Problem 1

Design Choice

I made the choice for NN over GP as I had never worked with NNs before. I started with trying a simple single layer perceptron (size = 64) with no activation functions. I tried different variations for the design and made the following observations :

1. I couldn't get any architecture to converge quick enough so updated the total number of episodes for training to 5000 episodes.
2. Due to the nature of the problem, simple fully connected layers were chosen.
3. A simple mean squared loss was chosen as the problem was similar to a regression function (This choice also worked best amongst other loss functions that were tested).
4. A single layer perceptron (size) with no activations function wasn't able converge and had total number of successful attempts ¡ 100. I assumed this is because there may not be enough neurons to approximate the Q function. Increasing the number of layers more than 3 made the loss drop down to 0 very quickly without converging to a solution. This may have been because of overfitting because of having too many neurons. Ended up using 2 layers.
5. Using the RELU function gave zero Q values or made the values converge to constant Q values for all actions. Given that the initial position state of the problem is mostly negative, most of the neurons proved useless and hene a proper solution could not be achieved (using this reasoning to explain weird zeroing of Q values I observed. Could also have been a tensorflow error. I couldn't completely figure this out). Hence used leaky RELU, so initial negative inputs to layer are not ignored and help weight update to correct values. Other activation functions didn't give more promising results.

Exploration - Exploitation Strategy

It was observed that the network diverged with high values of epsilon, however, a few successful episodes were needed at the start to make the network learn. Hence high epsilon was initially chosen(0.4), favoring exploration, and the value was decreased with each successful iteration, as the the network learned the Q function, making sure eventually exploitation is preferred and the network doesn't diverge.

Hyperparams

Not much affect was observed with changing discount factor. A high learning rate was initially chosen(0.5) to facilitate learning new observations and decayed with each successful episode, making sure learning is biased towards successful episodes. Choice of loss function explained above. Termination condition is a fixed number of episodes, large enough that enough successful episodes are observed by agent during exploration (around 2000). Once enough such episodes are observed, convergence usually occurs quickly. Hence 5000 episodes are chosen.

Q function

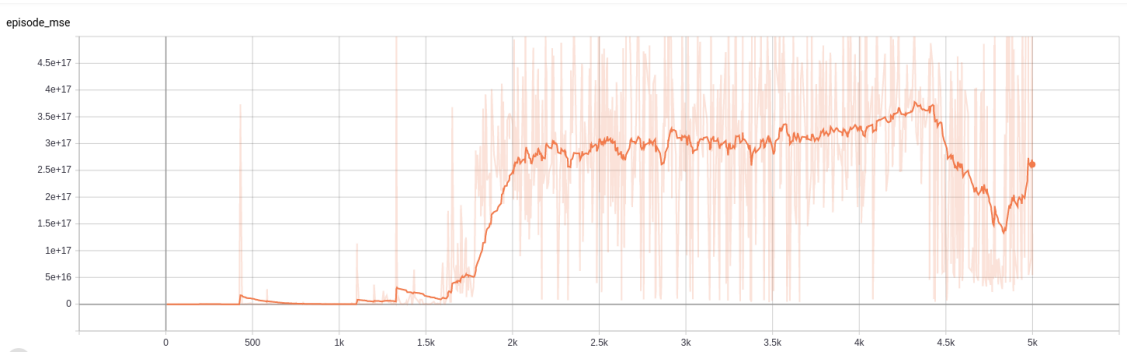

Figure 1: Total Mean Square Error Loss per Episode

Videos

All the videos may be found in the following Google Drive Link.

Performance

The trained model was tested for 1000 iterations with 957 successful iterations (95% accuracy). Out of these,

- Iterations with reward greater than -100 : 346
- Iterations with reward greater than -150 : 424
- Iterations with reward greater than -175 : 913

The reward per episode during training is given by figure 2, while the running average of reward per 100 episodes is given by figure 3. Figures 4 and 5 give the count of the number of times the reward was above -150 and -175 during training.

Another observation on the performance of the algorithm is it's dependence on finding successful episode during exploration. If the number of training episodes is fixed, the earlier the successful episodes are found, the better is the model accuracy. It was observed that this can happen as early as first 100-200 iterations to 1000-2000 iterations. Though the convergence pattern remained same once enough successful iterations were observed by the agent. The accuracy may vary from 75-95%, however if the training is performed long enough the accuracy becomes more or less constant.

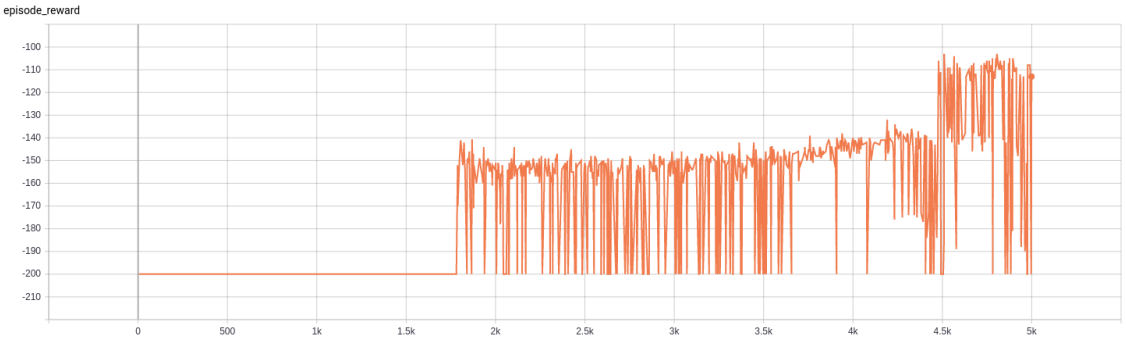

Figure 2: Reward per Episode

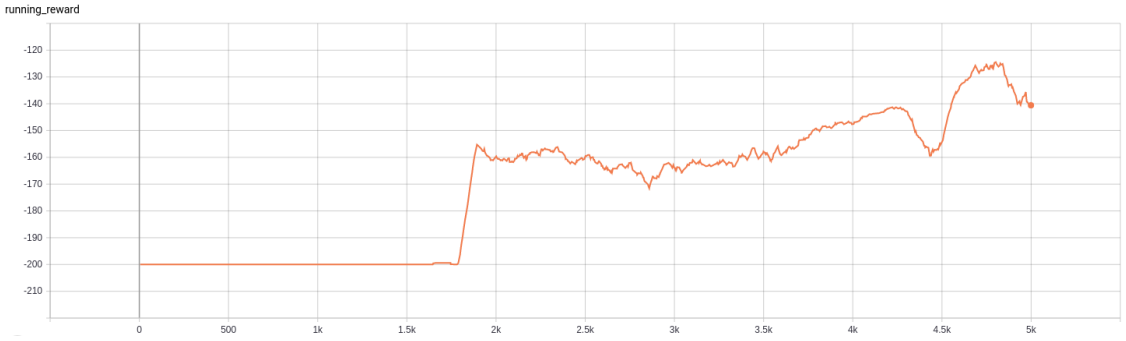

Figure 3: Running average of rewards for 100 episodes

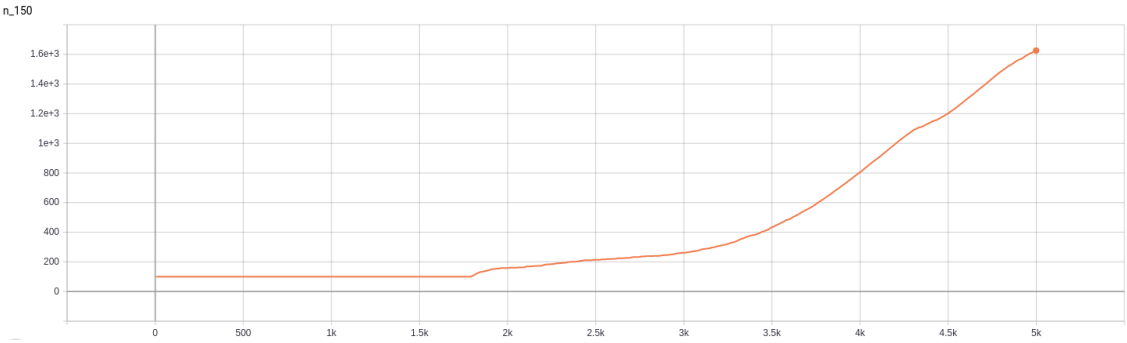

Figure 4: Number of times reward was above -150

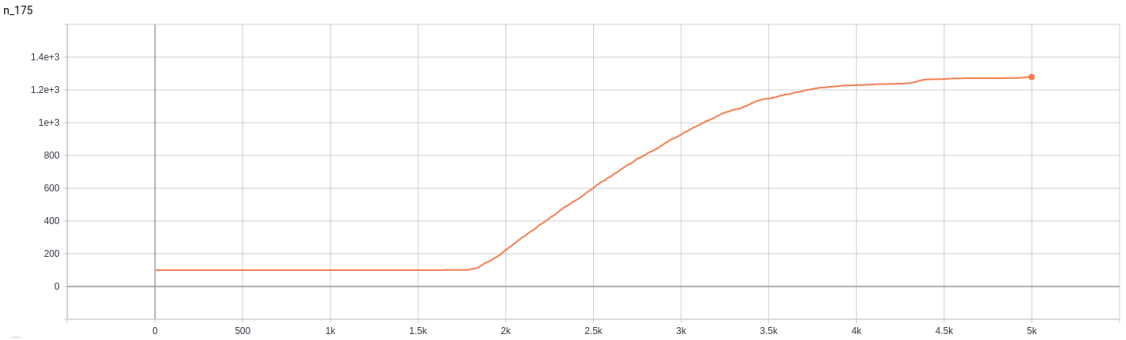

Figure 5: Number of times reward was above -175