RL Coursework 3

April 17, 2018

1 Watkins Q(λ) vs Sarsa(λ)

The first step I took to approach this was by implementing the Watkins Q algorithm (found in chapter 8 of Sutton & Barto). I wanted to see the results of updating the Q values off-policy compared to Sarsa's on-policy approach to see if this benefited the learning of the agent in anyway. I used what I believed to be the best performing parameters for both algorithms and the same number of tilings. Watkins parameters were λ=0.9, α=0.01, γ=1 $\epsilon = 0.05$ and sarsa's parameters were λ=0.9, α=0.1, γ=1 $\epsilon = 0$. Figure 1 depicts the results of testing both algorithms. It shows that both algorithms eventually converge to an optimal policy, however Sarsa has the more efficient learning approach. Sarsa needs on average less than a 1000 steps towards the beginning of the episodes whereas watkins need above 2000 steps on average. Sarsa also seems to have learnt by episode 20, whereas Watkins learns at around the 40th episode.

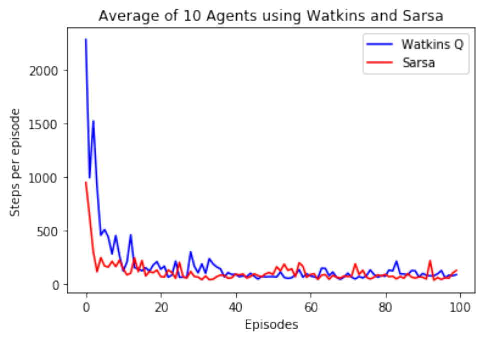
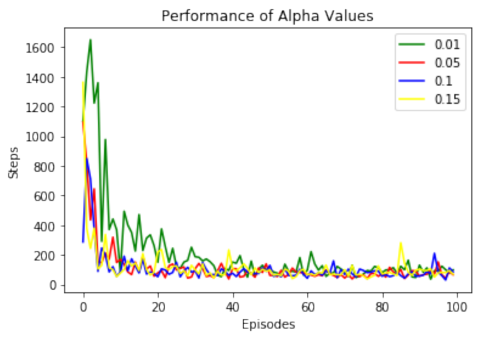
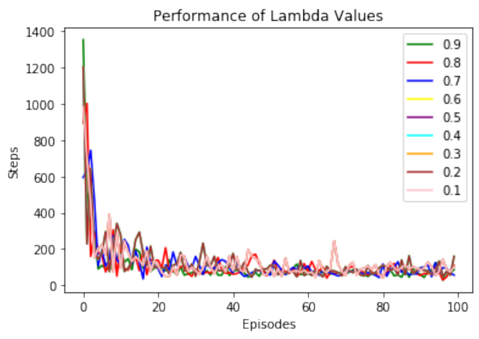


Figure 1: Average of 10 agents using Watkins and Sarsa.

2 Parameter Experiments



(a) Alpha.



(b) Lambda.

Figure 2: Parameter tests.

I performed a series of parameter changes, however found that the parameters I used in the first place seemed to give me near optimal results. I first wanted to see the effects on increasing the α value, as I found from question 1 that it was unable to learn if I made the value to high. So I wanted to see where the threshold was for increasing the α value to the maximum it could be to see if that alpha was the best learning alpha. I found that increasing the α value to more than 0.15 meant the agent was not able to anymore, so this was the threshold I was going to use for the test.

I ran some tests with different α values up to 0.15, with λ=0.9 and replacing traces. My second parameter tests was to test the values of λ, I also wanted to see if changing this would effect the learning. The results for both tests are displayed above in Figure 2.

The results shows that 0.1 was the most effective value used for α and anything above 0.1 caused the learning efficiency to deteriorate, however not by much. α=0.1 used the least steps at the beginning of the game than all other values, and this was the determinant of the optimal alpha.

Having found 0.1 was the most effective value for α I used it as the α value to test my λ changes. While all show similar patterns, changing the λ did not show too many differences. As the graph is difficult to read and the main purpose was to show that each lambda produced similar results, the λ with the best result is shown in figure 3 and was found to be 0.9. This could be because it explored the most states towards the beginning than any other lambda value. The lambda values closer to 1 showed better performance towards the last episode, so 0.9 was concluded as the best option to use.

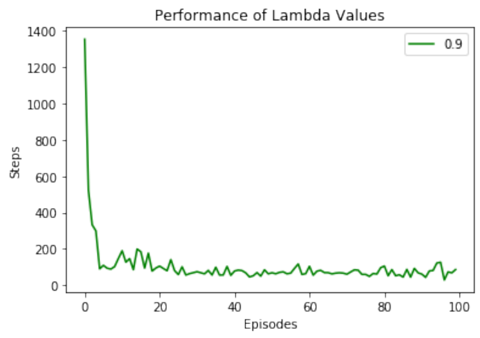


Figure 3: Optimal lambda.

3 Logspacing

Currently the tiling has equal spacing between indexes. I thought that by applying log spacing to the position axis it could improve the agents learning efficiency. I made the indexes closer together towards the goal state and the top of the other hill. I was thinking that having less states in the middle would cause the agent to move up both hills more often, thus encouraging the agent to gain inertia and reach the goal state faster. I hard coded the array, which is found below. Figure 4 shows the results of logspacing:

Log spacing array: [-1.2 -1.1 -1.0 -0.5 0. 0.3 0.55 0.6 0.65 0.67].

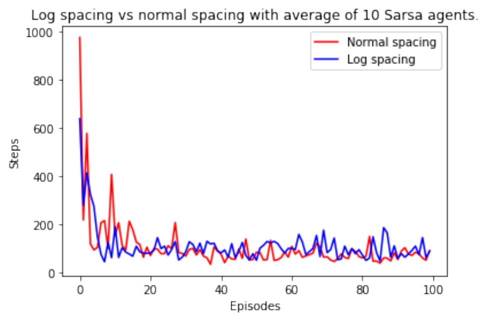


Figure 4: Logspacing vs Normal spacing.

Having run the test several times with 10 agents, consistently the log spacing caused the agent to reduce the average amount of steps taken on its very first few goes. Figure 4 shows that the log spacing agent needed on average only 600 steps to complete the first episode whereas the normal spacing was nearer 1000 steps. It seems that the agent using the original spacing benefits from the sheer amount of exploration it has on its first few goes in the long term, whereas the log spacing agent is not able to benefit as much because it has explored less. Because the log spacing agent is drawn up the mountain it reaches the goal state quicker because it has the inertia from driving down the opposing hill, it seems my prediction was correct. This benefits near the beginning, however because it has explored less during the first few episodes, it performs ever so slightly worse towards the end.