ROB-GY 6323 Second Project Report

PROJECT 2 - Q LEARNING

Done by:
GORDON OBOH

Department of Mechanical & Aerospace Engineering
Tandon School of Engineering , NYU

December 21, 2022

Contents

1 Introduction

The goal of this project is to utilize Q-learning in inverting a pendulum.

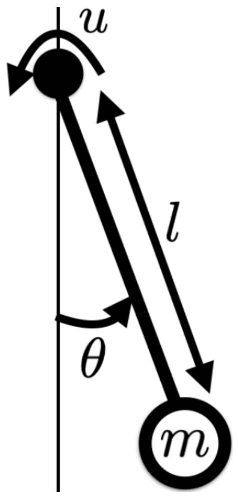

Figure 1: a diagram of a simple pendulum

This Project report serves to explain some of the steps taken in implementing the Optimal Control techniques used.

our states are given as:

$$x = \begin{pmatrix} \theta \\ \omega \end{pmatrix}$$

where:

$$\theta = [0, 2\pi) \qquad \omega = [-6, 6]$$

$$x_{n+1} = \texttt{get\_next\_state}(x_n, u_n)$$

Our goal is to minimize the discounted cost function in equation(1):

$$\sum_{i=0}^{\infty} \alpha^i g(x_i, u_i) \tag{1}$$

where:

$$g(x_i, u_i) = (\theta - \pi)^2 + 0.01 \cdot \dot{\theta}_i^2 + 0.0001 \cdot u_i^2 \qquad \text{and} \qquad \alpha = 0.99 \tag{2}$$

This cost in equation(2) penalizes deviations from the inverted position but also encourages small velocities and control.

2 Q-Learning with a Q-table

In implementing the learning algorithm the first step would be to discretize the state. For this project we would use 50 discretized states

2.1 Get Cost Function

For this function, equation(2) is implemented, where i serves as the index

2.2 Q-table

My implementation of the Q-table has a numpy dimension of $[3 \times 50 \times 50]$

- there are only the 3 possible options for controls.
- θ has been discretized into 50 points.
- ω has been discretized into 50 points.

2.3 Get Policy and Value Function

The Optimal value function and Optimal policy is given below:

$$J^*(x_t) = min_u Q(x_t, u)$$

$$\mu^*(x_t) = argmin_u Q(x_t, u)$$

2.5 Number of Episodes

with $u \in \{-4, 0, 4\}$ and a learning rate(γ) of 0.1, it takes approximately 2900 to 3200 episodes for Q-learning to learn how to invert the pendulum.

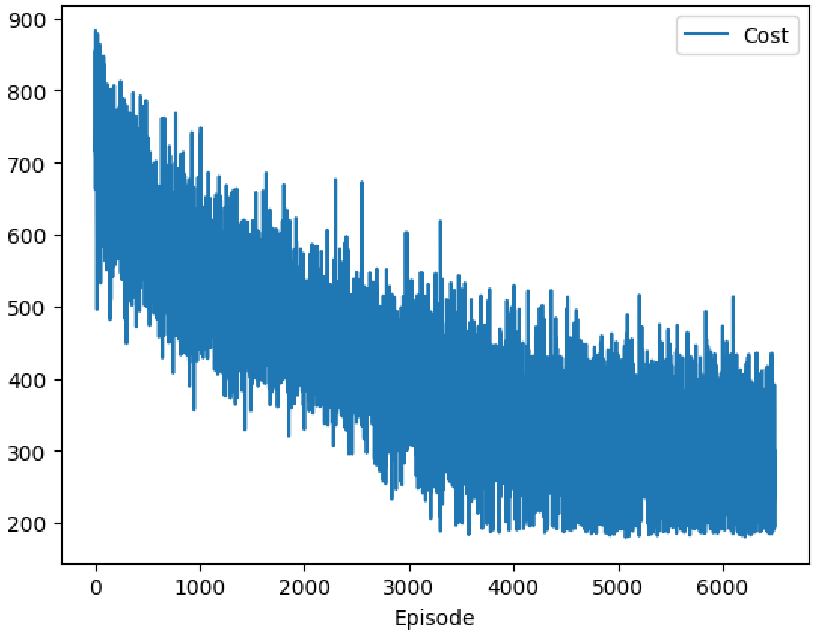

Figure 2: Plot of Learning Progress with $\gamma = 0.1$ starting from x=[0,0]

2.6 Time evolution of θ and ω

From x=[0,0] it takes 2 swings, i.e it passes [0.0] 2 times.

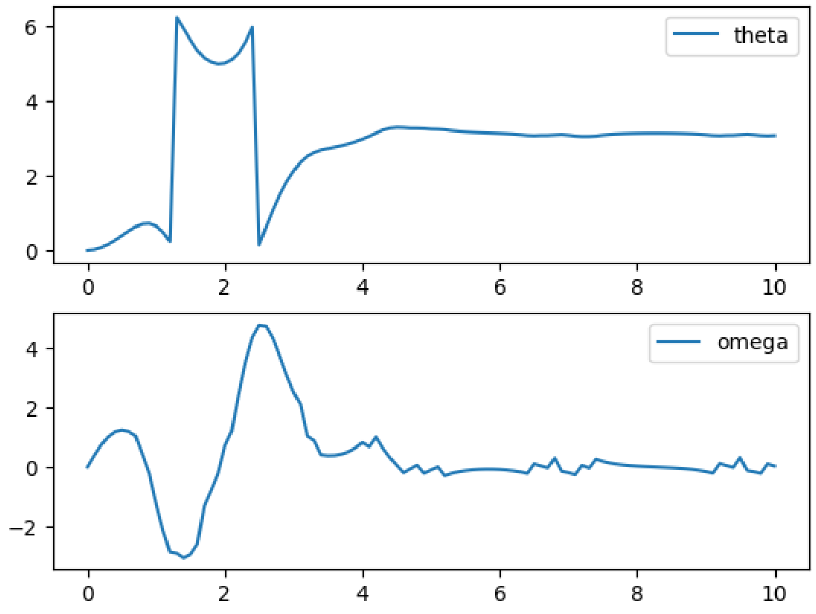

Figure 3: Time evolution of θ and ω with $\gamma = 0.1$ starting from x=[0,0]

2.7 Optimal Policy and Optimal Value Function

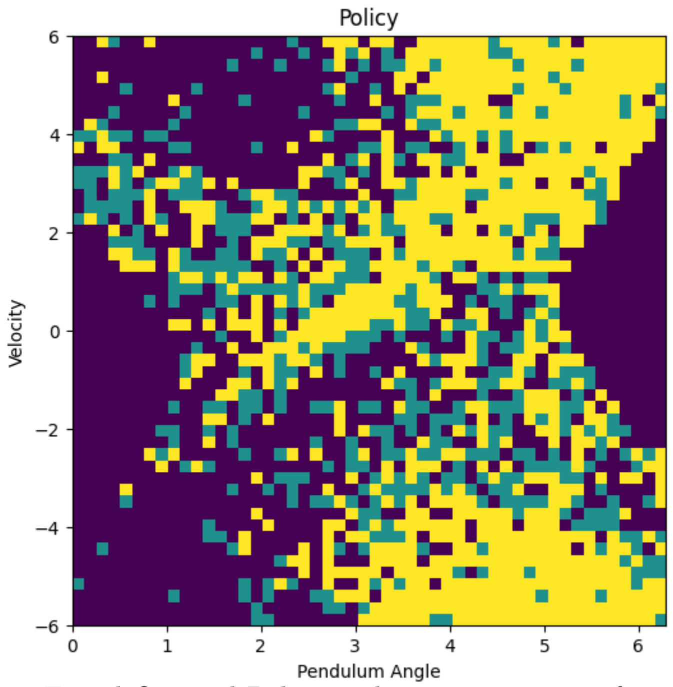

Figure 4: Found Optimal Policy with $\gamma = 0.1$ starting from x=$[0, 0]$

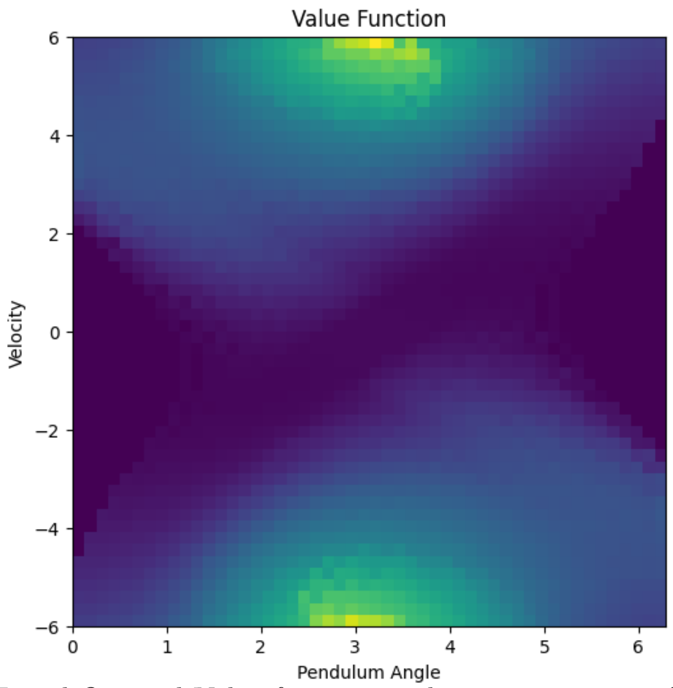

Figure 5: Found Optimal Value function with $\gamma = 0.1$ starting from x=$[0, 0]$

2.8 $u \in \{-5, 0, 5\}$

2.8.1 Number of Episodes

with $u \in \{-5, 0, 5\}$ and a learning rate(γ) of 0.1, it takes approximately 2500 to 3000 episodes for Q-learning to learn how to invert the pendulum.

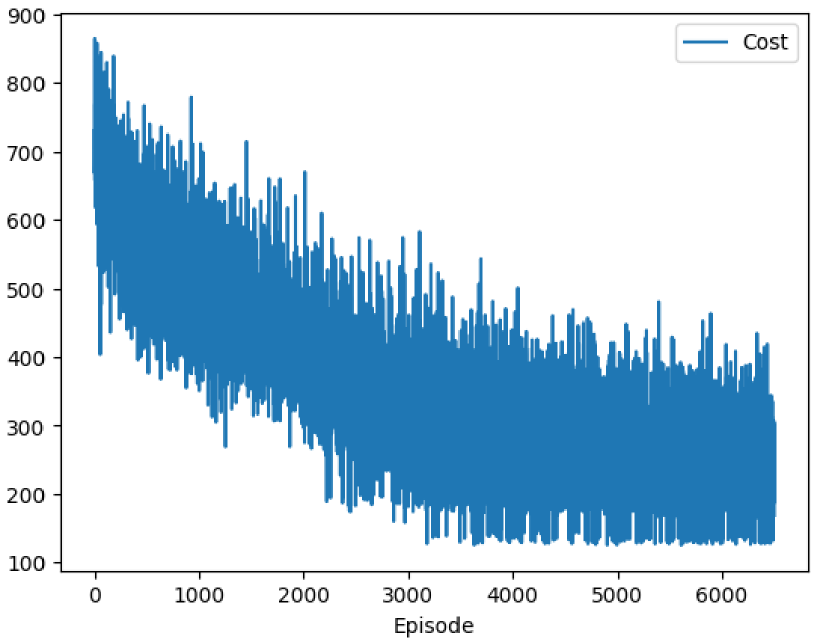

Figure 6: Plot of Learning Progress with γ = 0.1 starting from x=[0,0]

2.8.2 Time evolution of θ and ω

From x=[0,0] it takes 1 swing, i.e it passes [0.0] 1 time.

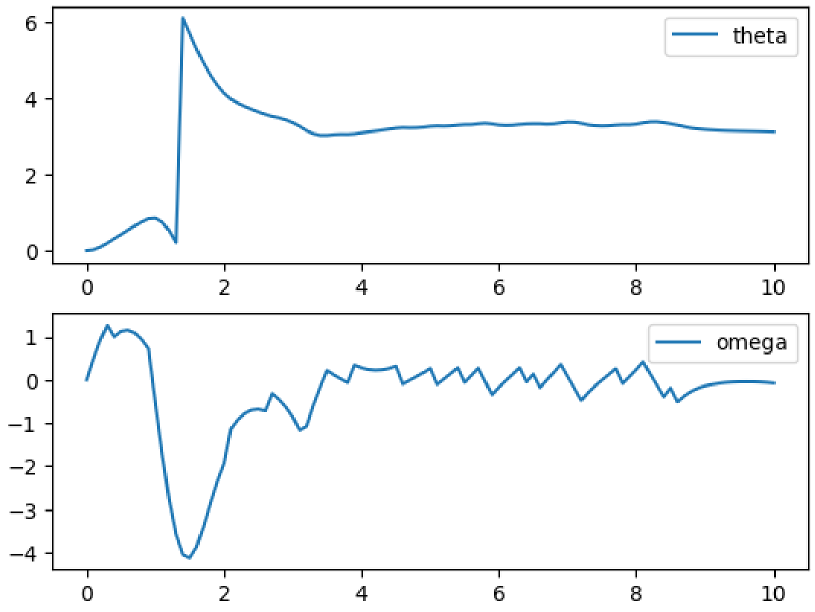

Figure 7: Time evolution of θ and ω with γ = 0.1 starting from x=$[0, 0]$

2.8.3 Optimal Policy and Optimal Value Function

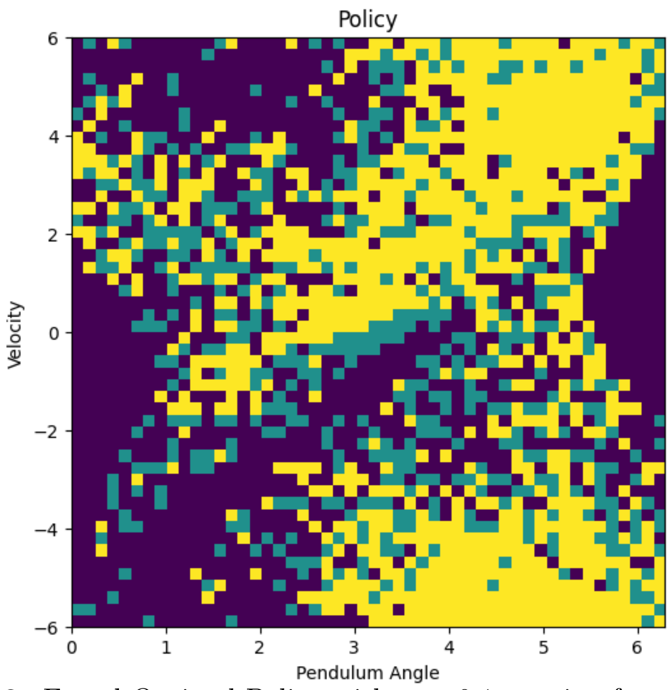

Figure 8: Found Optimal Policy with $\gamma = 0.1$ starting from x=$[0, 0]$

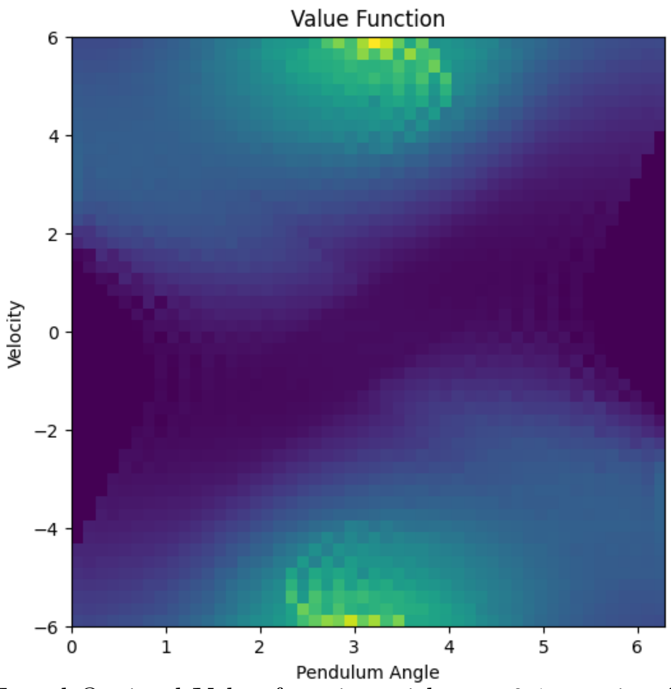

Figure 9: Found Optimal Value function with $\gamma = 0.1$ starting from x=$[0, 0]$

2.8.4 Quantitative Differences

Table 1: Tabular Quantitative Differences

$\boldsymbol{u \in \{-4, 0, 4\}}$	$\boldsymbol{u \in \{-5, 0, 5\}}$
Learning rate converges slower	Learning rate converges faster
uses more episodes	uses less episodes
requires more time to learn	requires less time to learn

Table 1 acts as a quick summary for the Quantitative differences between having higher values for the control input vs having lower values.
As shown in figure 2 and figure 6, the learning progress with lower control input values requires more time and more episodes to converge as opposed to the higher control input, this also affects how long it takes for the the pendulum to reach a fully inverted position. x_0 also affects both control input values, when $x_0 = [0, 0]^\top$ more time, iterations and swing may be required when compared to $x_0 = [1.4, 0]^\top$.

2.9 Changing ϵ and γ

- As the learning rate(γ) $\in$ [0 1] gets bigger the Learning Progress requires fewer iterations to converge.
- As $\epsilon \in$ [0 1] gets bigger the Learning Progress requires more iterations to converge.

2.9.1 Why?

- In Q-learning algorithm the learning rate determines the extent to which newly acquired information will override old information.
  A higher learning rate can lead to faster convergence and better performance in the short term, but it can also make the algorithm more sensitive to noise in the data and could lead to divergence or a sub-optimal solution. A lower learning rate can lead to slower convergence, but it can also make the algorithm more stable and less sensitive to noise.
- ϵ is a parameter that determines the probability of the algorithm choosing a random action (exploration) rather than the action with the highest expected reward (exploitation). The value of ϵ can have a significant effect on the performance of the algorithm.
  A high value of ϵ will cause the algorithm to explore more, which may be helpful in the beginning, but it can also slow down the learning process(slower convergence) and may prevent the algorithm from converging. A low value of ϵ will cause the algorithm to focus more on exploitation, which can lead to a faster learning process(faster convergence), but it can also cause the algorithm to get stuck in a sub optimal solution