Computational Robotics, Fall 2019

Lab 1 - Slide Supplement

Name: Andrew Choi, UID: 205348339

Github Link: **https://github.com/QuantuMope/ucla-computational-robotics**

1 Preliminaries

1. Andrew Choi was the sole collaborator of this lab in terms of report and code.

2. As the sole collaborator for the lab, a more complete analysis of the problem statement, mathematical setup, and results can be found in the original lab report. Below is a brief one page overview.

2 Problem Statement

In this lab, Markov Decision Processes were used to solve a path planning problem consisting of a simple discretized robot in an 8x8 grid world. As the grid world contains a finite state space $s \in S$ with well-defined transition probabilities P_{sa} and rewards $r \in R$, dynamic programming methods were employed to create optimal control policies. Four different scenarios were explored with varying goal states and stochasticity.

3 Mathematical Setup

As the problem discussed above satisfies the Markov property and consists of discrete time decisions, the problem was framed as a Markov decision process. Furthermore, to be able to measure the favorability of a certain state, a value function $V^{(}s)\ \forall s \in S$ was used which is the expected sum of discounted returns for every state under a given policy. To compute value functions and policies, two dynamic programming methods were used, policy iteration and value iteration. Both employ the Bellman operator to obtain new estimates of the value function by using the previous value function estimate as shown below.

$$\text{loop } V^*_{t+1}(s) = max_a \sum_{s'} P_{sa}(s')[r(s,a,s') + \gamma V^*_t(s')] \tag{1}$$

$$\text{stop when } \|V_{t+1} - V_t\| < threshold \tag{2}$$

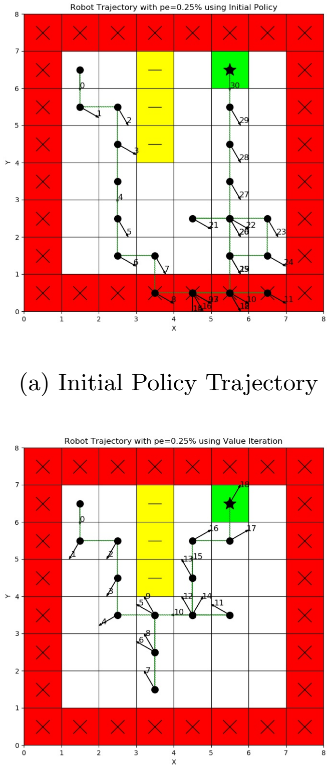
$$V^*(s) = V_{t+1}(s) \tag{3}$$

$$\pi^*(s) = argmax_a V^*(s) \tag{4}$$

Although policy iteration and value iteration differ in their methodologies, both achieve the same result of convering towards the optimal value function $V^*(s)$ from which an optimal policy $\pi^*(s)$ can be derived. A more in-depth formulation of both policy iteration and value iteration can be found in the mathematical formulation section of the lab report.

4 Results

For all scenarios, both policy and value iteration successfully computed robust control policies that allowed the robot to arrive at the goal state with maximized total reward. Even during scenarios in which significant stochasticity was introduced into the system and/or the goal state definition was narrowed, both methods were still able to compute robust policies that completely avoided negative reward states. No difference in performance was seen from either method which is to be expected as they mathematically achieve the same thing. It was seen that value iteration was faster by about 40%. Below, the advantages of an optimal policy can clearly be seen when compared to a trajectory generated from a crude hand-engineered policy.



(a) Initial Policy Trajectory

(b) Value Iteration Trajectory

Figure 1: Trajectory Results for Stochastic Environment with $p_e = 25\%$