# RL Homework 4

Rishabh Verma

N13315483

## 1 Q-Learning with Neural Networks

### 1.1 Problem Statement

Implement the Q-Learning with a neural network for the Q-function to solve the inverted pendulum problem as shown in Fig 1.

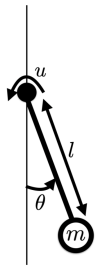



Figure 1: Inverted Pendulum

### 1.2 RL Model

The goal is to minimize the cost function

$$\min \sum_{n=0}^{N} \alpha^n g(x, v, u)$$

where

$$g(x, v, u) = 0.01 * (1 - \cos(x - \pi))^2 + 0.001 * v^2 + 0.00001 * u^2$$

The allowed control inputs are [-5, 0, 5].

### 1.3 Neural Network

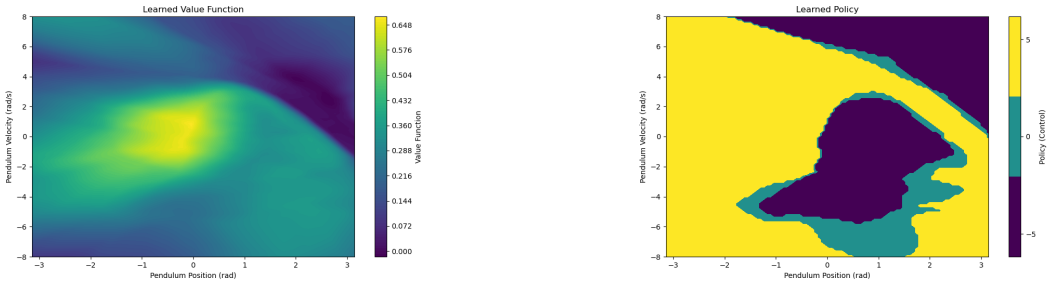
The Neural Network to be used by the Q-Learning is modeled as 2 hidden layers with 64 nodes with 2 inputs and 3 outputs for the 3 control inputs. We are using SGD optimizer and MSE loss.

### 1.4 Q-Learning

We begin by initializing each episode by the state $x_0 = [0, 0]$. We run each episode for 100 steps and in each step we select a control u according to the $\epsilon$-greedy policy which explores 10% of the time by a random control action and goes for the control action that optimizes the Q-Value rest of the time. Then the next state is computed which gives us the target value $y_n$ which has both the current reward and the estimated future rewards. Then the SGD step is performed to adjust the weights of the neural network. This optimization step then minimizes the MSE loss of the predicted Q-value and target $y_n$. Thus at each iteration the network slowly moves to optimizing the Q-value.

### 1.5 Results and Observations

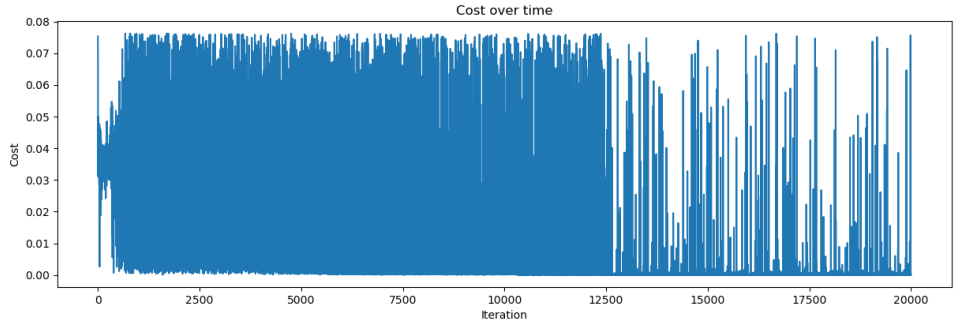
From the value function map in Fig 2a we can see that the model tries to prevent it from staying at the zero positon as shown in the yellow area and tries to move to $\pi$ position as show with the darker shades. The video for push and without push is attached with the zip file.



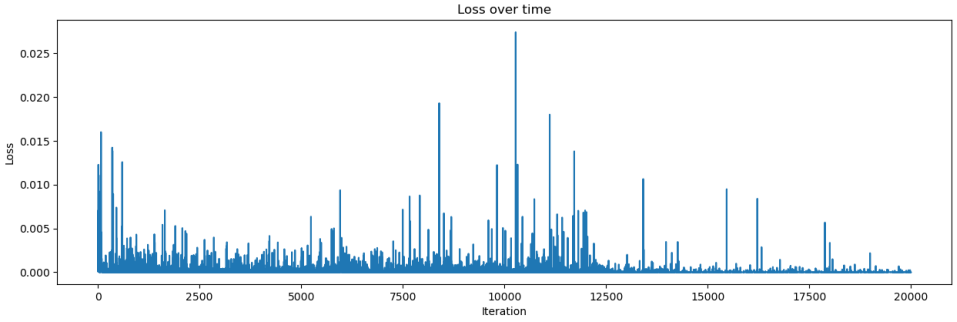

(a) Learned Value Function

(b) Learned Policy

Figure 2: Value and Policy as a function of position and velocity.




(a) Cost Over Episodes




(b) Loss over Episodes

Figure 3: Cost and Loss per episode.