CS 520 : Project 2 - Partial Sensing

16:198:520

Due: October 17 by 11:55 PM, Submitted via Canvas

We return to the gridworld of Project 1. We have a **dim** by **dim** square grid, where cells are blocked with probability p. A robot wants to navigate from the upper left to lower right corner as before. In the previous project, we looked at Repeated A^*, and the effect of the field of view - being able to see all four cardinal neighbors vs being 'blindfolded', and only being able to see/detect obstacles when you bump into them. One effect of this is that with the larger field of view, you can 'passively perceive' the neighborhood around you, and learn the location of obstacles without actually having to move in that direction - the blind robot can only actively perceive, via bumping into obstacles directly. You should've seen that the passive perception robot has an advantage, as it learns about the environment faster.

But what if you had partial information? Consider the following case: when the agent is in a cell, it can sense how many obstacles are neighboring it, but not where they are. A given interior cell (not against the wall) has eight possible neighbors (N,S,E,W,NW,NE,SW,SE) - the agent may sense that three of these neighbors are obstacles (for instance), but not which three. (Cells along the borders have smaller neighborhoods but the same structure applies - the agent senses how many of the neighboring cells are obstacles, but not which.)

Clearly this is informative - knowing that there are 0 neighboring obstacles means that you can move freely in any direction. But for larger numbers of sensed obstacles, while this provides information, it is not necessarily clear cut information. However, as the agent explores and determines that cells are in fact empty, the partial sensed information can become more informative: if you know that there is a single obstacle nearby, you try to move in a given direction and discover an obstacle, you immediately know the remaining directions are clear.

So consider the following setup: build an agent that is blind folded (can tell a cell is blocked when it attempts to move there and fails, as in Project 1), but also has access to the partial sensing information. Design your agent to utilize this partial information and the complete information (of discovering cells) to try to work out and understand its environment faster. How does this hybrid agent compare in performance to your Project 1 Agents (the Blindfolded Agent and the Agent That Can See Four Neighbors)? Notice that the agent for this project has a larger 'field of view' - it can see all eight neighbors at once - but it is partially obscured in that it can't know which neighbors are blocked. Is this better or worse than seeing only the four cardinal neighbors perfectly?

Your report needs to include graphs comparing the performance of four agents on a 101 by 101 size map, at densities p from 0 to 0.33:

- The Blindfolded Agent - the Agent from Project 1 that bumps into walls.
- The 4-Neighbor Agent - the Agent from Project 1 that can see all four cardinal neighbors at once.
- The Example Inference Agent - Described Below
- Your Own Inference Agent - Design your own agent that beats the Example Agent

You will need to generate a large number of maps at each density, running each agent on every map, and average out the performances for each agent, as you did in Project 1.

Your report should also include the design description for your own agent - how you approached the problem, how you represented the knowledge base and manipulated etc, etc. You should include your code for your agent as well, well commented and documented. Be sure to include what about your agent is meant to exceed the performance of the Example Agent - for instance inferring things that the Example Inference Agent cannot, and use your data to

show that it did. Does your inference agent infer everything that can be inferred? Analyze your results - how does the performance change with density, etc. Be thorough and clear.

The Example Inference Agent: The agent can detect obstacles by running into them, but at every location can also sense how many of the (at most 8) immediate neighbors are blocked. However, it cannot sense which of them are blocked without attempting to move in those directions. For each cell x, the example agent should track the following information:

- N_x: the number of neighbors cell x has.
- Whether or not cell x has been visited.
- Whether or not cell x has been confirmed as empty or blocked, or is currently unconfirmed.
- C_x: the number of neighbors of x that are sensed to be blocked.
- B_x: the number of neighbors of x that have been confirmed to be blocked.
- E_x: the number of neighbors of x that have been confirmed to be empty.
- H_x: the number of neighbors of x that are still hidden or unconfirmed either way.

Given this information, we can infer things about cells in the following way: for any cell x, let N_x be the number of neighbors of x, C_x be the number of blocks known to neighbor x (from the partial sensing), B_x the number of blocks confirmed to neighbor x, E_x the number of empty cells confirmed to neighbor x, and H_x the number of neighbors of x that are still 'hidden'.

- If $C_x = B_x$: all remaining hidden neighbors of x are empty.
- If $N_x - C_x = E_x$: all remaining hidden neighbors of x are blocked.
- If $H_x = 0$: nothing remains to be inferred about cell x.

At any time step, the Example Agent applies the above rules to every x (potentially looping multiple times) until no more cells may be inferred.

The Example Agent follows the following workflow:

- A^* is used to determine a path from the current location to the goal, based on the current knowledge of which cells are blocked/empty and the freespace assumption.
- The Agent attempts to execute this path one step at a time.
- When the agent attempts to move into a cell x:
  - If the cell is empty, the stored info for all cells is updated accordingly. The inference agent is run until nothing new can be inferred.
  - If the cell is blocked, the stored info for all cells is updated accordingly. The inference agent is run until nothing new can be inferred.
- After the agent completes the inference: if a cell along the planned path is inferred to be blocked, or the cell the agent attempted to enter is blocked, $A*$ is run again on the current state of the board with the agent's current location to generate a new path plan.
- Repeat.