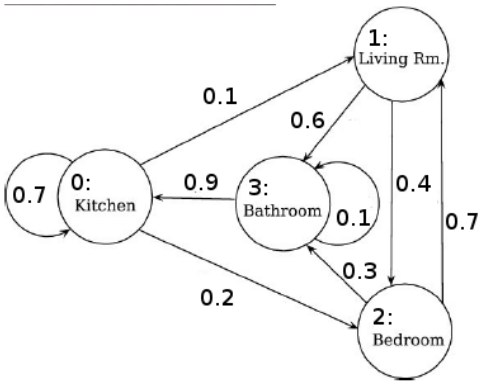
CS 4758/6758 Robot Learning: Homework 2

Due February 28 at 4:00 PM

For this homework, you have a choice between doing question 3 or question 4. You need not do both, and will not receive any additional points for doing so, but must do one to receive full credit on this HW. Only one will be graded, so please indicate which should be.

1 Markov Chains (25 points)



A ground robot is let loose in an apartment with four rooms and begins to randomly wander. After each minute passes, there is a chance it moves to a different room (assume the robot doesn't change rooms more than once a minute). So, for instance, if the robot starts in the kitchen, one minute later there is a .1 probability it moves to the living room, a .2 probability it goes to the bedroom, and .7 chance it remains in the kitchen. Using the information given in in the figue above, answer the following:

A) If the robot starts at $t = 0$ in the kitchen, what is the probability it is in the bedroom at $t = 3$? (5 points)

B) Suppose instead the robot begins at time $t = 0$ in a random room given by the probability distribution p_0. (p_0 is a four-dimensional vector; the first element of p_0 is the probability of starting in the first room, etc.). Then similar distributions $p_1, p_2, ...$ are defined for all subsequent t. Give a general expression for p_t . (Hint: Use 4x4 transition matrices at each time.) (5 points)

C) Evaluate $lim_{t->\infty} p_t$. Does it depend on p_0 ? (5 points)

Now let us suppose that the robot gets wet and useless in the bathroom, and cannot return back. So let us modify our transition matrix. (Let us number the rooms such: kitchen:0, livingroom:1, bed:2, bathroom:3.)

$$\begin{bmatrix} .7 & 0 & 0 & 0 \\ .1 & 0 & .7 & 0 \\ .2 & .4 & 0 & 0 \\ 0 & .6 & .3 & 1 \end{bmatrix}$$

D) Starting from kitchen, what is the probability that the robot will remain in a useful state after 2 timesteps? (5 points)

E) What would $lim_{t->\infty}p$ be for the rooms? (5 points)

2 Reinforcement Learning (50 points)

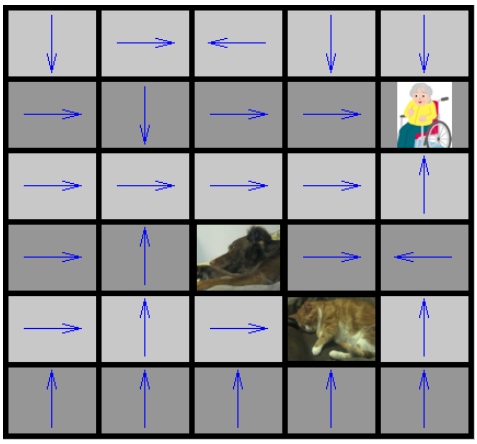
We're going to help PR2 traverse a room to reach someone in need of robotic assistance. Unfortunately, her lazy pets have decided to take a nap in the middle of the room, and we do not want PR2 to run them over!

Assume the room can be divided into a grid with 6 rows and 5 columns. For positions, we will MATLAB-style notation, ie (row,col). Starting from any location in the room, the goal is to move the robot to the old lady at (2,5). The robot can move in any of the four cardinal directions.

We will assume that the reward R(s) is a function of the current state only. Take $\gamma = 0.96$ as the discount factor for the rewards. The rewards are +100 for the goal state, and -50 for cells containing pets. For all other states, the reward is -1. When the robot hits a pet or reaches the goal state, the run is over.

Assume that the robot makes decisions every second. At every time-step, the robot must choose one of the four actions – left, right, up, or down. However, the robot's movement is not deterministic. Whenever the robot tries to move, it reaches the intended cell with a probability of 0.98, and reaches each of the grids perpendicular to the action with a probability of 0.01 (eg if the robot tries to move upwards, there is a 0.01 chance that it actually moves left and a 0.01 chance that it actually moves right.) Any actions which would move the robot outside the map leave it in the same location it took the action from, eg if it moved left from (1,1), it would be in (1,1) at the next timestep as well.

For this problem, we provide some starter code. Please use only this code and basic MATLAB built-in functions to solve the problems - do NOT use any MATLAB functions or third-party libraries directly related to reinforcement learning, value/policy iteration, etc. Your code should be mostly your own. If you are unsure of whether or not a particular function violates this, please ask the course staff.



A) The above figure shows a sample policy that the robot may take. Clearly, this is not a good policy. Run Value Iteration for this model and report the optimal policy that the robot may take. Use a convergence criterion that checks if the maximum absolute change in the value function on an iteration exceeds some specified tolerance. Please mark out the optimal policy on the grid, as shown for the sample policy above. [Implement the *solveRL* function in the starter code.] (20 points)

B) Now, our robot fell down and has lost its precision. Specifically, it does not have the same transition probability for actions. Your next task is to write a program that learns the optimal policy without knowing the transition probabilities. You may only use the state transitions observed. We provide a function that simulates the actions in the environment, and returns the next state of the robot upon an action.

Initially, the estimated transition probabilities are uniform (equally likely to end up in any other state). During the simulation, you must choose actions at each time step according to some current policy. As the program goes along taking actions, it will gather observations on transitions and rewards, which it can use to get a better estimate of the MDP model. Since it is inefficient to update the whole estimated MDP after every observation, we will store the state transitions and reward observations each time, and update the model and value function/policy only periodically. Thus, you must maintain counts of the total number of times the transition from state s_i to state s_j using action a has been observed (similarly for the rewards). Note that the rewards at any state are deterministic.

Each time a failure occurs (i.e. the robot hits the obstruction) or the robot reaches the goal state, you should re-estimate the transition probabilities and rewards as the average of the observed values (if any). Your program must then use value iteration to solve Bellman's equations on the estimated MDP, to get the value function and new optimal policy for the new model. Finally, assume that the whole learning procedure has converged once several consecutive attempts (defined by the parameter $END\_LEARNING\_THRESHOLD$) to solve Bellman's equation all converge in the first iteration. Intuitively, this indicates that the estimated model has stopped changing significantly.

How many trials(times the robot hit one of the trees or reached home) did it take before the algorithm converged? Submit your code, and print/draw out the optimal policy obtained as before. (30 points) Note: Please add meaningful comments in the code that you provide.

3 Planning Kinematic Trajectories (25 points)

Write a program which use ROS's arm_navigation stack to move the end effector of a simulated PR2 in Gazebo to a series of poses while avoiding obstacles. The tutorial at `http://www.ros.org/wiki/move_arm/Tutorials/MoveArmPoseGoal` will probably be useful.

Use the following command for launching PR2 simulation which will set the initial pose for the PR2 as well as the table object.

```
roslaunch pr2_gazebo pr2_table_object.launch
```

The set of goal positions for the right arm is given below in the format $P_x, P_y, P_z, O_x, O_y, O_z, O_w$ where P is position of 'r_wrist_roll_link' in the frame of 'base_link' and O is orientaiton (in quaternions as per ROS). Each line represents one pose in a sequence. Plot the projections of the coordinates of the end-effector and elbow to the x-y and x-z planes as it executes this sequence of poses.

(Tip: In order to visualize collision map PR2 is building from its laser sensor, add "CollisionMap" in Rviz and subscribe to "/collision_map_occ".)

```
0,-0.7,1.2,0,0,0,1
0.75,-0.2,0.8,0,0,0,1
0.5,-0.35,0.35,0,0,0,1
```

4 Reinforcement Learning (25 pts.)

In this problem, you will show that policy iteration is guaranteed to find the optimal policy.

Notation:

Consider an MDP with discrete sets of actions A & states S and a discount factor $0 \leq \gamma < 1$.

V^π is the value function for the policy π. It is is the solution to the Bellman equation:

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s')$$

The corresponding Bellman operator $B^\pi(V)$ is defined as:

$$B^\pi(V) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V(s')$$

(Notice that $B^\pi(V^\pi) = V^\pi$)

Similarly V^* is the value function for the optimal policy π^*. $\forall s, \pi\ V^*(s) \geq V^\pi(s)$. It is the solution to:

$$V^*(s) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^*(s')$$

And the Bellman operator $B(V)$—again defined so that $B(V^*) = V^*$ —is:

$$B(V) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P_{sa}(s') V(s')$$

Questions:

(a) Prove that if $\forall s \in S\ V_1(s) \leq V_2(s)$, then $\forall s \in S\ B^\pi(V_1)(s) \leq B^\pi(V_2)(s)$.

(b) Prove that for any finite-valued V,

$$\|B^\pi(V) - V^\pi\|_\infty \leq \gamma \|V - V^\pi\|_\infty$$

where $\|V\|_\infty = \max_s |V(s)|$.
Hint: Use the fact that for $\alpha, x \in \Re^n$, if $\forall i\ \alpha_i > 0$ and $\sum_i \alpha_i = 1$, then $\alpha^T x \leq \max_i x_i$.
[Intuitively, this means that applying B^π to any V brings it closer to the value function for π, V^π. Therefore, informally speaking, applying B^π an infinite number of times—$B^\pi(B^\pi(B^\pi(\dots B^\pi(V)\dots)))$—will result in V^π.]

(c) We say that V is a fixed point of B if $B(V) = V$. Prove that B has at most one fixed point, and thus that V^* is unique. Use the fact that $\|B(V) - V^*\|_\infty \leq \gamma \|V - V^*\|_\infty$, which can be proven in the same way as part (b).