Advanced Robotics: HW3 Write-up

Name: Mandar Deshmukh
Student ID: made2806

Part 1: Setup

Completed

Part 2.1: Policy Iteration

1. Try a few different values of γ for your policy iteration algorithm (γ = 0.5, 0.75, 0.9 and 0.99 may be good choices). Does your policy change based on the value of γ? Why is this the case? Why not?

Ans. No, the policy does not change with the value of gamma due to the following observations:

1. On simple Gridworld 0, all gamma values achieved optimal performance, that is, the reward was 10. But, on complex Gridworld 1 with obstacles requiring longer paths, only gamma=0.5 and gamma=0.99 reached optimality. Gamma=0.75 and gamma=0.9 converged to suboptimal policies (9.00). This means that optimality depended on problem complexity.
2. Higher gammas converged fastest but didn't achieve optimality. For gamma=0.9 converged in only 6 iterations on Gridworld 0, faster than gamma=0.99 (8 iterations). But it was achieved on a suboptimal policy. The faster convergence was a result of discounting values propagating less far, and therefore requiring fewer iterations to stabilize.
3. The value functions change significantly with gamma, which means increasing smoothness and broader value propagation for higher gamma, but the deterministic policies remained the same across all gamma values for these simple gridworlds

Therefore, for problems requiring long-term planning, γ should be close to 1.0. Lower gamma values may converge faster, but risk suboptimal policies in complex environments.

Part 2.2: Value Iteration

1. Are your policy and value iteration results the same? Do you expect them to be?

Ans. Yes, both the results are the same. They both achieved a reward of 10 and found the same optimal policy with 7 steps to the goal. I expected it to be the same because both value and policy iteration are guaranteed to converge to the same optimal solution, the difference being just using two different ways of solving the Bellman Optimization equation.

2. Compare the runtime of your policy and value iteration implementations. Which runs faster? Why?

Ans. My implementations got the following results:

Gridworld 0-
Value Iteration: 0.007 seconds
Policy Iteration: 0.016 seconds

Gridworld 1-
Value Iteration: 0.006 seconds
Policy Iteration: 0.011 seconds

Value iteration was about 2 times faster than policy iteration since value iteration has cheaper per-iteration computations, which is simply updating each state's value by taking maximum over actions. Policy iteration fully evaluates a policy before improving it, making each iteration computationally expensive.

In this context, Value Iteration runs faster despite policy iteration needing fewer iterations to converge, as it has an advantage in per-iteration speed that outweighs policy iteration.

But, in large state spaces policy iteration would probably become faster than value iteration due to its convergence with fewer iterations, compensating for higher per-iteration costs.