Homework #3: MDPs, Q-Learning, & POMDPs

Out on March 25
Due on April 8
Problem 1: MDPs [15 pts.]

Figure 1: MDP for Problem 1. States are represented by circles and actions by hexagons. The numbers on the arrows from actions to states represent the transition probability, for instance, $P(s_3|s_2, a_0) = q$. Not shown are arrows for actions that have transition probability 0, for instance, $P(s_0|s_0, a_0) = 0$. Each of the parameters $p$ and $q$ are in the interval $[0, 1]$. The reward is 10 for state $s_3$, 1 for state $s_2$ and 0 otherwise.

For this question, consider the infinite-horizon MDP $M$ represented by Figure 1 with discount factor $\gamma \in [0, 1)$.

a) List all the possible policies for $M$. [2 pts.]

b) Show the equation representing the optimal value function for each state of $M$, i.e. $V^*(s_0), V^*(s_1), V^*(s_2)$ and $V^*(s_3)$. [2 pts.]

c) Is there a value for $p$ such that for all $\gamma \in [0,1)$ and $q \in [0,1)$, $\pi^*(s_0) = a_2$? Explain. [4 pts.]

d) Is there a value for $q$ such that for all $\gamma \in [0,1)$ and $p > 0$, $\pi^*(s_0) = a_1$? Explain. [4 pts.]

e) Using $p = q = 0.25$ and $\gamma = 0.9$, compute $\pi^*$ and $V^*$ for all states of $M$. You can either solve the recursion of item (b) or implement value iteration. In the latter case, the error between $V^t$ and $V^*$ allowed is $\epsilon = 10^{-3}$, therefore the stop criterion should be $\max_s |V^{t-1}(s) - V^t(s)| \leq \epsilon (1-\gamma)/\gamma$. [3 pts.]

Problem 2: Q-Learning [35 pts.]

You are to implement the Q-learning algorithm. Use a discount factor of 0.9. We have simulated an MDP-based grid world for you. The interface to the simulator is to provide a state and action and receive a new state and receive the reward from that state. The world is a grid of 10 $\times$ 10 cells, which you should represent in terms of the $x$ and $y$ coordinates of each cell. And there are four possible actions, North (N), South (S), East (E), and West (W). All the actions are non-deterministic. We have provided C++ versions of the simulator (available off the homework page, in both 32-bit and 64-bit form).

The C++ interface is `my_next_state(State, Action)` and `my_reward(State)`, where State and Action are defined in `mdp-simulation.h`. Instructions for using this file are also in the `.h` as comments.

You are to hand in the learned policy in terms of a function and a Q-table that we can query if needed. You are also to show a graph with the reward gathered by successive episodes of length 100 steps starting in some random position, as learning progresses.
Problem 3: POMDP Action Models [15 pts.]

Let the initial belief state \( b_0 \) for the 4x3 POMDP world be the uniform distribution over the nonterminal states, i.e., \( \left( \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \right) \). We will calculate the exact belief state \( b_1 \) after the agent moves LEFT and its sensor reports exactly 1 adjacent wall using two possible realistic action models. For this question, you may either (a) show your calculations analytically or (b) write a small program/script to do it. If you choose option (a), make sure to show all your work. If you choose option (b), please upload your code with the solution along with a read me file on how to run your code.

1a) What is your sensor model, \( P(e|s') \)? [1 pt.]

1b) Presume the action model states that every time that an action is against a wall, nothing happens and the agent/robot stays in the same state. Thus, for example, if we are in the state \((1,1)\) and are looking only at the action LEFT, the action model is \( P((1,1)|\text{LEFT},(1,1)) = 1 \). Under this model, calculate the exact belief state \( b_1 \) after the agent moves Left and its sensor reports exactly 1 adjacent wall. [6 pts.]

1c) Presume the action model states that every time an action is against a wall, the probability of staying in the same state is additive over the possible cases. Thus, for example, if we are in the state \((1,1)\) and take the action LEFT, the probability of hitting the wall to the left is 0.8 and the probability of hitting the wall to the right is 0.1. Thus, the total probability \( P((1,1)|\text{LEFT},(1,1)) = 0.9 \). Now, under this model, calculate the exact belief state \( b_1 \) after the agent moves Left and its sensor reports exactly 1 adjacent wall. [6 pts.]

1d) In each case, which location is the highest probability location? Are the two answers the same or different? [2 pts.]

Problem 4: POMDP Grid World Navigation [35 pts.]

In this problem, we will explore using POMDPs to plan for navigating in discrete grid worlds. While you will not have to actually solve the POMDPs, if you want to try to solve them, we suggest using the zMDP solver: https://github.com/trey0/zmdp.

(1) POMDP Localization

In this problem, the map of the world is known (see Figure), but the robot’s position and orientation is unknown. The goal for the robot is to get to square S31 and “announce” that it has reached the goal. If it announces and is actually at the goal location, it receives a reward of +100; if it announces and is not at the goal, the reward is -1000. After announcing, the robot is “magically” transported to a sink state (not illustrated) where all actions leave it in that state and it gets no further reward. The robot can also move forward, turn left, and turn right. Turning left or right is deterministic. Moving forward takes it to the square in front of it, unless that square is blocked by a wall, in which case the robot stays where it is. Walls are the outside boundary of the grid and the boundaries of the gray squares, and are represented as solid black lines. The robot can also observe what is in front of it at a cost of 10. The observation tells whether there is a wall in front with 90% accuracy.

1a) Write this domain as a POMDP. Include descriptions of the states, actions, transition function, observations, observation function, and rewards. Do not forget that all actions and observation functions must be defined for all states. Use the 3 x 3 grid above as the environment that the robot needs to explore. For cases where the transition or observation functions are repetitive, you can just enumerate the functions for one set of states and point out what the pattern is for the other states. [5 pts.]
b) Assume that the robot’s initial belief is uniformly distributed amongst all the white cells in the environment, but it knows it is facing north (up, in the diagram). Assume that the robot executes the sequence (forward; right; observe; left; forward; observe). Assume the first observation returns “wall” and the second returns “open”. Show the belief states of the robot after each action. [5 pts.]

c) Show the optimal MDP policy (it is the obvious thing to do—you can easily generate it by hand). Using this policy and the assumption about the initial state given in (2) above, answer the following (in case of ties, choose an arbitrary, but fixed, tie-breaking method, and indicate which method you are using, such as choosing actions that come earlier in the dictionary): [10 pts.]

   (a) What is the sequence of actions chosen (up to the “announce”) using the “most likely” heuristic?
   (b) What is the sequence of actions chosen (up to the “announce”) using the “voting” heuristic?
   (c) If the sequences differ, explain, in general terms, why the two heuristics produce different results; if the sequences are the same, explain, in general terms, what it is about this particular domain that they produce the same results.

(2) POMDP Mapping

In this problem, the robot knows its initial starting position and orientation, but it does not know which environment it is in (out of the six environments shown in Figure 3). The goal is for the robot to “announce” which environment it is in. If it announces correctly, it gets a reward of +100; if it announces incorrectly, the reward is -1000. After announcing, the robot is “magically” transported to a sink state (not illustrated) where all actions leave it in that state and it gets no further reward. The robot can also move forward, turn left, and turn right. Turning left or right is deterministic. Moving forward takes it to the square in front of it, unless that square is blocked by a wall, in which case the robot stays where it is. The robot can also observe what is in front of it at a cost of 10. The observation tells whether there is a wall in front with 100% accuracy. As in part 1, walls (solid black lines) appear at the borders of the grid and surrounding all gray squares.

   a) Write this domain as a POMDP. Include descriptions of the states, actions, transition function, observations, observation function, and rewards. Use the six environments above as the set of environments that the robot could possibly be in. For cases where the transition or observation functions are repetitive, you can just enumerate the functions for one set of states and point out what the pattern is for the other states. [5 pts.]

   b) Assume that the robot’s initial belief is that it is in state S31 facing north (up), and it has uniform belief about which environment it is in. Assume that the robot executes the sequence (forward; right; observe; forward; left; forward; right; observe; announce environment 1 (the top left map above)). Assume the first observation returns “wall” and the second returns “open”. Show the belief states of the robot after each action. What is the expected, undiscounted reward of this sequence? [10 pts.]
Figure 3: Uncertainty over six possible worlds in which our robot might reside.