Dylan Shadduckk
913902897
EEC 289A Lai

Programming Assignment 1

Program Run Time: About 1 minute

Introduction

The reinforcement learning techniques explored in this assignment are the Epsilon-Greedy (e-greedy) and Upper-Confidence-Bound (ucb) methods. Both of these methods are trained on the same environment known as the 10-Armed-Testbed with the same goal of optimizing the total reward.

Environment Definition

The 10-Armed-Testbed is a learning environment in which there are 10 possible actions to choose from. Each of these actions has a true value for the reward that is decided by a normal distribution with a mean of zero and a variance of one. The reward received for choosing a given action is also determined by a normal distribution with a mean of the randomly assigned true value and also a variance of 1. This way, each time our agent chooses a certain action it will not know the true value of the reward, but must estimate it given the randomness of the reward received for choosing that action.

The true values for each of the 10 actions are determined by the normrnd function which takes a mean, variance, and array dimensions as inputs and outputs an array of random numbers based on the normal distribution and the given parameters. This gives us a 1 by 10 array of true values for each of the 10 actions.

Epsilon-Greedy Model

This model is essentially a greedy learning algorithm with some element of randomness as described by the epsilon value. This model is implemented by first initializing a 2 by 10 array where the first row is where the model will predict the expected value of a given action and the second row is the number of times that action was chosen. Each time an action is chosen, we receive a reward for that action and then we must update our estimation of the expected value for

that action given by the equation $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$. Where Q_{n+1} is the new prediction, Q_n is the old prediction, n is the number of times the action has been chosen, and R_n is the reward for that action.

With an updated expected value table, we should be able to better predict the expected value of that action and thus make choices in the future that are better optimized for the maximum reward. When the agent chooses an action the next time there are two options, the agent will either choose the action that has the maximum predicted true value or will choose a random option. The probability of choosing the random action is set by the value of epsilon. In this example epsilon is set to 0.1 or in other words, 10 percent of the time our agent will choose a completely random action. This allows us to eventually sample all the different actions and get a feel for which action will produce the highest average reward.

This random action selection is handled by the rand function. When this function produces a value less than or equal to epsilon, a random action is chosen. Otherwise, we use the max function to return the index of our expected value array with the maximum expected value. Once the action is chosen the reward is found by using the normrnd function and setting the mean to the true value of our action and the variance to 1. This reward value is then used to update our estimate of the expected value and is stored in a reward avg array that takes the current average of all the rewards thus far. The reward average array is used to plot the average reward over time in my recreation of figure 2.4 from the textbook.

Upper-Confidence-Bound Model

This model follows a similar set up to the epsilon greedy model. We use the same true values as before and we initialize the same expected value 2 by 10 array. The updating of the expected value is also the same following from the equation in the section above.

The key difference between this model and the e-greedy one is the method used for selecting the optimal action. The formula used to determine the optimal action is defined as $A_t = max(Q_t(a) + c\sqrt{\frac{log(t)}{N_t(a)}})$. Where A_t is the optimal action, $Q_t(a)$ is the estimated value for an action "a", c is a tuning parameter that determines how much the model explores, t is the current time step, and $N_t(a)$ is the number of times action "a" has been performed. This

method for selecting the optimal action places a lot of emphasis on exploration at the beginning and less as the model continues to train.

The formula in the paragraph above is used to select an action. Once the action is selected the corresponding reward is found using the normrnd function where the mean is the true value of the action and the variance is one. This value is then used to find the average reward at the current time step and then appended to the average reward array. The reward value is also used to update the expected value array so that we can make better informed decisions with each run of this game.

Results

Both the e-greedy model and the ucb model were designed to run with 1000 steps and calculate the average reward across each time step. Since this average reward is heavily influenced by the randomly generated true values of each action, I ran this experiment 500 times and computed an overall average where each experiment plays the 10-Armed-Testbed game for 1000 time steps. After the overall averages were calculated for both methods I plotted the results as shown in figure 1.

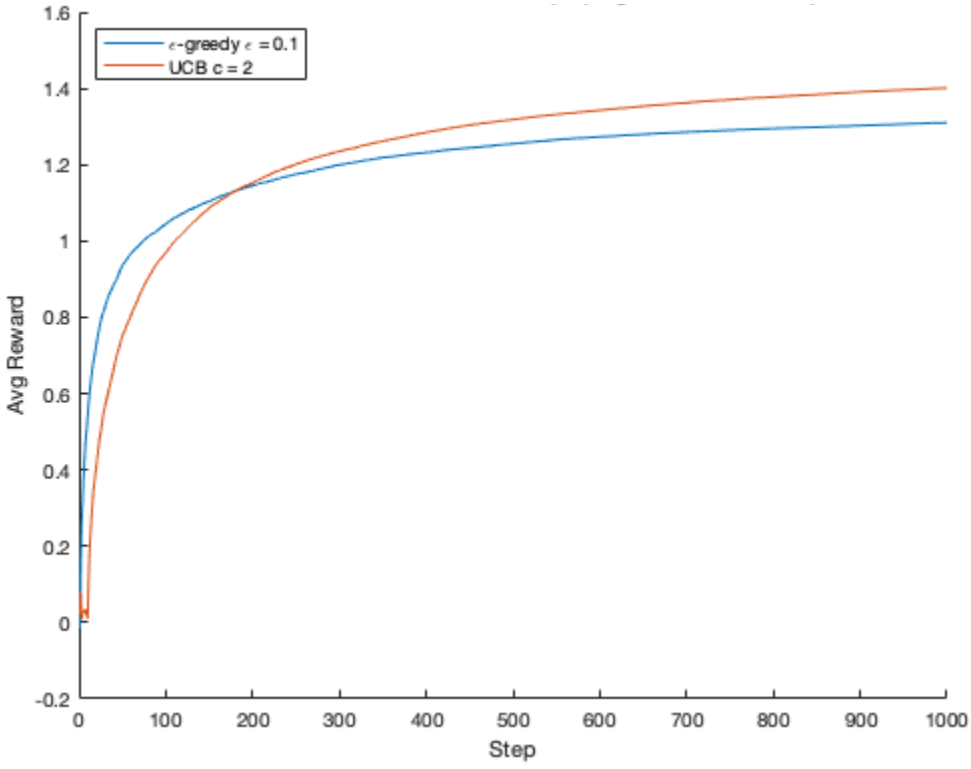

Figure 1: ε-Greedy and UCB Models on 10-Armed-Testbed

The average reward is shown on the y axis of figure 1 and the time steps are shown on the x axis. As the models train it can be seen that the e-greedy model reaches a higher average reward early on, but the ucb model achieves a higher average reward starting after around step 200. The exploration done early on by the ucb model diminishes its average reward in the earlier stages of training, but this gives it an advantage for the remainder of training since it has knowledge of all possible actions and the corresponding rewards (to some degree).

For comparison, figure 2.4 from the textbook is provided below. This figure was produced using the same test problem and the same two reinforcement learning methods. The major difference between my figure and the one from the text is in the smoothness of my response. This smoothness comes from averaging the same test problem 500 times while the text only runs the test problem a single time.

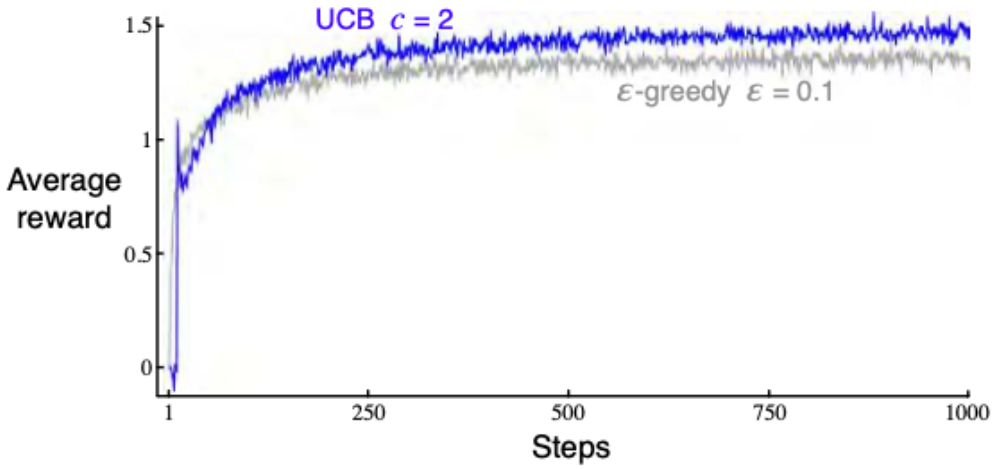

Figure 2.4: Average performance of UCB action selection on the 10-armed testbed. As shown, UCB generally performs better than ε-greedy action selection, except in the first k steps, when it selects randomly among the as-yet-untried actions.