Problem 1

The given Mountain Car problem is a standard problem in the OpenAI gym. The observation states is as follows

State	Min	Max
Pos	-1.2	0.6
Vel	-0.07	0.07

and the action space is a set of 3 discrete numbers: For learning the Q-function I have used a Multi-Layer

Num	Action
0	push left
1	no push
2	push right

Perceptron neural network with the following design parameters:

1. **Number of layers and number of neurons in each layer** Given the state as input and Q-values corresponding to each action as the output of the neural network, the model has 2 inputs and 3 outputs. The network also has a single hidden layer with 64 parameters. This choice was made pertaining to the face that that the function to be approximated is fairly simple and could be learned with fewer number of parameters. Also this helps keep the training time small as there are less number of compositions. The sequence of the layers is - input layer(2) fully connected to the hidden layer(64) which is then fully connected to the output layer(3). This choice was also inspired by many blogs and implementations for the same problem available online[1],[2].

2. **Activation Function** The input and output layers do not have any activation function whereas the hidden layer contains a ReLU(Rectified Linear Unit). The major benefits of using ReLu are sparsity and reduced likelihood of a vanishing gradient. The constant gradient of ReLU unlike sigmoid and tanh leads to faster learning. Also ReLU is more computationally efficient as it just needs to pick max(0,x) as compared to expensive exponential operations in sigmoid.

3. **Loss Function** Two loss functions were tried - Huber's Loss

$$L(y, f(x)) = \begin{cases} \frac{1}{2}\left[y - f(x)\right]^2 & \text{for } |y - f(x)| \le \delta, \\ \delta\left(|y - f(x)| - \delta/2\right) & \text{otherwise.} \end{cases} \tag{1}$$

   and Mean Squared Error loss

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{2}$$

   . Of these Mean squared error loss was chosen as it helped in faster convergence of the model.

4. **Discount Factor** Since the termination condition was very strict, with the defined goal state and a very small number of steps in which goal is to be reached, high reward was being given to future actions. The chosen γ was 0.99.

5. **Learning rate** This factor required a lot of tuning. At first, the learning rate was chosen to be 0.0001, which was very small for the model to learn in under 15K epochs(number of episodes). It was increased by a factor of 10 but lead to no convergence. Finally a very high *learning rate* of 0.5 was chosen by looking at the Q and target_Q values being predicted by the network which were of the order of 10^8.

Problem 1 continued on next page...

This could have been due to seeding for weight initialization and high magnitude of reward values. This learning rate forced the network to oscillate on a very large scale which resulted in the car climbing up the hill in multiple cases.
Since the learning rate was too high the model also oscillate in the other direction. To counter this I applied a rate scheduler to decrease the rate by a factor of 10 whenever an episode was completed successfully i.e., the position of the car > 0.5m. Now whenever the episode completed successfully the learning rate would decrease resulting in smaller consequent oscillations and finally convergence.

6. **exploration-exploitation** The exploration-exploitation strategy was fairly simple. The initial ϵ was chosen to be 0.3. A random number(0-1) was then generated. If random_number$< \epsilon$ then a random action was executed - exploration, otherwise the most optimal action available was chosen. This was also updated after every successful episode i.e., the position of the car > 0.5m. The chosen criterion was:
if ((n_iters $<$ 200) or (pos $>$ 0.5)):
$\epsilon = \epsilon * 0.99$.

   This helped in an exponential increase in model convergence as the car starts to learn to climb the mountain.

Results and Discussion

The model was trained for around 15K iterations with above said parameters. The MSE loss plot for the training is shown below: Once the training was complete the model was tested for 1000 iterations by setting

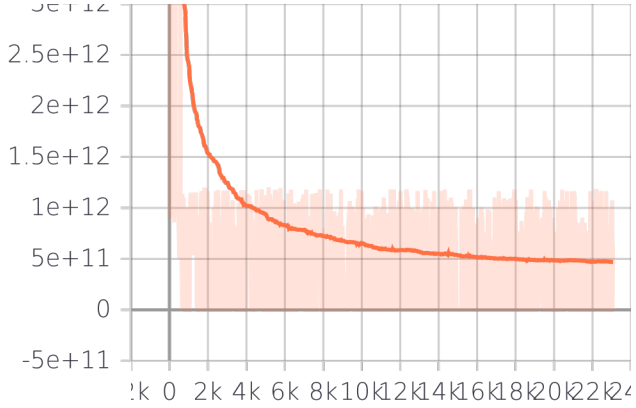
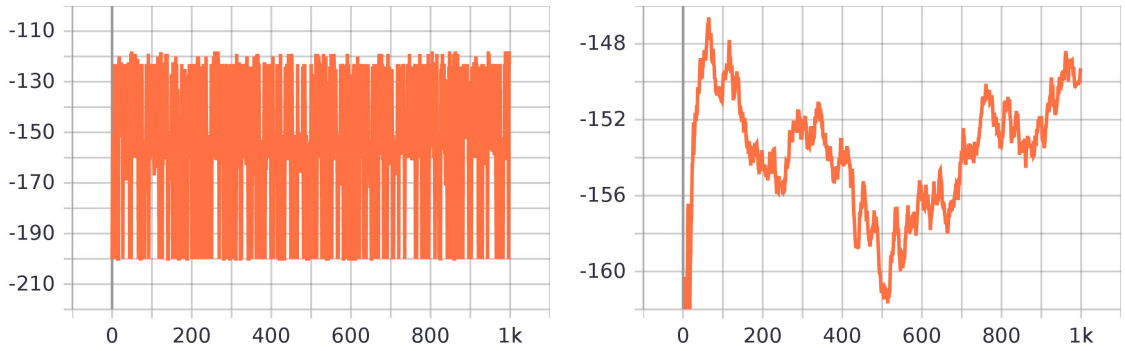


Figure 1: MSE loss per episode

epsilon = 0(only optimal actions executed as learned by the model). The model reports an accuracy of 83%, exactly climbing the mountain 836/1000 times. The running average of the rewards collected in the last 100 episodes was also observed:

- Number of episodes where the running average of the rewards drops below -175 is 998/1000 - 99.8%
- Number of episodes where the running average of the rewards drops below -150 is 113/100 - 11.3%
- Number of episodes where the running average of the rewards drops below -100 is 0/1000 - 0%

This shows that the car is able to climb up the mountain in most of the cases but takes some time to reach there. The plot of reward collected per episode and running average of the collected rewards is shown below:



(a) Reward collected per episode

(b) Running average of reward collected

A video of the rendering of the environment is provided in the submission folder. It shows the behaviour of the model from start of the training to the end of the training showing improvement in the model's ability to climb up the hill.