ROB 537
Learning Based Control
HW#3 Search and Optimization
TA: Shauharda Khadka

Eric Klinkhammer

October 30, 2017

1 Problem 1: Action-Value Learning

My solution to the N-armed bandit problem was an action-value table, mapping an action to a value (for code reuse, the key is actually an action and a default state).

I used a learning rate of 0.1, and report the average of 50 statistical runs of the current total reward at that time step for 10 and 100 time step episodes. The epsilon greedy variant selected a random action with probability 0.1. If there was a tie for which action had the greater reward, one was chosen at random. The table was initialized with a value of 1.

Figure 1: Action-Value Learning Performance per learning step for N-Armed Bandit

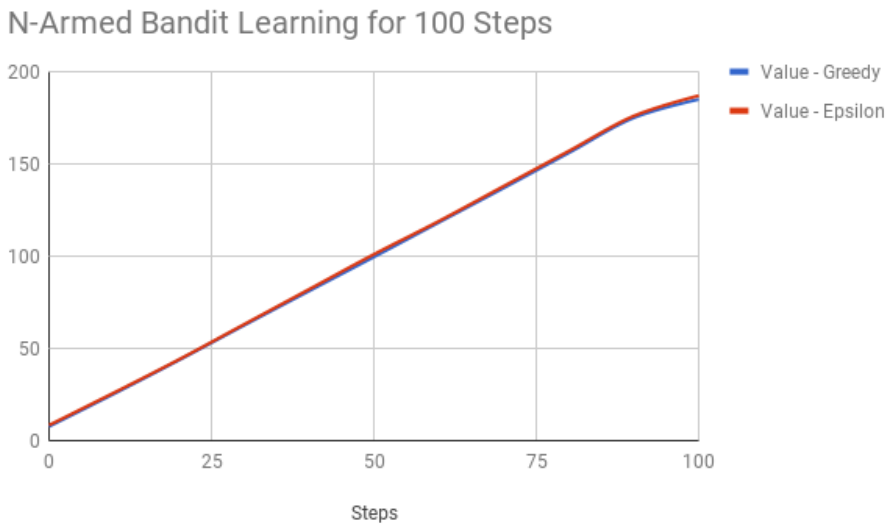

Figure 2: Action-Value Learning Performance per learning step for N-Armed Bandit

The results did not change substantially. Both the greedy and epsilon greedy approach worked.

2 Stateless RL for Problems with State

The algorithm did not perform because there is no meaningful mapping from action to value. The agent can receive a reward for any action (except left) depending on its starting location. However, the reward is dependent on the random (and unknown to a stateless representation) initial state.

3 Q-Learning in Gridworld

For my implementation of Q-learning, I used a variable learning rate of 0.1 for the first 1000 epochs, and then a learning rate of 0.05 for the subsequent 4000. My Q Table was initialized to 1s. The discount factor was 0.9. Epsilon was 0.1. For 100 statistical runs, I report the average value of a 20-step episode at that epoch. The value plotted is the average of the surrounding points.

The key difference was that this approach accounts for state. This is crucial for a domain in which the reward is dependent on state.

The algorithm did not sufficiently learn - you would expect a convergence to a value around 95. The reason the results look substantially worse than that is that exploration through the full state space is inefficient, and some statistical runs simply never reach the goal. Moreover, because Q-Learning only propagates one step, not only does the correct state and action have to be found, but the nearby states have to be explored after the reward of the goal state is determined. Unfortunately, this is typically not the case, so subsequent exploration of the path requires that the nearby states be randomly selected (with the epsilon greedy). This dependence on randomness (to escape the local minimum) is the reason for the slow convergence.

A value iteration method (instead of an exploration-based method) converges within 2 iterations to a good policy, and within 10 to an optimal policy. Results for value iteration also reported as an average of 100 statistical runs, and with a constant learning rate of 0.1 and discount factor of 0.9.

Figure 3: Q-Learning in Gridworld

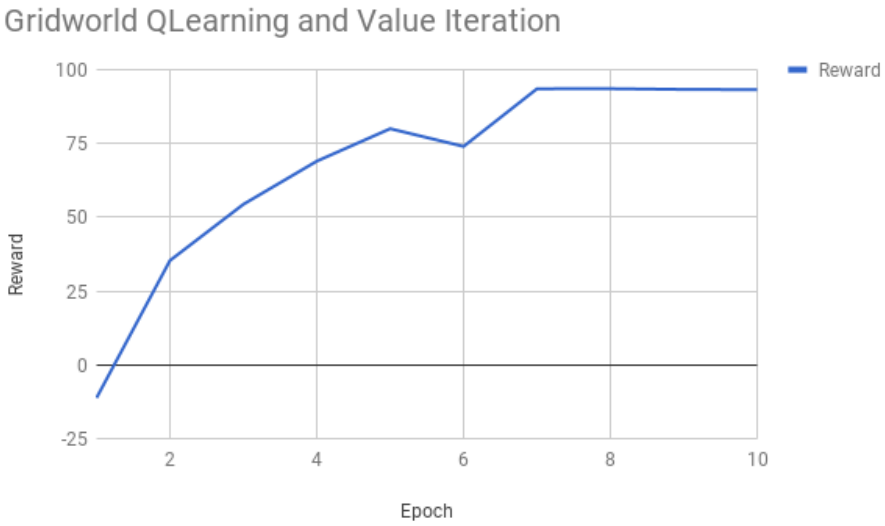

Figure 4: Value Iteration with QLearning