Initial target in Searched Cell)}}\right)$$

   Thus, cells in which the greater initial probabilities to contain the Target will increase more significantly than those cells with smaller probabilities.

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:**

   $$\textbf{P(Target found in Cell}_i \mid \textbf{Observations}_t\textbf{)?}$$

   The probability that a target will be found in a cell is,

   P(Target found in Cell | Target in Cell) * P(Target in cell | Observations)

   P(Target found in Cell | Target in Cell) is equal to 0.9, 0.3, 0.7, or 0.1 depending on, if the cell is Flat Land, Hilly, Forested, or Maze of Caves respectively.

   P(Target in cell | Observations) will be stored for each cell in our knowledge base based on our calculations in question 1. Initially, for t=0, this probability will be (1/2500) for all the cells.

   P(Target found in Cell$_i$|Observations$_t$) = P(Target found in Cell$_i$|Target in Cell$_i$) * P(Target in cell$_i$|Observations$_{t-1}$)

$$\text{P(Target in cell}_i \mid \text{Observations}_0) = \frac{1}{2500}$$

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?

   Rule 1 is consistent with the probabilities calculated and stored in our knowledge base in Parts 1 and 2. Rule 2 then takes into consideration the probability of successfully finding the target given the target is located at that cell. In other words, Rule 2 scales the probabilities calculated in Rule 1 for each cell by their respective probabilities P(Target found in cell | Target in cell). Flat Land will be multiplied by 0.9, Hilly will be multiplied by 0.7, Forested will be multiplied by 0.3, and Maze of Caves will be multiplied by 0.1.

   After simulating 100 trials for each rule in a 50 * 50 grid, we get the following average numbers of searches for each rule.

   Increased cap to 50*50*100 searches, 500 trials

   All results below are average values for stationary target

Terrain type	Rule 1 average	Rule 2 average	Location based action average for number of moves plus distance traveled	Location based action average for number of moves
Land	2820.3831775700933	869.675	3887.5978260869565	2243.413043478261
Hill	6192.5723684210525	2063.846625766871	4620.196078431372	2635.97385620915
Forest	7129.368055555556	4860.3057324840765	9389.72463768116	5516.840579710145
Cave	9577.958762886597	13843.29	20287.4358974359	13167.863247863248
Weighted Average	6476.2505152843205	5019.8387074752845	9037.98295953833	5528.09958904409

   On Average, Rule 1 performs worse than Rule 2 because Rule 1 searches the different types of cells more evenly than Rule 2 does. Rule 2 prioritizes finding the target over where the target actually is, so Rule 2 will search Flat Land first, then Hilly, then Forested, and finally Maze of Caves.

   If the target is in the Flat Land or Hilly area, then Rule 2 will search these spaces first before searching and forest or cave, whereas Rule 1 will impartially search all of different cells. Rule 2 will then be more efficient. If the target is in a Forested or Maze of Caves cell, then Rule 2 will search these cells only after searching the Flat Land or Hilly Area. Even though it seems like Rule 2 should be less efficient in this case, because the Forested and Maze of Cave cells are difficult to search (0.3 and 0.1 probabilities of being successful, respectively), the average number of searches to find the target will be significantly larger, and the order in which the types of cell are searched will has a smaller impact on the average number of searches.

   If we compare the differences between the number of searches in Rule 1 and Rule 2, we see that for Flat Land:
   (2820-870)/870 = 224%
   Rule 1 requires 224% more searches than Rule 2 if the target is in Flat land
   For Caves: (9578-13843)/13843 = -30.0% Rule 1 requires about (1-30.0) = 70% of the number of searches of Rule 2 if the target is in a Cave.
   Thus, we see that in the best case, Rule 2 significantly improves the efficiency of the search if the target is relatively easy to find, but is only slightly worse if the target is difficult to find.

   If the frequency of Flat Land and Hills were to increase, or if the relative frequencies of the four types of land favored Flat Land and Hills in any way, then we should expect to see Rule 2's average search time improves even more, as the target will be even more likely to be located in Flat Land or Hills, and Rule 2's bias towards these two lands will be more efficient. Also, if the probabilities of successfully finding the target given the target is in a cell are more high, Rule 2 continues to get better.

   We simulated an alternate situation where P(Target found in Cell | Target is in Cell) for Flat Land = 0.99, for Hilly = 0.9, for Forested = 0.1, for Maze of Caves = 0.01

We obtained the following averages:

All results below are average values for stationary target

Terrain type	Rule 1 average	Rule 2 average	Location based action average for number of moves plus distance traveled	Location based action average for number of moves
Land	6838.263157894737	448.4059405940594	2142.2772277227723	1209.148514851485
Hill	17138.8875	3067.3333333333335	2999.53164556962	1703.7974683544303
Forest	65716.6690140845	14029.865771812081	19741.716981132075	15499.987421383648
Cave	53872.30097087379	83249.98	89039.12195121951	79934.53658536586
Weighted Average	36998.77977997906	21868.836919662434	25058.654423798966	21389.872486964894

We ran the simulation by changing the frequency of different terrains as follows:
Lands - 0.4, Hills - 0.4, Forests - 0.15, Caves - 0.05

We got the result as follows:

All results below are average values for stationary target

Terrain type	Rule 1 average	Rule 2 average	Location based action average for number of moves plus distance traveled	Location based action average for number of moves
Land	2195.6439024390243	1011.4522613065327	2979.19801980198	1463.648514851485
Hill	3454.8445595854923	2727.265625	5124.868020304569	2585.776649746193
Forest	3611.614457831325	5132.058139534884	9373.613333333333	4833.173333333333
Cave	4542.105263157895	8572.08695652174	14141.423076923076	7576.2307692307695
Weighted Average	3029.0428166424	2693.9002232789326	5354.739569888773	2723.55760430061

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.**

We create the rule such that “searcher” begins at (0,0) and must travel to other cells to search with each unit distance traveled equal to 1 action. To choose which cell to search, we calculate the “average” utility of a potential search given the required number of actions needed to perform the search. To search the current location, 1 action is required. To search a location 4 units away, 5 actions are required. So we create the rule to calculate as:

$$\frac{\text{P(target is in cell) * P(Target found in cell | Target is in Cell)}}{\text{Costs for the required action}}$$

By selecting the maximum utility, we ensure that our moves maximize the likelihood of finding the target per action spent. Under this rule, we expect our performance to be worse that Rule 1 and Rule 2, as we are now constrained by how far we must walk to search a location.

As per our results below, on average we perform 36050.33 searches and walk 12785.85 distances for a total of 48836.18 actions. But really, we are only concerned with 36050.33 searches, which is significantly larger than Rule 1’s average of 30754.0 and Rule 2’s average of 26000.61. Again, this makes sense as when we start at (0,0), we are unable to search (49,49) as willingly as we are (1,0), despite (49,49) potentially being more probable to contain the target.
500 trails, max search of 50*50*100

Terrain type	Location Based Action Average for Number of Moves plus Dist Traveled	Location Based Action Average for Number of Moves
Land	3887.5978260869565	2243.413043478261
Hill	4620.196078431372	2635.97385620915
Forest	9389.72463768116	5516.840579710145
Cave	20287.4358974359	13167.863247863248
Weighted average	9037.98295953833	5528.09958904409

But given a real time scenario, this method of searching would make more sense, as one has to take into account the logistics constraints attached to the search. To be more realistic, this method is being implemented by the military while tracking down the terrorists, they search and clear grid by grid even though they are less likely to find the terrorists near by.

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 drunk man's behavior is consistent with Rule 2 in part 3. Even though the man is confident that his keys are in the park, the man chooses to search under the streetlight, where he has a higher probability of successfully finding his keys if at all his keys were there. While the light may seem hopeful to the man, if the keys have a minimal chance of being under the light to begin with, then the man's probability for finding his keys is also minimal. Conversely, even though his keys may more likely be in the park, the man is discouraged by the difficulty of actually finding the keys in the dark.

From our results in part 3, the man's best course of action depends on the true probability of his keys being in the park vs. being under the street light and the success rate of finding the keys in the park vs the street light given the keys are truly in the park or light. If the probability of the keys being in under the light is significantly greater than the probability of the keys being in the park or if the probabilities are relatively equal, then it would be more efficient for the man to search under the light first. If the man loses his keys frequently, then his strategy of looking under the light first would average out the time he spends searching for the keys, especially if the difficulties for finding the keys in the park vs. under the street light are drastically different. If the man wishes to also save time it takes to move to a search location and the man happens to be closer to the street light then the park, then it is more efficient for him to exhaustively search under the street light before moving on to the park that is farther away.

Again, these conclusions were made under the assumption that the probability of the man losing his keys in the park vs. under the light are relatively equal.

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 - whenever the target moves, surveillance reports to you that the target was seen at a Type1 x Type2 border where Type1 and Type2 are the cell types the target is moving between (though it is not reported which one was the exit point and which one the entry point.
Implement this functionality in your code. How can you update your search to make use of this extra information? How does your belief state change with these additional observations? Update your search accordingly, and again compare Rule 1 and Rule 2.

Initially the belief state for each cell is constructed as (1/2500). Once, a random check reveals the movement of the target between 2 cells, we mark the cells that contains any of these 2 terrain types. Relative probability for rest of the cells is made as 0 and their initial probabilities are split equally among the cells that are marked for target. Now the cell with maximum probability is chosen next to look for the target. In case of rule2, the maximum of relativeProbability * probabilityForFinding is chosen.

RelativeProb(i,j) = RelativeProb(i,j) * (1 + (Probabiliies that are made to zero | Cells that are not zero))

In the initial steps, since there were too many options to chose from, the program was random in choosing the cells and was not near the target most of the times. But as the program starts building the knowledge base, the cells were narrowed down to few numbers and the program was able to identify the target in a fewer number of steps, compared to stationary targets.

As it can be seen from the data collected, the average steps taken in moving target is smaller than the stationary target. This is because, we constantly get details about the movement of the target and this helps us to predict and locate the location of the target much faster.

Moving Target Rule 1 Average: 4946.864
Moving Target Rule 2 Average: 5235.728