CAP 6671: Robotics Assignment

Dr. Gita Sukthankar
gitars@eecs.ucf.edu

Due Mar 28th

In this assignment, you will be designing a physically-embodied agent (aka a robot) that can learn and plan.

You can implement this in any language (or combination of languages). You are allowed to work in teams of 2-3 students; please use the webcourses chat/roster functionality to find appropriate teammates. To make things easier for single person teams, single student teams do not have to use a robot simulator to demonstrate their planner but only need to show a video/demo that their planner works in a grid world.

If you experience problems with the implementation details on this assignment, you may contact my graduate student Bulent Tastan (`bulenttastan@gmail.com`) to ask for help.

Please submit your code, a short video of your robot demo, and a 1-2 page writeup describing the key parts of your implementation along with the answers to any of the questions.

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 both a deterministic world and a stochastic one. In the deterministic world, 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.

For debugging purposes, it is useful if you have a visual representation of your world and the ability to load different map configurations. There are no points for this section of the assignment but you will need to have this for the subsequent sections.

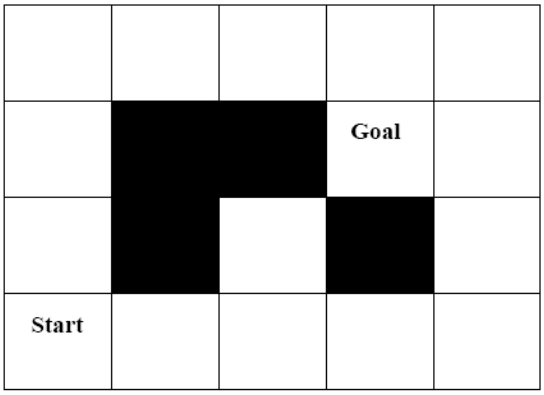

Figure 1: Small example occupancy grid

Q-Learning (10 pts)

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. 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 length of time (e.g. 150 steps). Episodes can start at different starting locations. During the learning period, the agent should use an exploration policy to select its methods; you can use any of the methods we discussed in class (e.g. greedy, softmax, exploration bonus). Whenever the agent takes an action, it should update its Q-table, using the Q-learning rule. Try modifying the values of α, γ, and the number of training episodes to see how it affects the performance.

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.

Please include the following for your writeup:

- 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)
- the learning and training parameters (number of episodes, α, and γ)
- two example action sequences generated using your policy.

A* Planning (5 pts)

For the deterministic version of the world, implement an A* planner. You should be able to plan from different starting positions to different goal states. For the admissible heuristic ($h(x)$), you should use the Euclidean distance metric.

For your writeup include:

- pseudocode of your planner
- f, g, and h values for one example map

Robot Planning Demo (5 pts)

Demonstrate that your robot can follow the path generated by the A* planner using one of the following simulator options. Single person teams only need to show a demonstration or video of their planner correctly navigating two different paths in their own grid world simulation. Multi-person teams should be able to show that they can follow paths in a continuous world.

Simulator options are:

- Teambots: A very simple Java-based simulator designed for Robocup. `http://www.teambots.org/`

- Microsoft Robotics Studio and Visual Simulation Environment: `http://msdn.microsoft.com/en-us/robotics/aa731520.aspx` A sample project for this is available in webcourses. Programming for this option can be done in C# or the VPL.
- simulator/game of your own choice (must be approved)

Note that your planner doesn't have to be integrated with your robot; you can have the path stored in a file that is read by the the robot. Submit a video of your robot (or show me a demo) for two different start and goal positions.

Bonus Points

1-3 bonus points will be awarded for particularly good implementations.