RL-3

Abhilash Jain

October 25, 2019

1 Q-learning

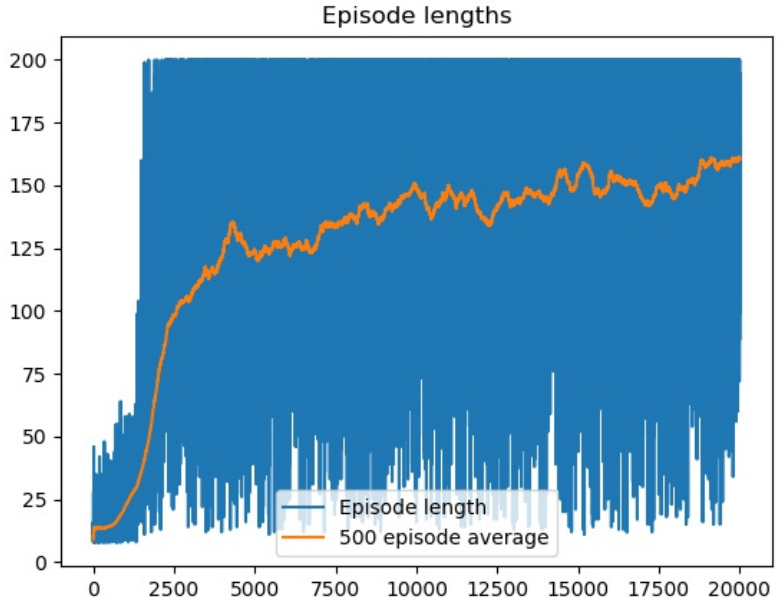
1.1 Cartpole

1.1.1 Task-1

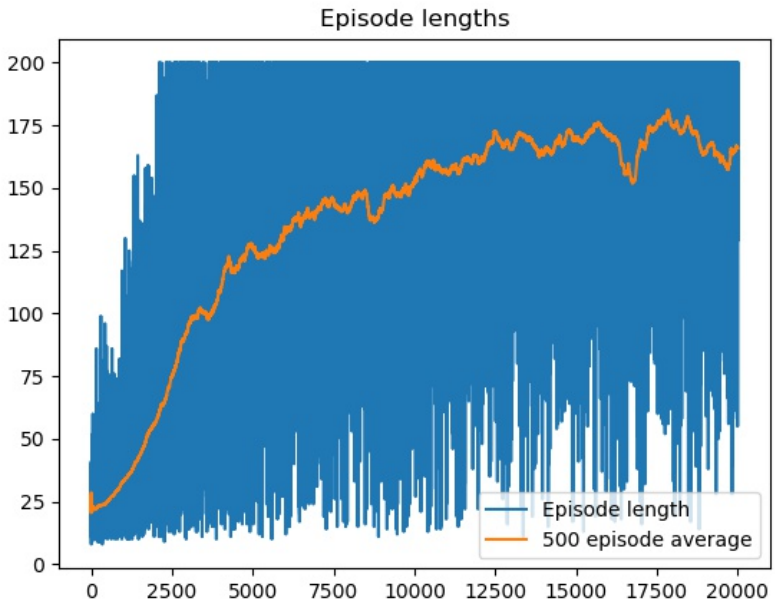
Implement Q-learning for the Cartpole environmen in the file qlearning.py. Compare using a constant value of ϵ 0.2 to reducing the value of over time (use the greedy in limit with infinite exploration formula from the lecture). Use the following hyperparameter values: α 0.1 and γ 0.98. For GLIE, aim at reaching ϵ 0.1 after 20000 episodes and round the value of a to the nearest integer.

Implementation can be found in the code file attached.

Average with constant ϵ of 0.2:



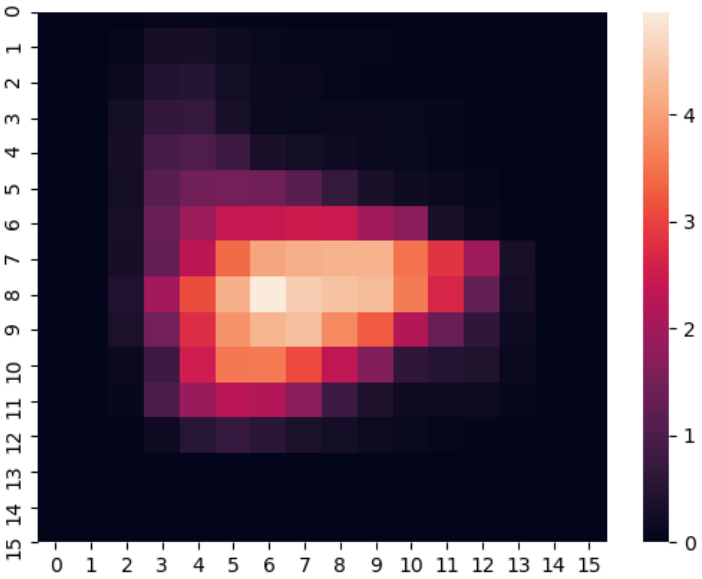
Average with GLIE:



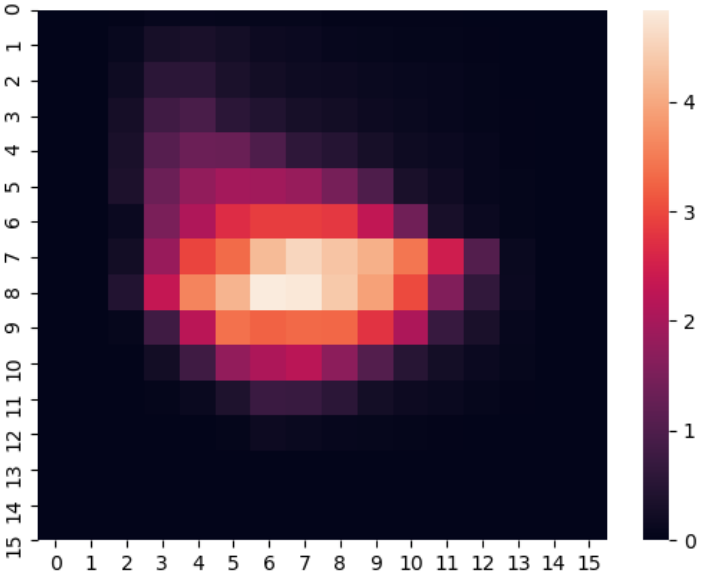
As we can clearly see that, with constant epsilon of 0.2, we do get an initial jump in the average episode length, but overtime it doesn't improve much and the average episode length is around 150. But with GLIE, even though it takes time to get good (more than the constant), but in the long run, it does perform better than the constant epsilon

1.1.2 Task 2

Heat Map with GLIE:



Heat Map with constant ϵ

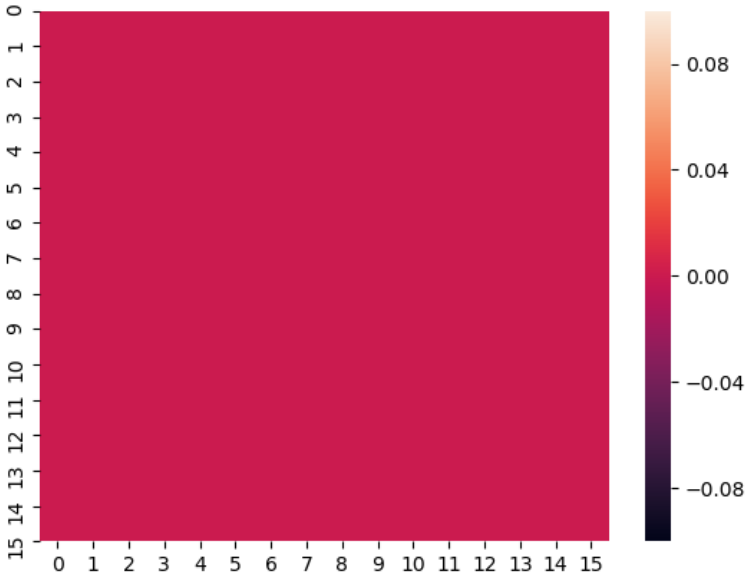


As we can clearly see, the Cartpole learns to stay at the centre as much as it can and the spread across the y-axis is because of the tilt to balance the pole as it tries to move around the centre.

1.1.3 Question 1

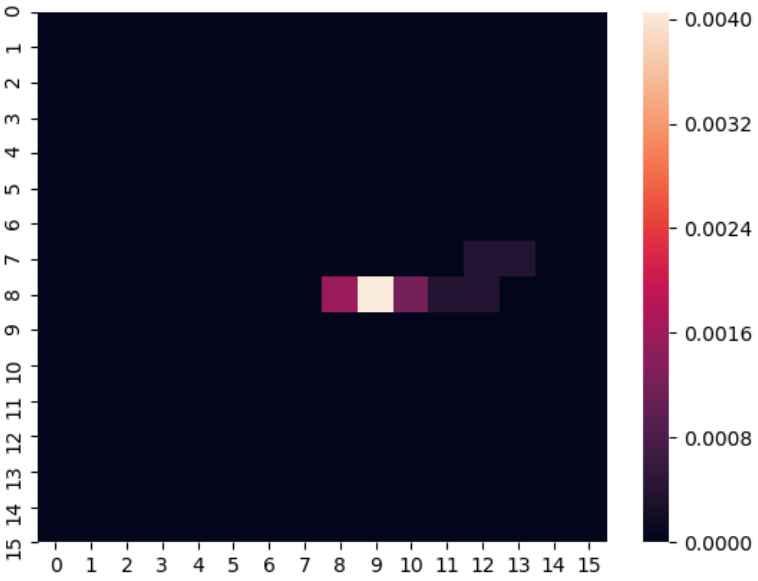
What do you think the heatmap would have looked like:

(a) before the training? The heatmap should have 0(the initial) values, as we have not started to train, the values are not updated.



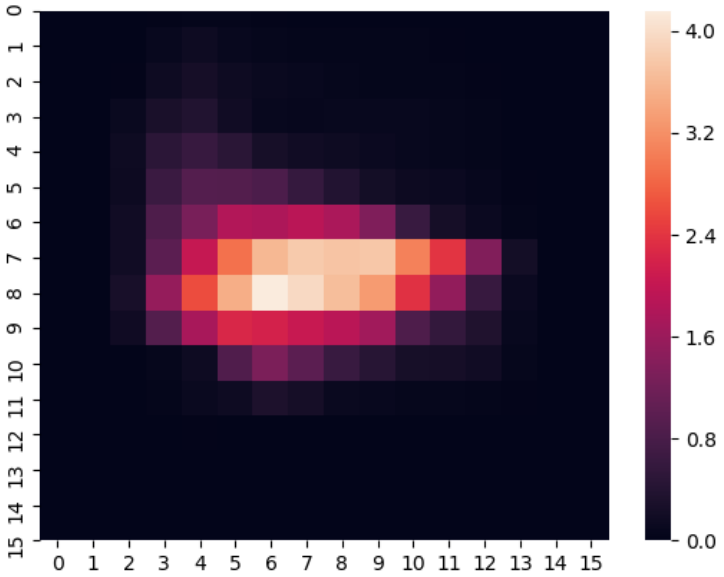
(b) after a single episode?

GLIE starts with a epsilon zero i.e. in our first episode, the starting action will be completely random, nevertheless, it stays alive for some while(30 timesteps) and tries to be at the centre that is what we can make out from the heat map. Expected behaviour as it is just the first episode.


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0.0040
0.0032
0.0024
0.0016
0.0008
0.0000

(c) halfway through the training?

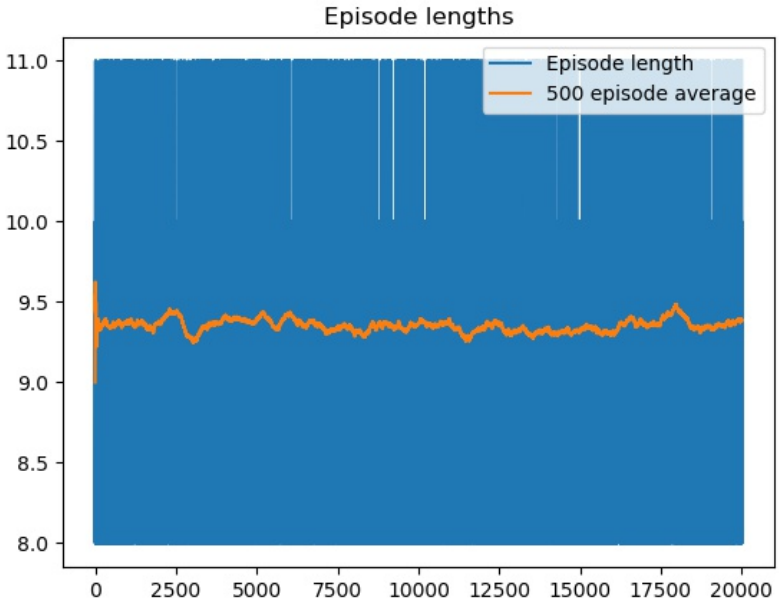
At half way, with GLIE, the model learns that the best policy is to be at the centre and it is starting to learn that. When we compare it from the full 20000 run. we see a difference that, the value at the centre is still at around 4 whereas for the 20K run the values around centre are above 4 and close to 5 which suggests that, the values would increase more at the centre. And we are slowly moving to convergence. One more thing to notice is that at 10K the epsilon is still at .18. so it is still pretty greedy, and less random than we would have thought. There-fore the heatmap makes sense at half-way training, with the given epsilon and the values.



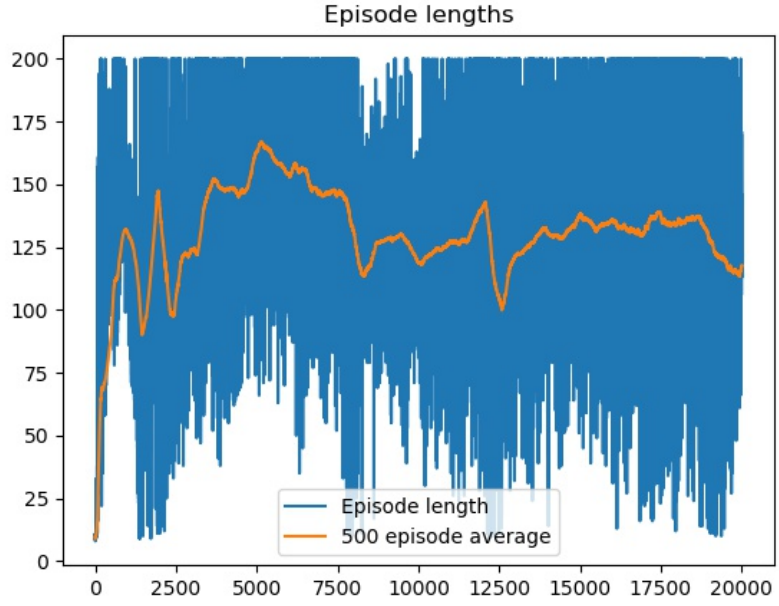
1.1.4 Task 3

Set ϵ to zero, effectively making the policy greedy w.r.t. current Q-value estimates. Run the code again,

(a)keeping the initial estimates of the Q function at 0



(b) setting the initial estimates of the Q function to 50 for all states and actions.



1.1.5 Question 2.1

In which case does the model perform better?

We can clearly observe that, that setting 50 as initial values is way better than the initial values as 0 indicating that, for this problem at-least with initial values of 50 it tries to learn, even if it was not as good as GLIE.

1.1.6 Question 2.2

Why is this the case?

In this problem, where we have set the epsilon to be zero, indicating that we need to take the most greedy action, from the get go. When the initial values is zero,it would always select the action which is the most greedy as it would be better than the value 0,it is not able to select actions which would payoff in the long run, instead the model chooses the most greedy action with no hint of exploration every episode. That is why it barely averages 10 timesteps per episode constantly, It always selects the same type of actions, hoping to improve but doesn't as the initial estimate was 0.

When we set the initial values to 50, we are encouraging in the sense 'exploration', and even though the first few episodes it doesn't perform well, it quickly improves when compared to Initial 0. What we mean by this here is, an initial estimate of 50 for this problem is highly optimistic, and when the greedy algorithm,selects the action which doesn't pay off as high as the initial estimate, it tries other actions and when those start paying off, it learns to do it. We clearly see a jump in the average immediately, indicating that the model learned to take some other actions which is quite different from the previous model, telling us that high optimistic values such as 50 are very good in this case. To be also noted is that the model didn't perform good when the initial value was set to 100, indicating this is a parameter which can be optimized

In summary, when we are selecting the most greedy action(ϵ is 0), and initial estimate of 0, doesn't help the model at all, as there is no 'benchmark' it tries the same pattern over and over again indicative of the straight line around 9.5, but with the initial estimate of 50 the model has this high benchmark which it quickly learns from trying different actions when compared to the previous model.

1.1.7 Question 2.3

How does the initialization of Q values affect exploration?

Initial values can be used as a simple way to encourage exploration. "We have discussed before of how inherently the initialising of values encourages exploration".When we initialize a optimistic initial value, we set the model a 'benchmark' which it needs to learn to perform better from,after some unsuccessful(disappointed) attempts, the model will learn to take a variety of actions(inherent exploration even though it is temporary). There-fore, when we set the initial values as zero, we basically say any attempt having a positive outcome is good and the model is satisfied(no exploration.) And as we increase

the values we increase the exploration initially. We need to be careful of the fact if we set the initial value to be too high, our model may never learn any important behaviour/action. Q values can be thought as a parameter we can optimize on. And also not always setting the Q values is helpful. It is not well suited to non-stationary problems because its drive for exploration is inherently temporary. If the task changes, creating a renewed need for exploration, this method cannot help.

1.1.8 Task 4

Modify your code for Task 1 and try to apply it in the Lunar lander environment. Run it for 20000 episodes (which was enough for the Cartpole to learn).

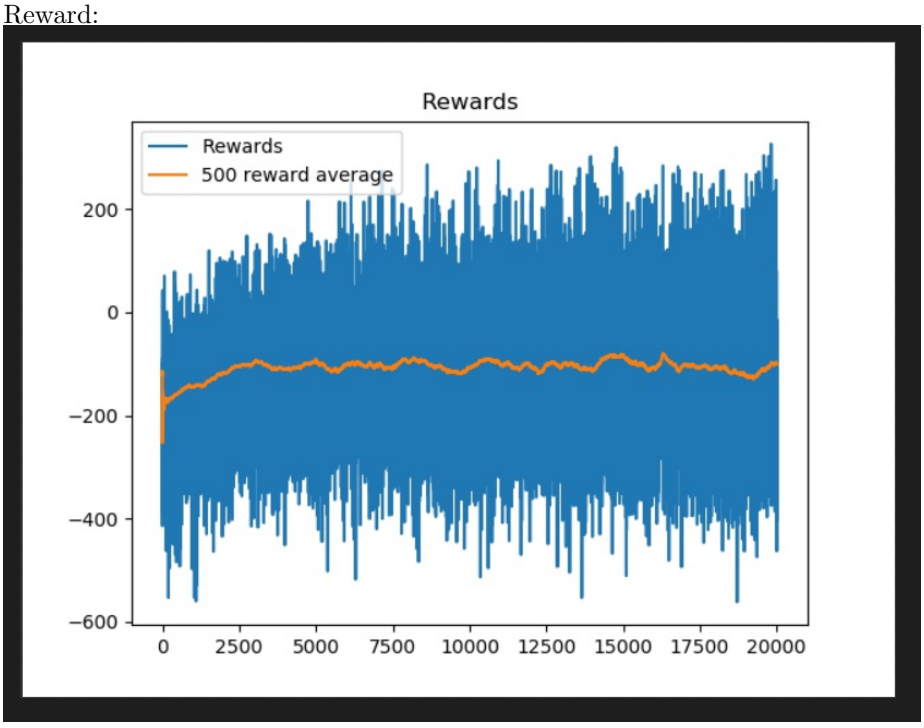
Code is implemented, and attached.

1.1.9 Question 3.1

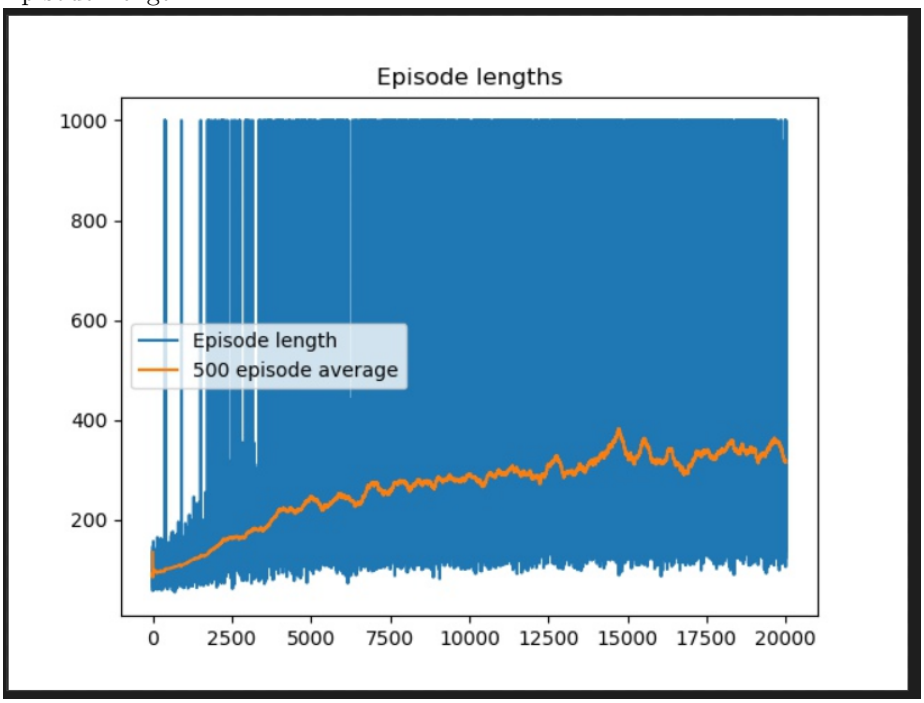
Is the lander able to learn any useful behaviour?

While rendering the environment it was noticed that the lander seldom reached its goal and didn't learn any useful behaviour, it reached its goal, once out of 10 tries. We can also see from the cumulative reward graph, that useful behaviour was not learned.

Reward:



Episode Length:



1.1.10 Question 3.2

Why/why not?

We use the cumulative rewards as a better means to judge the performance, as the average timesteps, didn't really make sense to judge the performance. From the graph it is evident that the increase per episode is very minimal/non-existent. One of the reasons for that could be that the state space is very complex with 8 value observation vector and 4 dimensional action space. hence to run q-learning on it is not feasible. We also have a more complex goal compared to the cartpole, which the q-learning algorithm is not able to learn. Q-learning at its simplest stores data in tables. This approach falters with increasing numbers of states/actions. which for the Lunar Lander is the problem. For this problem combining Q-learning with function approximation will help it tackle large problems like the Lunar Lander.