CS 470 Programming Assignment: Value Iteration

In this lab you will use value iteration to develop an algorithm that guides the robot to the goal in the code base you developed in the Bayes Filter Lab.

The code is already set up for you to do this. To run the code in "automatic" mode (the robot makes the decision instead of you), just run the client program as follows:

```
java theRobot automatic 1000
```

The first argument (`automatic`) causes the program to use the function `automaticAction()` to determine how the robot should move (as opposed to using keyboard input when the argument is `manual`). You'll need to write the function `automaticAction()` (see Task 2 below). The function should use the policies you create using value iteration to tell the robot how to act.

The second argument passed into `theRobot` is a time delay (in milliseconds) that allows you to see the robot move.

You should design your decision-maker to work with your Bayes Filter code from the previous lab. Using the Bayes Filter your robot will get beliefs about where it is, and using the value function computed by value iteration, it will determine the best action to take in order to reach the goal.

You need to test your approach in two different scenarios:

- Scenario 1: the robot's position is fully known to you

  This will be the easier scenario, where there is no uncertainty over the location of the robot. This basically takes the Bayes Filter code out of the equation and lets you focus on this assignment. (You encouraged to run with manual control in this situation to confirm that your Bayes Filter gives the correct location of the robot at all times). You are encouraged to develop/test your code with this scenario, as it will be easier to detect bugs.

  To run this scenario, with the robot's position fully known to you, run the server as follows:

  ```
  java BayesWorld mundo_maze.txt 1.0 1.0 known
  ```

- Scenario 2 – the robot's position is NOT fully known to you

  The more general scenario is when the robot's position is not initially given and your sensors and transitions are noisy, so our beliefs include a positive probability of the robot being in a large number of states. To run this scenario, run the server as follows:

```
java BayesWorld mundo_maze.txt 0.8 0.8 unknown
```

1 Tasks

1.1 Task 1: Value Iteration

In the `doStuff()` function, on the second line of code, there is a commented out call to a function: `valueIteration()`. This function does not yet exist. Your first task is to implement this function, and uncomment out the line of code. This function will run the value iteration algorithm before your robot runs to determine the value of being in each state on the current map, give the current p_m values. You will then use these values to decide what the robot should do (Task 2).

The `Vs` variable exists that you can use to store these values (you will need to create them at the beginning of the function, see `initializeProbabilities()` for an example of how this is done with the `probs`).

The value iteration algorithm repeats the following update for every state s (a state here is an (x, y) location on the map where the robot could be), until convergence:

$$V(s) := R(s) + \gamma \max_a \sum_{s'} P(s'|s, a)V(s')$$

You will have to determine the reward $R()$ to use for each state (this should depend on whether it is open (`mundo.grid[x][y] = 0`), a stairwell (which is bad (`mundo.grid[x][y] = 2`)), or the goal (`mundo.grid[x][y] = 3`). You can adjust these rewards to get the robot behavior to be what you want (i.e. there isn't a "right" answer for these values). You will also need to select a discount factor γ. Convergence can be detected when none of the values across all the different states changes by more than some ϵ, which you will also have to choose. Play around with these adjustable values until things work the way you think they should.

1.2 Task 2: Action Selection

This task requires you to implement the `automaticAction()` function that is in the code. This function will use the state values computed by value iteration to determine what action to select. You will need to implement this function in two different ways, and compare how well they work under different p_s and p_m scenarios.

1.2.1 Approach 1

The first approach will use the principle of *maximum expected utility*, where your utilities are given by the utilities derived in the value iteration algorithm ($V(s)$), and probabilities are given by your probabilities of being in each state ($(Bel(s_t)$ calculated by your BayesFilter). Then, you would want to select the action that maximizes the expected utility:

$$EV(a) = \sum_{s_t} Bel(s_t) \cdot Q(s_t, a)$$

where

$$Q(s_t, a) = \sum_{s'} P(s'|s_t, a) \cdot V(s')$$

Once you compute $EV(a)$ for each action (up, down, left, right, stay), you can have the robot pick the action with the highest expected utility, given your beliefs and value function. How well does making decisions in this way work? Try it out for different values of p_m (the probability the robot goes in the direction it intends) and p_s (the probability the robot's sonar returns the correct reading) as well as when the robot's initial position is known and unknown.

1.2.2 Approach 2

The second approach will be something that you come up with yourself. It just needs to differ from Approach 1 in some way, and needs to still use the information from your beliefs and value function. Perhaps, instead of acting optimally with respect to the robot's utilities, it could sometimes take actions that help it know where it is. You decide.

Conduct and report on an experiment comparing your own ability to get the robot to the goal (when the program runs in manual mode) with that of your two algorithms. Who's better at getting the robot to the goal (both in terms of success rate and moves required? What do you do that the robot does not seem to do (and vice versa)? How to the different approaches fare when p_s and p_m are varied?

Note that the number of moves taken to reach the goal (if you indeed reach it) are printed out by the server to the console at the end.

In addition to reporting and writing about your experiment, submit videos showing your robot moving to its goal using both of the approaches.

---

2 What You Should Turn In

Submit the following:

- Several videos showing your robot navigating to the goal (using the code you wrote) on several maps. Ideally, the video should simultaneously show both the server GUI which shows the robot's actual position and the GUI showing the output of the Bayes Filter (client GUI). Videos should be provided for multiple worlds and multiple values of `probMove` and `sensorAccuracy`. Make sure you mark which parameters were used for each video (either on the video or in a `.pdf` file).

- A `.pdf` including the code you modified/wrote for this assignment and also documenting experiments evaluating various versions of your algorithm (and comparing them to your own performance when you control the robot manually). You should also give a description of what you did for Approach 2. Also include what parameters you used in your value iteration (rewards and discount factor) and how you decided on those values.