EEC-289A Reinforcement Learning
Homework #3

Jonathan Dorsey
https://github.com/JonnyD1117/EEC-289A-RL/tree/main/HW

April 21, 2021

Runtime

Note that the approximate runtime for all code submitted in this assignment was approximately 34ms.

Policy Iteration Algorithm

The **Policy Iteration Algorithm** is a dynamic programming algorithm which cycles between phases of policy evaluation and policy improvement to eventually obtain the optimal policy for the Markov Decision Process (MDP). Specifically, given some initial policy, this algorithm will fully evaluate the value function (for that policy) to convergence, and then it will use that value function as a way of generating an improved policy that preforms, at least, as good as the initial.

With the probability of head $p = 0.9$ ...

Optimal Policy

$$\pi^* \approx \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 8 \end{bmatrix}$$

Optimal Value Function

$$V^* \approx \begin{bmatrix} 0 & 13.9266 & 14.5852 & 13.7695 & 12.79 & 11.7923 & 10.7926 & 9.7926 & 8.7926 & 7.7927 \end{bmatrix}$$

Value Iteration Algorithm

The **Value Iteration Algorithm** is a dynamic programming algorithm which determines the optimal value function, by taking the maximum valued action during each iteration of the value function loop. Under the action of updating the current value (at a given state) with the maximum value possible for all feasible actions in that state, the value function is effectively improving itself to acheive the largest value function possible under the dynamics of the MDP. Since the optimal value function is the one which maximizes the value across the entire MDP, once the value function has coverged, we know that in order to obtain the optimal policy we only need to iterate over each state and select the $argmax()$ for each feasible action from the current state, and use that action to develop a deterministic optimal policy. Since Value Iteration does not accept any initial policy, but rather updates the value function implicitly through maximized updates, Value Iteration removes the necessity to evaluate intermediate policies and generate intermediate updated policies from the resulting value functions. Effectively this means that Value Iteration typically will converge to the optmal solution faster since it is constantly updating the value function estimates without the need for extra step of policy evaluation.

With the probability of head $p = 0.1$ ...

Optimal Policy

$$\pi^* \approx \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 2 & 1 \end{bmatrix}$$

Optimal Value Function

$$V^* \approx \begin{bmatrix} 0 & -0.9979 & -1.9792 & -2.88 & -3.792 & -4.00 & -4.8 & -5.6 & -5.92 & -6.128 \end{bmatrix}$$