ROB 599: Sequential Decision Making in Robotics, Winter 2015
HW #3: Planning Under Uncertainty
Due: 3/9/15 (10am)

Questions (30 points)

1) Consider the popular "tiger and pot-of-gold" problem from the supplementary course reading [1]. A robot is faced with the choice of opening two doors (left and right). Behind one door sits a tiger, behind the other door is a pot of gold, and the robot is uncertain which door is which. The robot can open the left door, open the right door, or listen to try to determine which door the tiger is behind. There is a cost of listening, and there is a chance that the robot will hear the tiger behind the wrong door. If the robot opens the door with the tiger, it receives a negative reward. If it opens the door with the pot-of-gold, it receives a positive reward. The world resets after the robot opens a door. The rewards are discounted after each action.

Use the following values for your formulation:

- +10 reward for opening pot-of-gold door. -100 for opening tiger door. -1 for listening
- Opening a door resets the tiger to a 50-50 chance of being behind either door
- When listening, the robot will hear the tiger behind the correct door with a 75% chance. It will hear the incorrect door with a 25% chance
- The discount factor $\gamma = 0.9$

a. Formulate the tiger and pot-of-gold problem as a POMDP. **Describe the states, actions, observations, and rewards.** (10 pts)
Hint: In class, we associated rewards with states. Here, you should associate rewards with performing an action while in a state. This is a more general model for POMDP and MDP reward functions.

b. **Draw the horizon 1 value function (immediate rewards) for this POMDP putting the one-dimensional belief space on the x-axis.** (7 pts)

c. Suppose you believe the tiger is behind the right door with probability 0.75, then perform the listen action and hear that the tiger is on the left. **What is the new probability that the tiger is behind the right door?** (8 pts total)

Now answer the same question in each of the following circumstances:

- Assuming you heard the tiger on the right instead of on the left
- Assuming you performed the open-right action instead of listening
- Assuming the initial belief was 0.3333 instead of 0.75 (and you heard the tiger on the left)

d. **Describe what happens to the value function as the horizon is extended. Draw a qualitative diagram of the value function as horizon increases.** You need not do the exact calculations. (5 pts)

Programming Assignment (Do not start at last minute!) (70 points)

E-mail your code for this section to the TA Matt Rueben (ruebenm@onid.oregonstate.edu).
To avoid mixup, e-mail with the following subject line and archive file name:
Subject: ROB-599 Homework 3 Submission
Archive name: (yourID)-hw3.zip
Where (yourID) is your onid ID (e.g., ruebenm-hw3.zip).

The zip file "Homework3.zip" contains helper functions and data for this homework. Download it in addition to this word document. You may use any programming language you wish to complete the homework. Helper functions are provided in the zip file if you choose to use Matlab.

In this assignment you will operate a simulated mobile robot in a maze environment. The mobile robot gains reward if it moves onto the cells in the maze with positive reward and loses reward if it moves on the spaces with negative reward. The robot has movement uncertainty, and it will sometimes end up in an adjacent cell if it tries to move in a given direction. **Note:** if the way to an adjacent cell is blocked, the robot will remain stationary instead of moving to that cell.

The maze uses the following convention for directions:
North = 1, East = 2, South = 3, West = 4
You should familiarize yourself with the helper functions before beginning:

- **create_blank_maze:** generates an empty maze
- **draw_maze:** draws a maze given a current location for the robot. Locations in the maze are labeled with their immediate rewards. You can also labels cells with any function if passed as a third variable.
- **get_reward:** gets the immediate reward for the robot being in a given cell in the maze.
- **get_start:** gets starting index for maze.

- **is_move_valid:** determines if a move (North, East, South, West) is valid for a given cell (is it blocked by a wall or not).
- **load_maze:** loads a maze from file
- **maze_index_from_XY:** returns the index of a cell given its X,Y coordinates
- **maze_XY_from_index:** returns X,Y coordinates of cell given its index
- **move_maze:** attempts to move the mobile robot in a given direction within the maze. The robot may move to a different location due to navigation uncertainty.

Step 1 (25 points): "maze0.txt" gives a maze where the top right corner provides -10 reward and the cell below that provides +1 reward. The robot starts in the top left corner and moves with some navigation uncertainty. The robot will not always move as requested; it has a probability of ending up in some other adjacent cell (noise = 0.1 = 10% failure chance). Run the file randomExplore.m with noise=0.1 and discount=0.9 to see random behavior in this environment. **Note:** the game does not end when the robot enters the -10 or +1 reward cells, it continues with the same navigation uncertainty as before.

Formulate this maze navigation problem as an MDP and solve it using value iteration. You should use a discount factor of 0.9. Once you have the value function, you should use the Q values to generate optimal behavior for the robot and write a qNavigate.m file to execute this behavior. **You should provide (a) a list of the states, actions, and rewards for your MDP formulation, (b) a screenshot of your learned value function for noises 0.1 and 0.2 printed on the maze (using draw_maze.m), (c) the final reward found by running your qNavigate algorithm for 100 iterations with navigation noises of 0.1 and 0.2, and (d) a description of how changing the noise from 0.1 to 0.2 qualitatively affects the optimal strategy.**

Step 2 (30 points): We will now examine a robotic search and tracking problem in the **"maze1.txt"** environment. In this problem, the reward "moves around", and the robot must try to find the square that contains the reward as many times as possible. The robot knows its own location, but it does not know where the reward is. However, it does receive a *perfect* observation about whether the reward is in its current cell. You can run the randomTargetSearch.m function with 0.1 noise and 0.9 discount factor to see random behavior. Notice how the cell with reward moves around. The moveTarget.m helper function moves the target around randomly. The getObservation.m helper function returns the perfect observation saying whether the target is in the robot's current cell.

For this problem, you must formulate this search and tracking problem as a POMDP and solve it in the "maze1.txt" maze using the most-likely heuristic and using QMDP. Once you have done this, you should implement mostLikelySearch.m and qmdpSearch.m functions to make the robot follow the appropriate policies. **You should provide (a) a list of the states, actions, observations, and rewards for your POMDP, (b) a description of how you implemented the motion update to the belief for the randomly moving target (you may do this in any way you find reasonable, but you should justify it in your writeup), (c) the final reward achieved after 100 iterations by your most-likely and QMDP algorithms at noise levels of 0.0 (no noise) and 0.3, (d) 4 subsequent**

“draw_map.m” frames using the most-likely heuristic at the 0.3 noise level, and (e) 4 subsequent “draw_map.m” frames using QMDP at the 0.3 noise level.

A few hints on how to get this done:
(1) You need to maintain your own belief of where the target is and update it on each iteration. This belief will need to disperse to neighboring cells (motion update) and also change based on observations (measurement update). **You will need to formulate and update the belief state for the target's position yourself. Come up with a reasonable way to do this, and describe it in your writeup.**
(2) Keep in mind that your state space is now NxN, where N is the number of cells. The target's state and the robot's state need to be considered in the value iteration. This is a larger state space and will take more time to run value iteration on. You may need to stop value iteration before it converges if it is taking a long time (this is ok). You should pre-compute the V and Q functions and store them *before* executing the behavior in the maze.
(3) Take advantage of the fact that you know the robot's location. There is only a subset of the state space that is a relevant at a given time when calculating your best action.

Discussion (15 points): Address the following question:
Compare and contrast the behaviors of the most likely and QMDP heuristics in the target search domain from the programming assignment. Which is more appropriate in this scenario? Choose one other scenario that could be formulated as a POMDP (e.g., from the course readings) and explain how QMDP would perform relative to the most likely heuristic in that scenario.

References

[1] Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains." Artificial intelligence 101.1 (1998): 99-134.