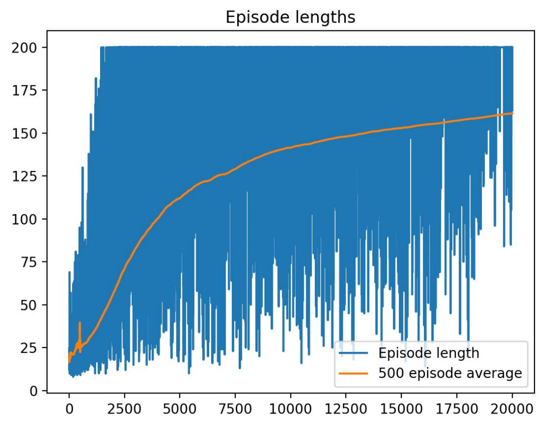
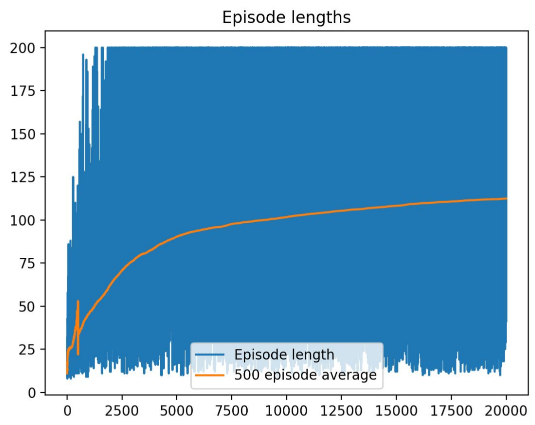
Reinforcement Learning - Exercise 3

Muhammad Kamal Memon
600442

Task 1

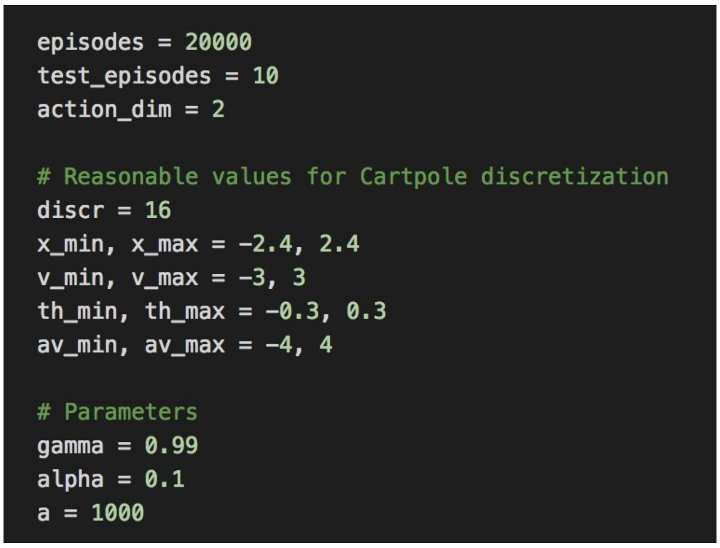


Decreasing ε over time



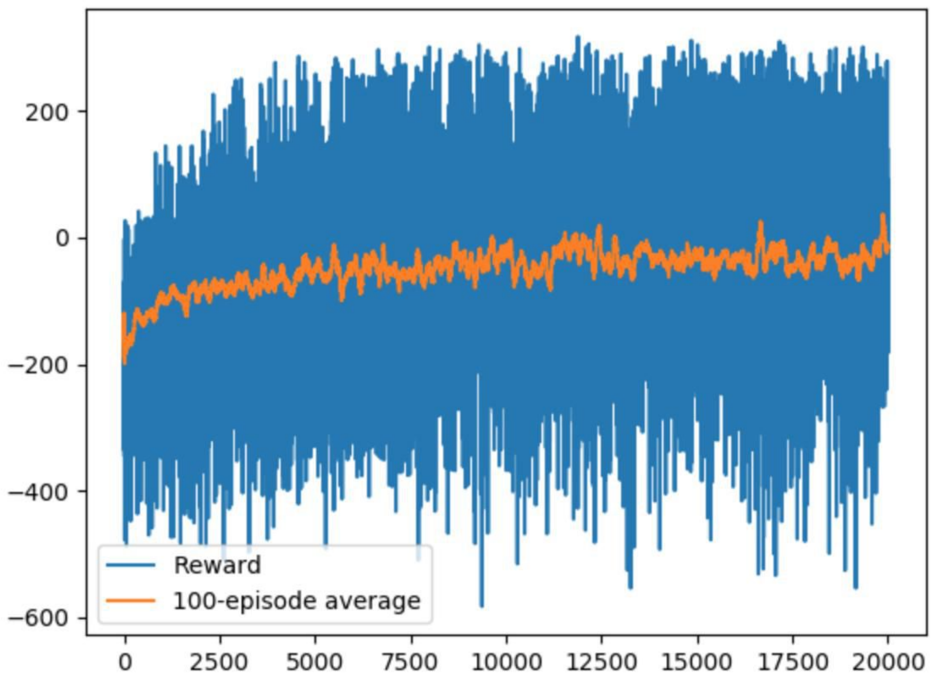
Constant ε = 0.4

As can be seen in the graphs, decreasing epsilon (ε) over time makes the policy more greedy, that is the probability to select the best action increases and hence the episodes length increase over time. Whereas with a feasible constant epsilon the episodes length still increase but its not consistent.



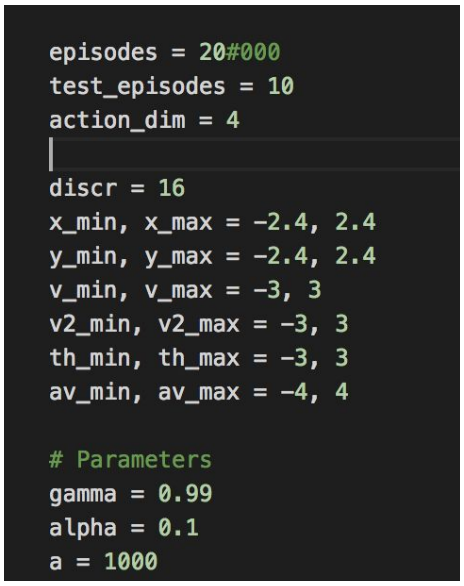
```
episodes = 20000
test_episodes = 10
action_dim = 2

# Reasonable values for Cartpole discretization
discr = 16
x_min, x_max = -2.4, 2.4
v_min, v_max = -3, 3
th_min, th_max = -0.3, 0.3
av_min, av_max = -4, 4

# Parameters
gamma = 0.99
alpha = 0.1
a = 1000
```

Task 2



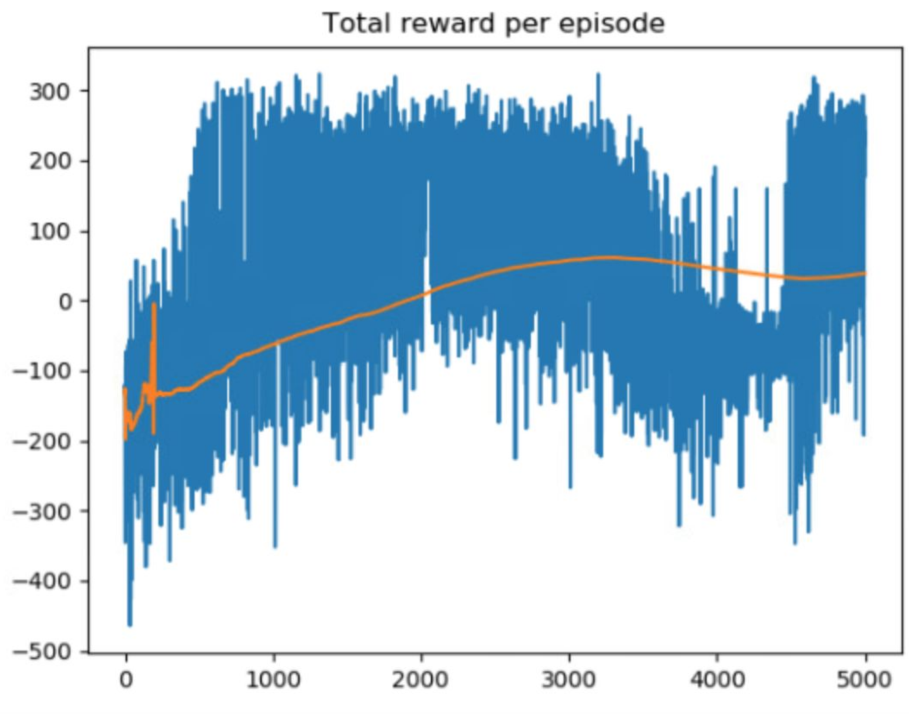
Rewards and average reward history of q-learning for Lunar Lander

As evident from the plot above, the training fail to converge satisfactorily and the average increase in rewards over episodes is very minimal. While rendering the environment we observed that the lander is not able to learn any useful behaviour and seldom reaches the goal properly. The reason being that the state space is too complex with 8 value observation vector and 4 dimensional actions space, hence to run the exact q-learning over it is not very feasible.



```
episodes = 20#000
test_episodes = 10
action_dim = 4

discr = 16
x_min, x_max = -2.4, 2.4
y_min, y_max = -2.4, 2.4
v_min, v_max = -3, 3
v2_min, v2_max = -3, 3
th_min, th_max = -3, 3
av_min, av_max = -4, 4

# Parameters
gamma = 0.99
alpha = 0.1
a = 1000
```

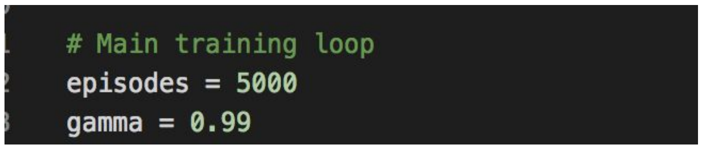
Task 3



Rewards and average reward history of function approximation for Lunar Lander

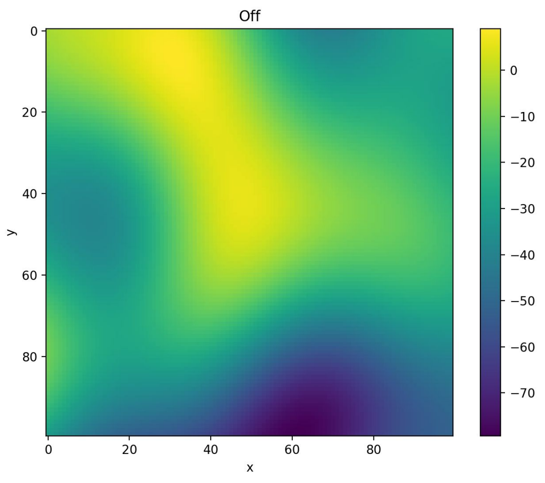
The rewards do better with rbf approximation as compare to q-learning. The average rewards increase relatively over time and near the end it seems to stuck in a local minima but gets out of with some consistent positive rewards in the end.

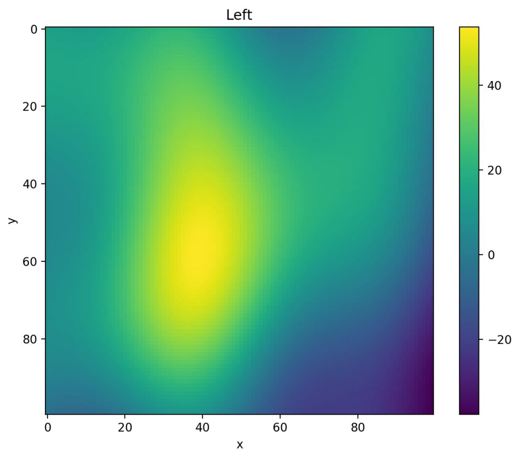
While testing the model and rendering the environment, the Lander does way better than q-learning one. It lands well most often than not and usually inside the flags. However, it seems that reward function isn't very intuitive as it rewards higher if the lander is fast and even give high rewards when landing way far from the flags. Furthermore there still seems to be a lot of room for improvement in Lander's behaviour hence we deduce that function approximation isn't working perfectly here.

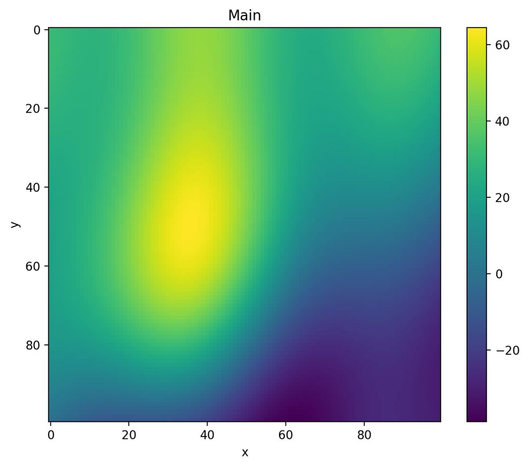


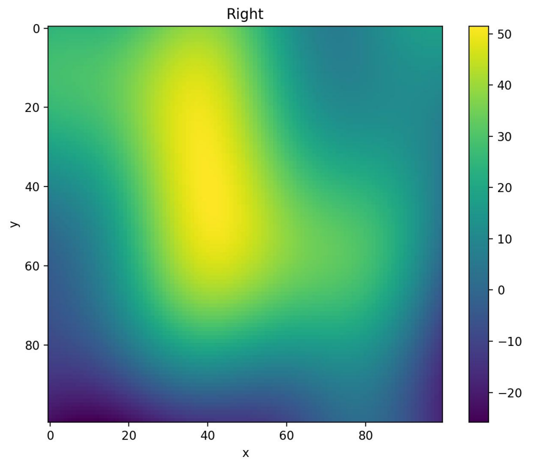
```
# Main training loop
episodes = 5000
gamma = 0.99
```

Task 4

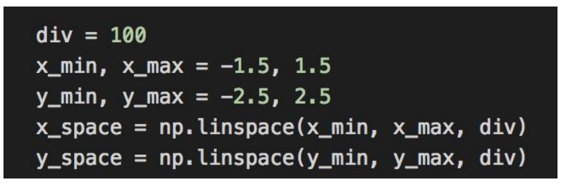








The "Off" action heatmap suggest that it gets most reward when in the top and center areas, that is, it should be landing directly beneath and turn the engine off. The "Left" heatmap is more bright on the central left and left side of the map as to suggest higher rewards are towards left. Similarly the "Right" action heatmap has higher rewards towards central-right and right sides of the map. The "Main" is mostly higher at the left and center as it complement left and right actions so that it can land down.



```
div = 100
x_min, x_max = -1.5, 1.5
y_min, y_max = -2.5, 2.5
x_space = np.linspace(x_min, x_max, div)
y_space = np.linspace(y_min, y_max, div)
```

Question 1:

Components:
Increasing this parameter results in longer computation and it means to have more samples of features.
Decreasing this reduces the time for training but the model have less samples to fit on.

Gamma values:
The values are the variance in the Gaussians in RBF Sampler. Increasing it will give more weightage to samples and won't take other samples into account and make the model overfit on the data. Decreasing it result in low variance and don't represent the data fully.

Question 2:

It is possible I think but is not suitable in this environment mainly because the observation vector of the state have non-linear values. It would be a bad approach to approximate these high dimensional values via linear approximation function.

Discussed with Gadidjah Ogmundsdottir and Fabio Collea