91.451, Robotics II
Spring 2005, Prof. Yanco

Lab 6: Reinforcement Learning

Out: Thursday, 7 April 2005
Due: Thursday, 14 April 2005

Overview: In this lab, you'll explore reinforcement learning (RL) using the symbolic simulator (a grid world).

Documentation for the RL Code:

There are two ways to run the RL code:

1. From the pyro interface:

   start pyro
   select SymbolicSimulator as the simulator
   for world: change the *.world to *.py then select RLWorld.py
   use SymbolicRobot60000.py
   use brain /usr/local/pyro/plugins/brains/RLBrain.py

2. From the command line:

   ```
   pyro -s SymbolicSimulator -w RLWorld.py -r SymbolicRobot60000.py -b RLBrain.py
   ```

   In an easier to read size:

   ```
   pyro -s SymbolicSimulator -w RLWorld.py -r
SymbolicRobot60000.py -b RLBrain.py
   ```

In this world, the following are the values that you can obtain:

robot/location:	Current position within the grid
robot/path:	Path taken thus far (this run only)
robot/util:	Utility grid
robot/obstacles:	List of tuples indicating areas that cannot be entered
robot/goal:	Final location we're looking for
robot/home:	Initial grid location
robot/final:	All final states (goal and pits)
robot/complete:	Have we reached a final state yet?

These are the movement commands:

up:	Move the robot up
right:	Move the robot right
down:	Move the robot down
left:	Move the robot left
start:	Move back to start point
reset:	Move back to start, restart the world (gets new pits/goal and reset utilities)
td:	Compute temporal difference (MUST BE AT GOAL STATE!)

Lab Exercises:

1. To start, load the simulator, RL world and RL brain. Run the program to watch how the system learns paths.

2. You may notice that once the robot reaches the goal the first time, that path is more likely to be followed in later trials, even if there is a shorter path. You'll see in the code that a random move is made 20% of the time. Change this percentage to several different numbers, noting the changes in the learning behavior as the random move percentage increases. Is there a range that seems to be better for finding multiple paths?

3. As we discussed in class, learning consists of exploration and exploitation. Modify the code to favor exploration. For this, you'll need to keep track of the number of times that you've visited a grid location. Once you've done this, when a random move is selected, move to the adjacent grid location that's been visited the least. Does this help to find shorter paths? You might want to change the random move percentage as well, to experiment with this.

4. Now that you have the ability to explore states that have not been selected often, let's allow the program to have a exploration and exploitation rate that changes over time. Initially, you will have a high exploration rate and a low exploitation rate. After some number of learned paths (say, 5-10), reduce the exploration rate and increase the exploitation rate. Explore the best way to change the percentages and discuss your findings in the lab write up.

5. The reinforcement value is always the same when the goal is reached. Modify the code to change the reinforcement value based upon the length of the path taken to reach the goal. Does this help to find the shortest path?

For this lab, turn in explanations of what you did for each part, as well as the results you obtained. You can either segment out the code for each part, or can turn in a single print out of the code. If you turn in a single print out, be sure to comment in the code to point out the code that was changed for each part of the lab.