National Taiwan University
Department of Electrical Engineering
Name: Pei-Fan, Liu
ID: R13921A02

113-1 (Fall 2024)
Reinforcement Learning

Assignment 1 Report

Q1. What methods have you tried for async DP? Compare their performance.

First method I've tried is in-place value iteration. This method updates value function of each state asynchronously:

$$v(s) \longleftarrow \max_{a \in A}(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v(s'))$$

This method traverses every state-action pair to find optimal values. It takes 1056 steps for this method to converge in the provided grid world.

The second method I've tried is prioritized sweeping. This method gives each update a priority. At each iteration, only the state-action pair with the largest priority are updated. The priority gives as follow:

$$|\max_a \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V(s') \right) - V(s)|$$

Only the updates with largest priority in each iteration are considered. It takes 460 steps to converge.

My novel method focuses more on agent interactions. In standard prioritized sweeping, agent will start from every state and update its action pairs to the priority queue, and only the largest pair is updated at each time. In my method, the state-action pair with top n largest priorities are updated at each time. It only takes 386 steps for this method to converge.

Q2. What is your final method? How is it better than other methods you've tried?

My final method is the n-step prioritized sweeping method mentioned above. Different from standard prioritized sweeping, it applies the updates with top n largest priorities. In this way, urgent states can be updated more efficiently.

Also, I observe that it is redundant to start the episode for each state. Starting from each state, the agent will explore multiple state paths to find optimal value. Once each of value function of the state on the path converges, the looping process terminates. We can mark each of the state as "converged", so there's no need to start from these states to do episodes again.