CS 4758/6758: Robot Learning: Homework 3

Due: March 15, 2012 (In class)

1 Reinforcement Learning (50 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 $\forall s \in S \; B^\pi(V_1)(s) \leq B^\pi(V_2)(s)$. (10 points)

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

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

where $\|V\|_\infty = \max_s |V(s)|$. (10 points)
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—$B^\pi(B^\pi(B^\pi(\ldots B^\pi(V)\ldots)))$—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). (10 points)

(d) Recall that in each step of policy iteration a new policy π' is chosen by:

$$\pi'(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^\pi(s')$$

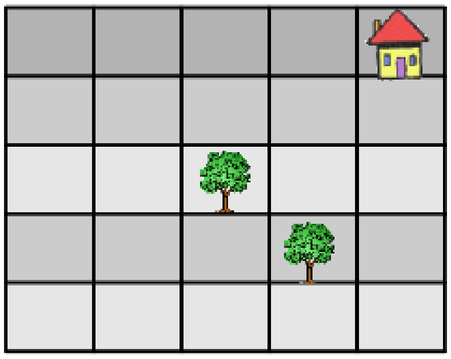
where V^π is the value function from the previous iteration. Show that π' will never perform worse than π. That is, show that $\forall s \in S, V^\pi(s) \leq V^{\pi'}(s)$. *Hint:* First prove that $V^\pi(s) \leq B^{\pi'}(V^\pi)(s)$, then use parts (a) and (b) to show that $B^{\pi'}(V^\pi)(s) \leq V^{\pi'}(s)$ (10 points)

(e) Finally, show that $V^\pi(s) = V^{\pi'}(s)$ if and only if $V^\pi(s) = V^*(s)$. Parts (c), (d) and (e) together prove that policy iteration always improves the current policy, until it converges to the unique optimimum solution. (10 points)

2 Aerial Robot: Finding its way to home (50 pts)

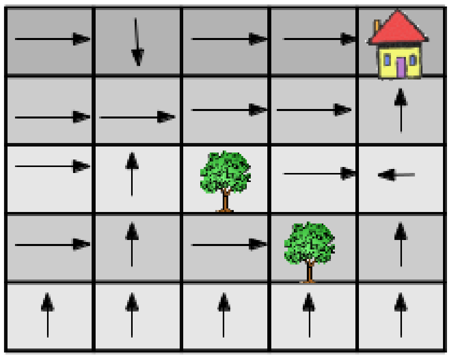
We are going to help an aerial robot traverse its way in a park to reach back home. There are 2 trees which it must avoid, or else it will fall down. Starting from any point in the park, the goal is to fly the robot to the top-right grid, marked by home in the figure.

Assume that the park can be divided into a 5x5 grid, as seen from above. The robot always moves at a constant vertical height. It can have the following horizontal motions that takes it to the other grids: left, right, top, bottom.



We will assume that the reward R(s) is a function of the current state only. Take $\gamma = 0.97$ as the discount factor for the rewards. The rewards are $+10$ for the goal state, and -5 for the trees. For all other states, the reward is -0.1. When the robot hits the obstruction or reaches the goal-state, the flight 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 a robot tries to move into another grid, it reaches the intended grid with a probability of 0.98, and reaches any of the grids perpendicular to the action with a probability of 0.01.



(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. [Starter MATLAB code provided, implement *solveRL* function] (20 points)

Now our robot has fell down a lot of times and has lost its precision. Precisely, 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.

(b) 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.