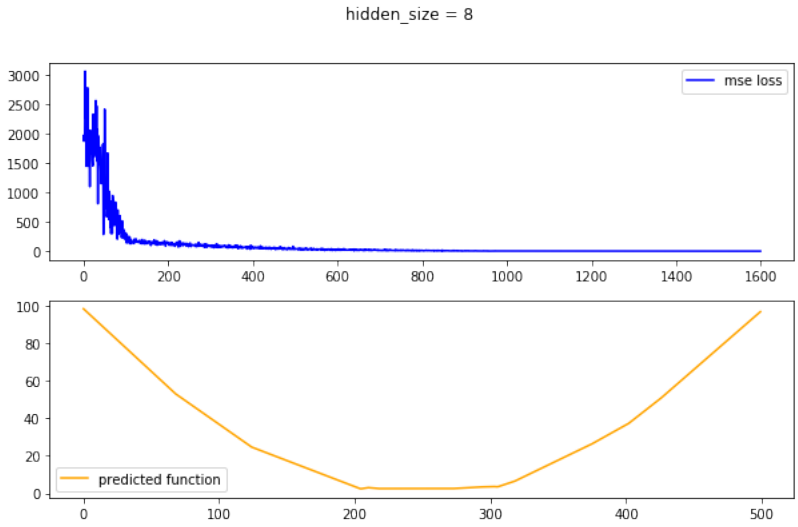
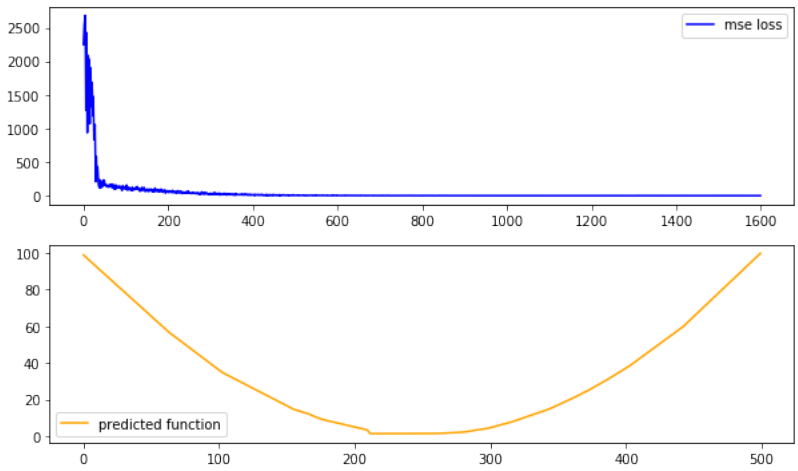
Garrett Partenza
Advanced Reinforcement Learning
April 5, 2022

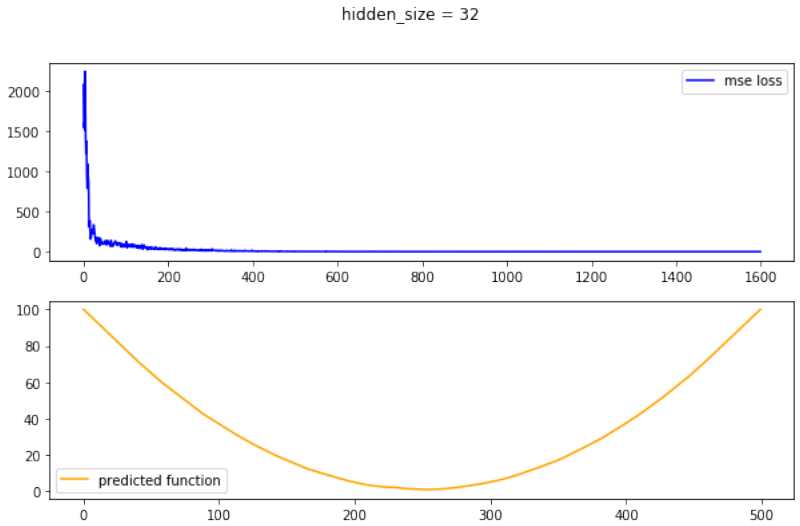
Exercise 8

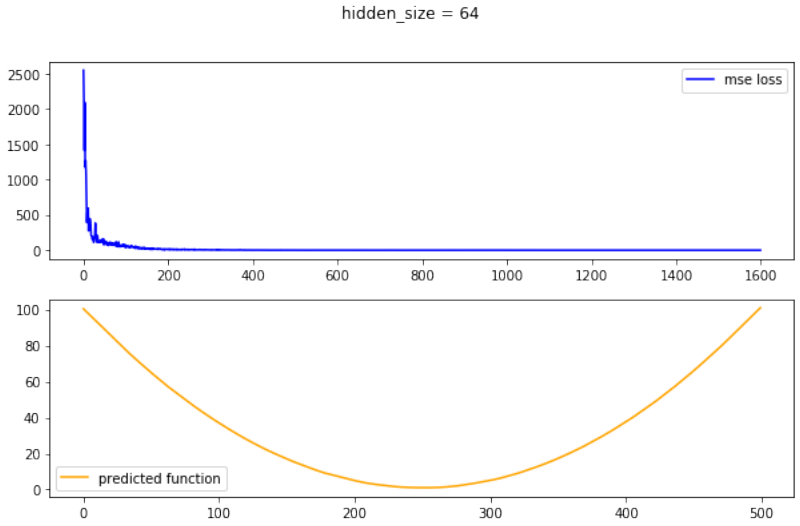
Question 1. How accurate is your learned model? How accurate it is within the range of [10,10]? How about outside of the range? Can you notice any difference between models having different layer widths? What do you think that may have caused the difference?

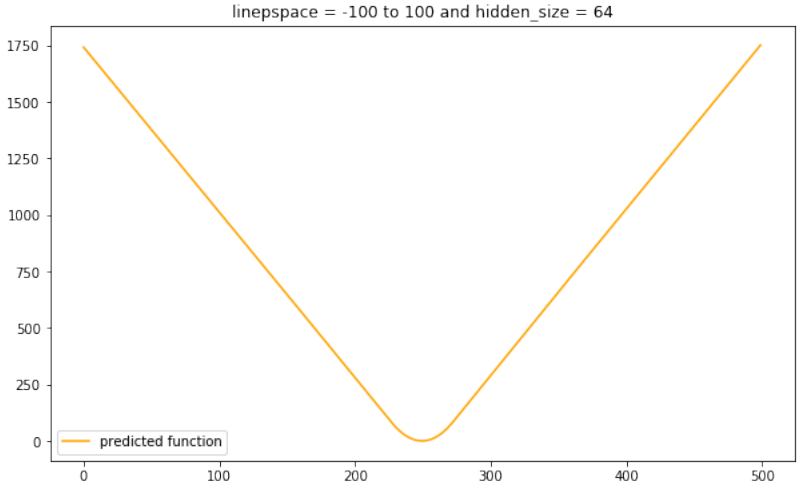
All code concerning this answer can be found in question1.ipynb. My learned model is very accurate, achieving a minimum MSE loss of 0.0020633218. It attains this level of accuracy both inside and outside of its trained range of (-10, 10). I noticed that as the number of hidden nodes increased, the prediction line better fit the line of the actual function. I assume this is because increasing the number of hidden nodes increases the expressive capability of the model to better approximate the function. I would assume this also results in the model having a higher variance.







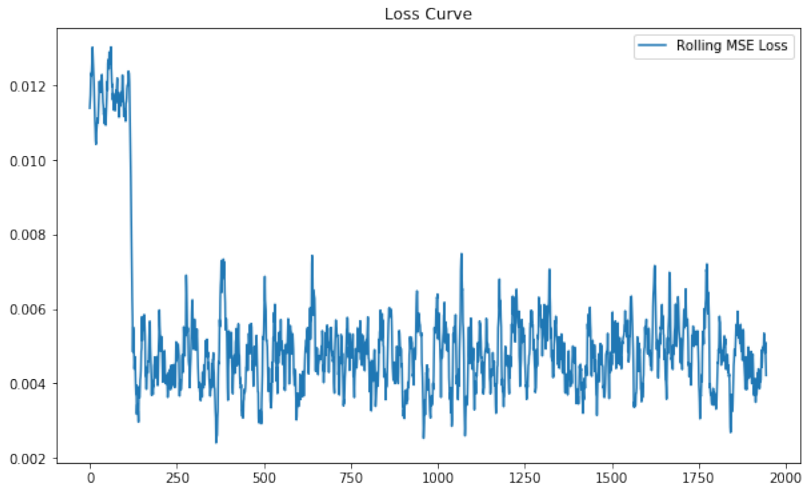




Question 2. Plot your learning curve (averaged over 10 trails) with confidence bands. How does it compare to your tabular methods from previous exercises?

Please excuse that I could not run multiple trials (the plot below only includes one trial) as my machine could not train in a viable amount of time given my network architecture. All relevant code for this question can be found in question2.ipynb. The below plot shows the HuberLoss for my model over 2000 episodes. I used a learned feature method to represent the state to the DQN network, whereby I passed the board state (a 2D-matrix where walls are 2, current location is 1, and empty positions are 0) into two sequential

convectional layers. The performance of my DQN model was poor and performed worse than the tabular methods trialed in previous assignments. I have not yet pinpointed what is causing the model to not learn, but I assume more time altering my hyperparamters could possibly solve the issue.



Question 3. Evaluate and tune your DQN on more environments

All relevant code for this question can be found in question3a.py. Helper functions and classes can be found in cartpoledqn.py, replaymemory.py and cartpole-helpers.py. Please activate the virtual environment requirements stored in requirements.txt. Below shows the live loss during training as demonstrated in the pytorch documentation here. The network parameters where three 2D convultional layers followed by a single linear layer while the hyperparameters I use were batchsize = 128, gamma = 0.999, target-update = 10, and episodes = 500. I also gradually decayed epsilon to 0.05.