EEC-289A Reinforcement Learning
Homework #1

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

April 6, 2021

ε-Greedy

Algorithm Implementation

The ε -Greedy algorithm was implemented as close as possible to the given algorithm in the course text.

1. Initialize any parameters and constants
2. Iterate over the total number of trials (2000)
3. Sample actual rewards for k = 10 problem from Normal distributions
4. Initialize action values (Q) and number of action selections (N) to arays of zero
5. Within this outter loop iterate over the total number of time steps (1000)
6. Sample *varepsilon* from a uniform distribution from 0 to 1
7. With probability $p = 1 - \varepsilon$ select the maximal action (greedy)
8. With probability $p = \varepsilon$ select a random action
9. Compute reward for given action

10. Update action values (Q) and number of actions selections (N)

11. Collect reward data over all iterations and plot

Upper-Confidence-Bound

The only material difference between the **UCB** algorithm and the ε **-Greedy** algorithm, is the action section. Instead of select the action according to a value of ε evaluated at each iteration of the loop by drawing a sample from a uniform distribution, the **UCB** algorithm leverages the action value update law by considering the amount of times an action has been selected as well as the current timestep. As we sample from the actions, the more often that certains actions are selected via UCB, the lower effective-variance the action value will be. This can be seen by the fact that as the number of action selections (for a particular action) goes to infinity, the quantity in the squareroot will trend to zero, which would effectively state that the variance in the value associated with that action is zero.

$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

By implementing this as the means of action selection (instead of stoichastic sample used in ε - Greedy), we can obtain the UCB algorithm.

Reproducability

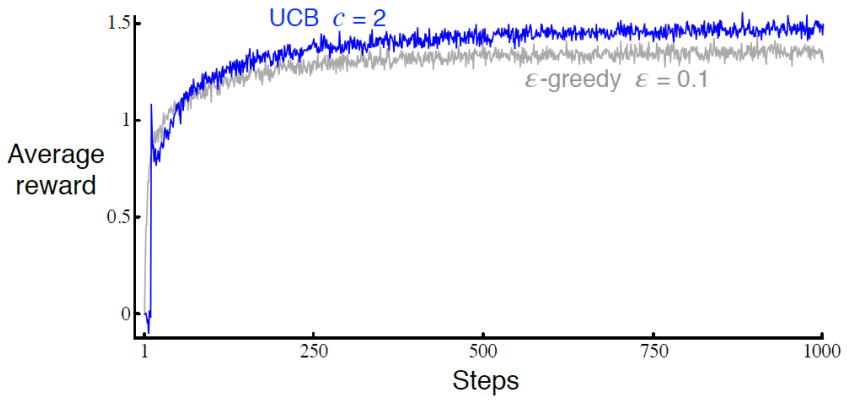
To the best of my knowledge, I had no issues with reproducing the results shown in the textbook, within numerical precision. As mentioned in the text, there is a noticeable spike in reward at the 11th time step, which is faithfully reproduced in the plots I generated vai my own implementation.

Figures

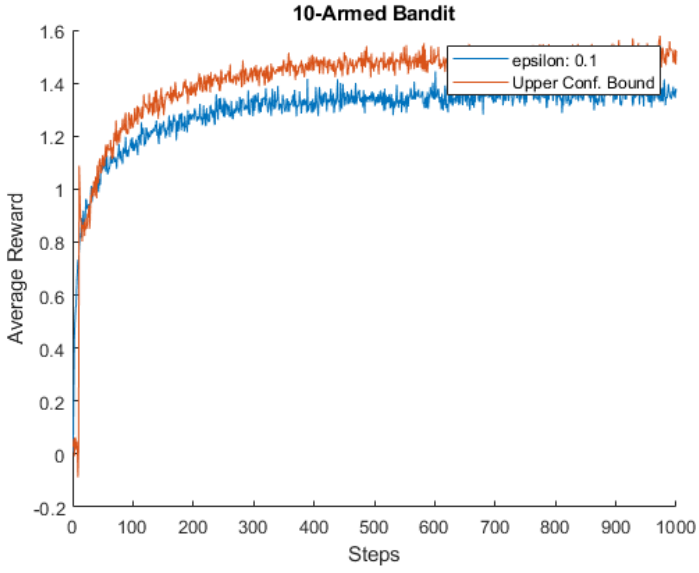
Both the Figure 2.4 from the Sutton & Barto text and the equivalent generated plots from my personal matlab code are shown below.

Program Run Times

- ε-Greedy: 24 seconds
- Upper-Confidence-Bound: 30 seconds



(a) Fig: 2.4 from Sutten & Barto



(b) Generated Results from Matlab Code