Keaton Spiller

CS 441

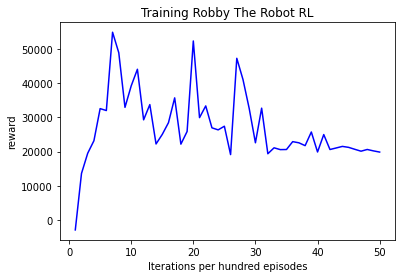
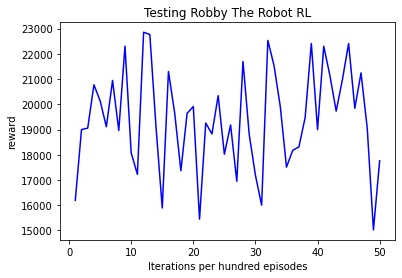
Spring 2022

Assignment # 3

Robby the Robot RL Learning Algorithm

With Q Learning and Greedy action selection

Part1: Plotting Sum of rewards of episodes per 100 episodes and the Test average and standard deviation per individual episode ( \*additionally, including the test plot for clarity)



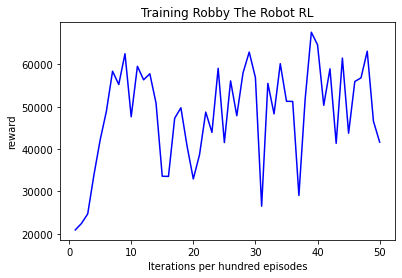
N = 5,000; M = 200; 𝜂= 0.2; 𝛾= 0.9; epsilon = 0.1

Test Average: 194.057; Test Standard Deviation: 167.26983993236797

The training initially has high rewards around 50,000, then seems to level out around 20,000 which is a good indicator that the Q-Learning matrix has learned more scenarios and can generalize better to states unseen. As seen in the test data, the average reward and standard deviation are high indicating that using the test data Robby has learned to make more of the right moves than wrong moves by a large margin. The highest reward is 10 for picking up a can and the lowest is -5 for going out of bounds. Essentially picking up 17-20 cans without a mistake.

Part 2: Experiment with Learning Rate. Choose 4 different values for the learning rate, 𝜂, approximately evenly spaced in the range [0,1], keeping the other parameters set as in Part 1. For each value, give the Training Reward plot (plotted every 100 episodes), and the Test-Average and Test-Standard-Deviation. Discuss how changing the learning rate changes these results.

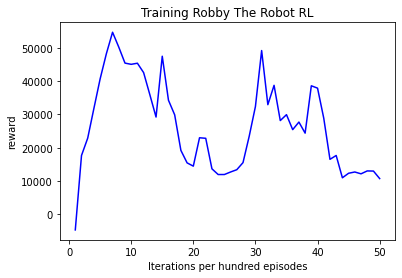
𝜂 = 0.1, 0.4, 0.7, 0.99



N = 5,000; M = 200; 𝜂= 0.1; 𝛾= 0.9; epsilon = 0.1

Test Average: 181.8044; Test Standard Deviation: 165.12499550534437

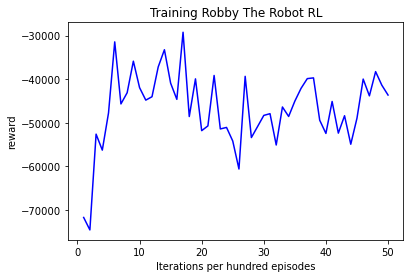
Looks like a smaller eta value has sharper changes in reward.



N = 5,000; M = 200; 𝜂= 0.4; 𝛾= 0.9; epsilon = 0.1

medium eta value smoothed out the training curve, this has a higher test average than eta=0.1 and 0.2, with relatively the same standard deviation.

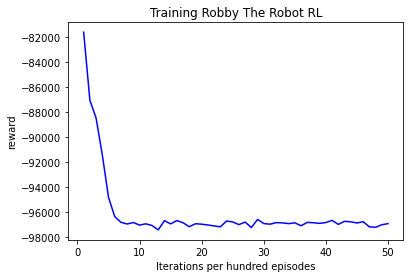
Test Average: 195.426 Test Standard Deviation: 167.74443693905323



N = 5,000; M = 200; 𝜂= 0.7; 𝛾= 0.9; epsilon = 0.1

Test Average: -419.2994 Test Standard Deviation: 526.5932036777915

A high eta value caused the reward to have massive swings with a standard deviation of 526, 5x larger than with eta 0.2. The rewards were all negative. This caused the Q-Learning algorithm to have a hard time distinguishing between good rewards from bad ones.

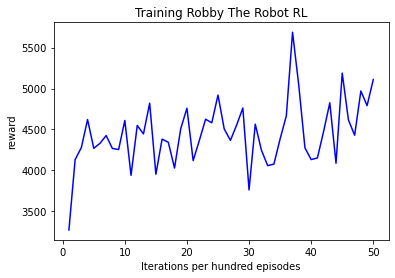


N = 5,000; M = 200; 𝜂= 0.99; 𝛾= 0.9; epsilon = 0.1

Test Average: -843.7674 Test Standard Deviation: 53.204027077280536

Too high of an eta value made it extremely difficult for the Q-Learning algorithm to learn positive rewards. However, I wonder if we wanted the negative reward if this could be a good thing. Although the test average is really low, the standard deviation is relatively small, I would think that the standard deviation would also have large swings but this large eta value stays pretty constant after -98000 reward.

Part 3: Experiment with Epsilon. Try learning with a constant epsilon (choose a value  in [0,1]). Give the Training Reward plot and Test-Average and Test-Standard-Deviation. How do your results change when using a constant value of epsilon rather than a decreasing value? Speculate on why you get these results.

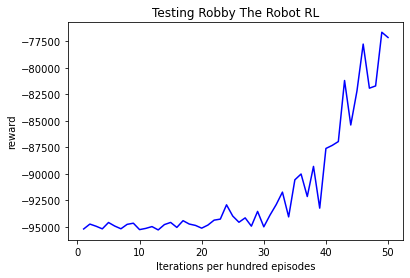


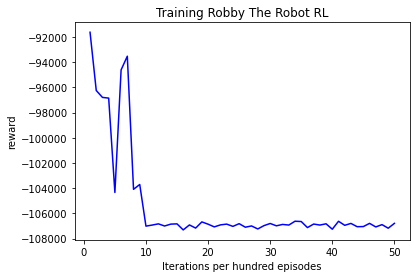
N = 5,000; M = 200; 𝜂= 0.2; 𝛾= 0.9; epsilon = 0.8

Test Average: 44.9398 Test Standard Deviation: 39.908081587067045

With A high constant epsilon in the training data we get low rewards, because the chance of choosing a random action is very high, and we never get to taper off to use more of the improved maximum’s learned from the Q-Learning matrix. This would be the same case if we used a smaller constant epsilon, but with a larger one we can see in greater depth, the severity keeping randomness can decrease the results. However, it is interesting that the reward remained positive, even when the majority of actions were random.

Part 4: Experiment with negative reward for each action. Modify your code so that a negative reward (an “action tax”) of −0.5 is given in addition to the original rewards for each action. Run learning and testing with the parameter values of Part 1, and give the Training Reward plot and the Test-Average and Test-Standard-Deviation. What differences do you see from the results in Part 1?





N = 5,000; M = 200; 𝜂= 0.2; 𝛾= 0.9; epsilon = 0.1

Test Average: -914.9034 Test Standard Deviation: 179.60039022351816

With an Action Tax, the data starts off with high negative rewards but over time it looks like the test data gradually climbs back up. I could see in running a larger experiment over time this would climb back up to positive rewards, and I assume it could do it in less actions per episode than without the tax with the right hyper parameters.

Part 5: Devise your own experiment, different from those of Parts 1−4 above. This can involve a change to a parameter value, a change in the rewards, a modification of the actions or sensors, etc. Describe your experiment and give plots or values that show the results. Are the results what you expected? Why or why not? One suggestion: use a Neural Network in place of the Q-table.