Reinforcement Learning

– Project 2 –

submitted by

Sanket Ankush Salunkhe

Sanket Ankush Salunkhe
sas9908@nyu.edu
Student ID: sas9908

1 Learning to invert a pendulum:

Goal: To learn a policy for an inverted pendulum model to make it do a swing-up motion.

The state of the given pendulum system is:

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

The given loss function is:

$$g(x_i, u_i) = (\theta - \pi)^2 + 0.01 * \dot{\theta}_i^2 + 0.0001 * u_i^2$$

This cost will be used to penalize the deviations from the inverted pendulum position.

The limitation over state parameters are:

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

The discretized state parameters will use 50 discretized states for θ and 50 for ω

1.1 get_cost() function:

As described above, $g(x_i, u_i)$ gives the cost of state at each time-step. Its implementation is defined in the given .ipynb file.

1.2 Dimension of Q-table :

Q-table store the cost value for each pair of states and control (x, u). As said above there are a total of 50 discretized values of θ and ω which makes 50×50 of the state matrix. While there are 3 control parameters are available for each of those states.

Thus the final dimension of the Q-table is $[50 \times 50 \times 3]$.

1.3 get_policy_and_value_function(q_table):

We are using a cost-based Q-learning algorithm. Here to compute an optimal policy u from a Q-table, we need to choose an action that has a minimum Q-value $Q(x_t, u_t)$.

$$u_t = argmin_u Q(x_t, u)$$

The optimal value function is computed from Q-table, which has a minimal Q-value for each state. The optimal value function is the expected return for each state under the optimal policy.

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

The implementation of get_policy and value_function is explained in given .ipynb file.

1.4 q_learning(q_table):

In the Q-learning algorithm, the agent tries to learn to take actions that maximize the expected reward by updating the Q-values for each state-action pair based on the observed rewards $Q(x_t, u_t)$ and the estimated future rewards $Q(x_{t+1}, u)$.

The algorithm for Q-learning can be summarized in the following steps:

Algorithm 1 Q-learning

Initialize hyper-parameters $\gamma \in [0, 1]$, ϵ and α
Initialize $Q(x, u)$ for all states and controls
for *episode* in *episodes* **do**
    Initialize state x_0
    for t in *max_iteration* **do**
        Choose a control using ϵ-greedy policy from Q
        Get next state x_{t+1}
        Compute $g(x_t, \mu(x_t))$
        Calculate TD-error using α
        Update $Q(x_t, u_t)$
    end for
end for

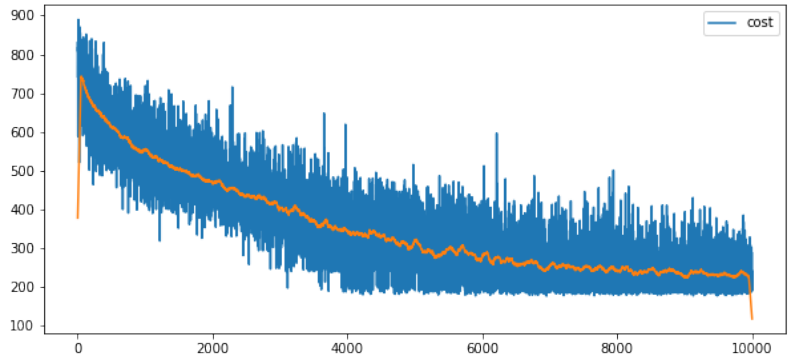
Where, Epsilon greedy policy can be defined as,

$$u_t = \begin{cases} arg\ \min_u Q(x_t, u) & if\ probability\ > \epsilon \\ random\ action & else \end{cases} \tag{1}$$

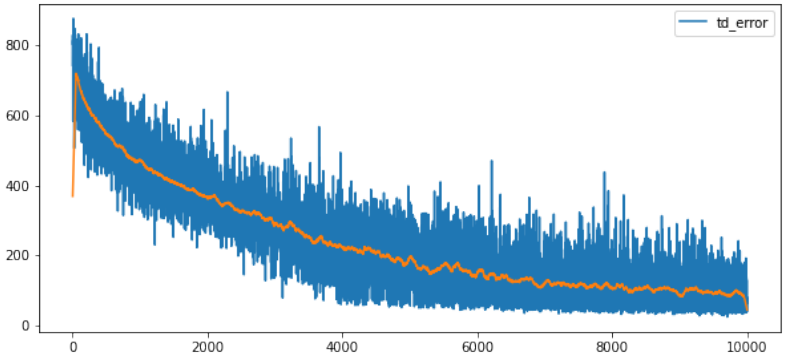
1.5 No of episodes:

Here we are training a Q-table for predefined 10000 episodes. From generated results and observation, for around 6000 episodes Q-learning algorithm learns to invert the pendulum.

From the graph attached below, we can see that the value of cost and TD error becomes constant after around 6000 episodes. The plot attached below in Figure 1: ((a) Loss VS Episodes (b) TD Error VS Episodes) shows the learning process in terms of cost value and TD error.



(a)

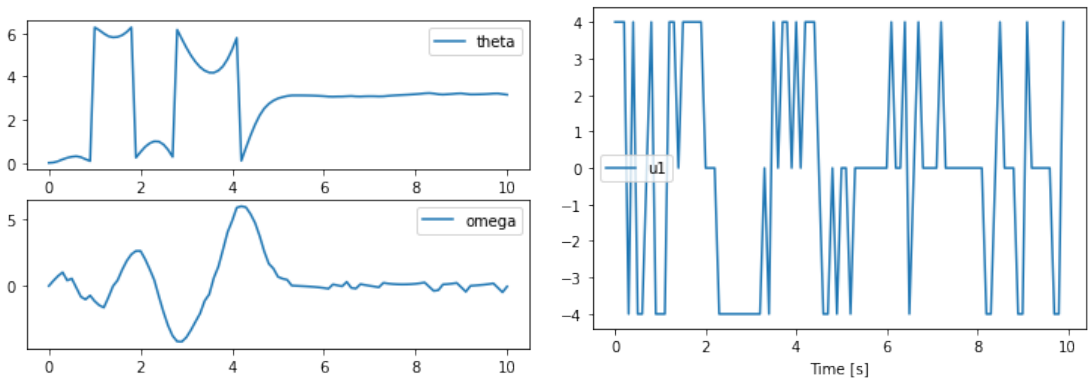


(b)

Figure 1: (a) Loss VS Episodes (b) TD Error VS Episodes

1.6 Plot of θ and ω:

For given 10000 episodes of training, the pendulum makes about three back-and-forth motions and reaches the desired inverted state from $x = [0, 0]$. This number changes at each training cycle. The plots attached below in Figure 2: ((a) State VS Time (b) Control VS Time) show the evolution of θ and ω w.r.t. time.

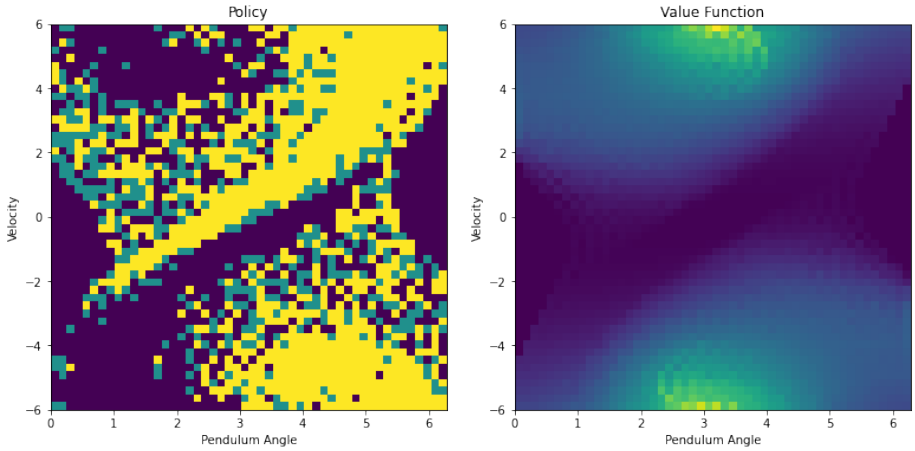


(a) (b)

Figure 2: (a) State VS Time (b) Control VS Time

1.7 Plot of policy and value function:

The plots attached below in Figure 3: ((a) Policy (b) Value Function) showcase the policy and value function.

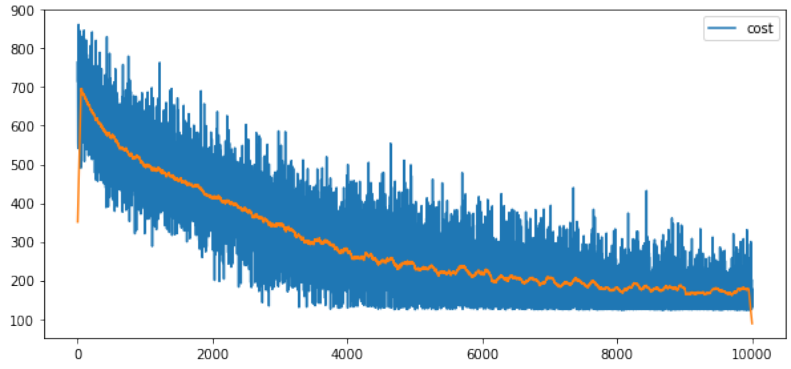


(a) (b)

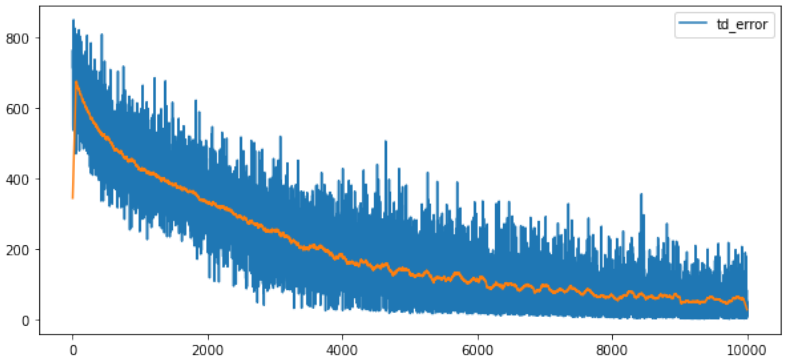
Figure 3: (a) Policy (b) Value Function

1.8 System at $u = [-5, 0, 5]$:

For an inverted pendulum system at $u = [-5, 0, 5]$ it takes around 5000 episodes to attain the desired position. From both graphs of different control we can see that, for an equal number of episodes, the current system of $u = [-5, 0, 5]$ has less TD-error value than the previous system of $u = [-4, 0, 4]$. So we can say that the current system converges in fewer episodes as compared to the previous system. The plot attached below in Figure 4: ((a) Loss VS Episodes (b) TD Error VS Episodes) shows the learning process in terms of cost value and TD error for $u = [-5, 0, 5]$.



(a)

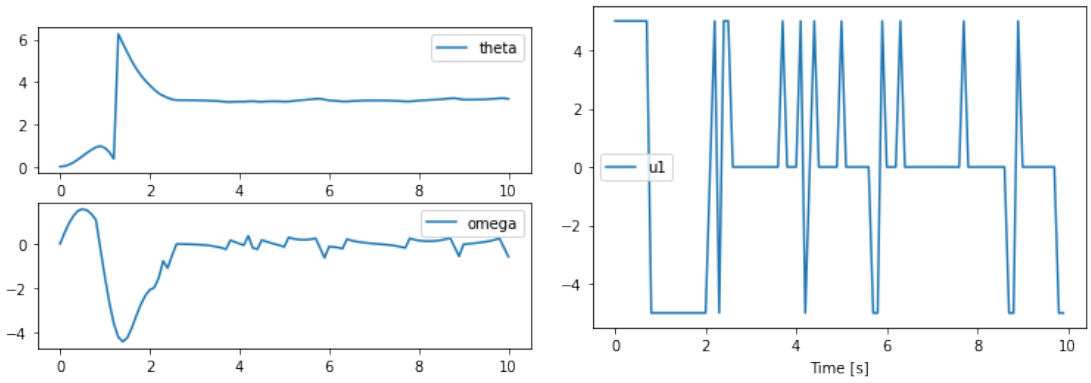


(b)

Figure 4: (a) Loss VS Episodes (b) TD Error VS Episodes

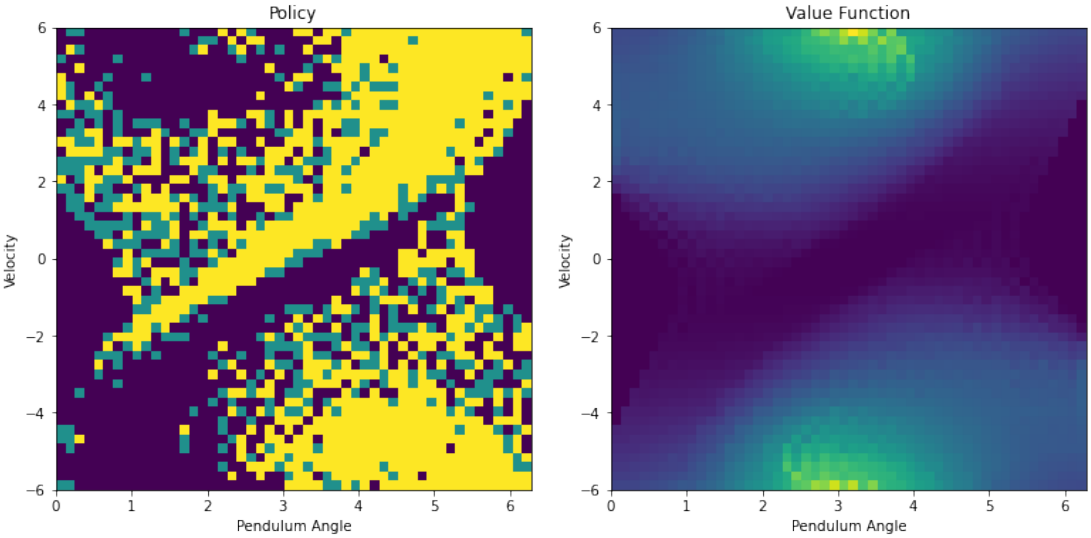
Also, the current system takes only one back-and-forth motion to reach the desired inverted state from $x = [0, 0]$. The plots attached below in Figure 5: ((a) State VS Time (b) Control VS Time) show the evolution of θ and ω w.r.t. time for the current system.

The plots attached below in Figure 6: ((a) Policy (b) Value Function) showcase the policy and value function for the current system.



(a) (b)

Figure 5: (a) State VS Time (b) Control VS Time



(a) (b)

Figure 6: (a) Policy (b) Value Function

1.9 Effect of ϵ and learning rate:

The epsilon-greedy policy is used to make a balance between exploration and exploitation. It determines which Q-values are updated based on the new experience.

The learning rate determines the extent to which agents' Q-values will update based on the new experience. The higher the learning rate, the agent will learn quickly in fewer epochs. But this higher learning rate may cause overfitting also. With a lower learning rate agents will learn slowly and will be stable.

Thus the value of Epsilon and LR affect the performance of the Q-learning algorithm. For high epsilon value, the agent may explore and will not exploit its current knowledge. If the value of epsilon is low, the agent will become greedy and not explore enough. Similarly for high LR. the agent may not converge to optimal value and become too much sensitive. For low LR, the agent may take too long to learn the optimal Q-value.