EGN4060C: Reinforcement Learning

Dr. Gita Sukthankar
gitars@eecs.ucf.edu

Demo complete by Oct 16; writeup due Oct 17, 2014

In this assignment, you will be implementing a reinforcement learning algorithm called Q-learning that we discussed in class. You can find a more comprehensive discussion of Q-learning at `http://webdocs.cs.ualberta.ca/~sutton/book/ebook/node65.html`. You only have to implement and demonstrate the simulated version of your robot moving on the grid world map that you used for Lab 3, and don't need to use the actual robot at all for this lab. A starting file for this lab, QLearning.java, has been included in the jar file which may be helpful for getting started.

Grid World

You will be learning and planning on an occupancy grid, a discretized map version of the simulated world that your robot will later use. Squares on the map should either be marked as occupied or unoccupied. The robot will try to find a path through the unoccupied squares from its starting position to a designated goal position.

Design a simple grid layout (about 10x10 in size) with some obstacles. Assume that 4-square connectivity exists (up, down, left, and right); no diagonal moves are allowed. For this assignment you will have a stochastic world. In the deterministic world like you used for the A* planning assignment, actions taken always occur (moving up always takes the robot up). In the stochastic world, you should have a *transition* function that moves the robot; a reasonable one would be to have an action take the robot where it wants to go with probability 0.6, 0.1 probability of moving in each of the other directions, and 0.1 probability of remaining in place. Make sure that the agent can never accidentally wander into occupied regions.

Demonstrate the robot moving across the MapGUI visualization (used in Lab3) in the stochastic world using a fixed policy (e.g. always moving to the right). [1 pt]

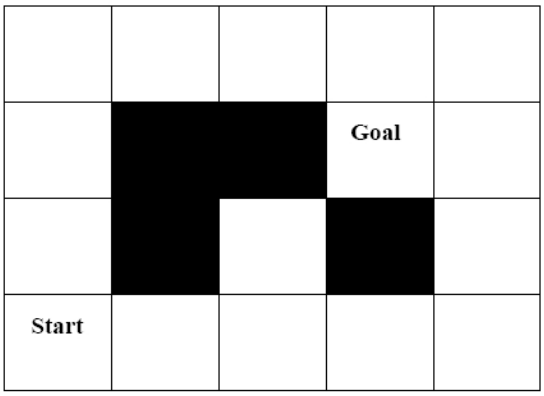

Figure 1: Small example occupancy grid—try increasing the size to 10x10

Q-Learning

Implement an agent that uses Q-learning to learn how to navigate the stochastic version of this grid world to reach a goal (high reward area). To indicate that a location is a goal, the agent should receive a high reward (e.g., 100) for reaching that location and small penalties (e.g., -1) for reaching any other square. The agent should explore the map in episodes; an episode should end if the agent either reaches the goal or the episode lasts beyond a fixed number of actions (e.g. 150 actions). Episodes should start at different starting locations to ensure that the robot explores the maze. During the learning period, the agent should use a random exploration policy to select its actions; this means that at every time period during the learning, the robot should randomly select an action. Note, that after it selects an action, the consequence of this action are determined not only by the robot's choice but also by the stochastic world dynamics. Please demonstrate 10 episodes of the (simulated) robot using your random exploration policy to the TAs. [2 pts]

After taking an action, the robot should update its Q-table, using the Q-learning rule. The Q-learning steps are shown in Algorithm 1. Try modifying the values of α, γ, the max episode length, and the number of training episodes to see how it affects the performance. Note that α and γ should should always be between 0 and 1 (usually in the range of 0.5 to 0.99).

Once your agent has finished learning, test to make sure that it can reach the goal from two different locations in the world using the learned Q-table and a max exploration policy such that the agent always selects the highest valued action for the current state. Note that the same Q-table should continue to work for both the locations in the world. Please demonstrate the (simulated) robot using the learned policy to reach two different goals in the world. [2 pts]

Please include the following for your writeup:

Algorithm 1 Q-Learning Algorithm

Define $\alpha = 0.65, \gamma = 0.99$
repeat
  Initialize s to robot or random location
  repeat
    Choose an action a from state s using random exploration policy
    Take the action a, observe reward $r(s')$ and next state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r(s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$
  until s reaches goal or max episode length
until number of episodes reached (e.g. 150 steps)

- the transition and reward functions used in your world
- a description of the exploration policy
- description of the Q-learning rule implementation
- the final Q-table (post-learning) in a 3x3 zone surrounding the goal
- the learning and training parameters (number of episodes, α, and γ)
- two example action sequences and the resulting robot movements generated using your learned Q-table from two different initial positions.