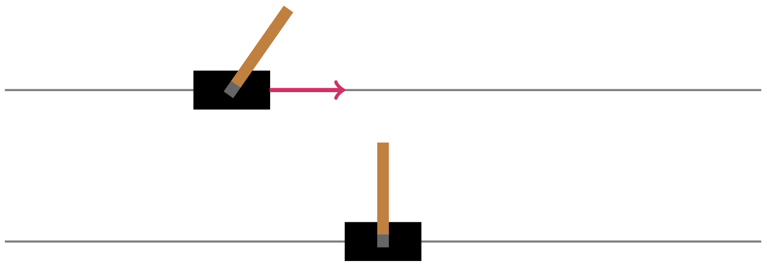
Inverted Pendulum - Cart and Rod

Michael Hansen, Brooke Mosby, Juan Rodríguez, Bailey Smith

April 10, 2018

1 Problem Statement

You will write code to interact with the provided virtual inverted pendulum. Your program should work for a pendulum of arbitrary mass and length. Given the mass and length of the pendulum and an initial position, angle, approximate change in position, and approximate change in angle, you will move the base of the pendulum horizontally to bring the pendulum to within a given tolerance of an unstable steady state of the system. The mass and length of the pendulum are constrained in a certain interval. There is also a limit on your control, which your code is given at run time.



2 Requirements

1. Report 50 points
   - (a) 20 points for clearly describing your algorithm for solving the problem
   - (b) 10 points for making the argument that your algorithm is in some sense optimal
   - (c) 10 points for fair share of the work load
2. Coding 100 points
   - (a) 80 points for achieving the goal of bringing the pendulum to within tolerance of an unstable steady state within a reasonable amount of time

(b) 20 points for optimality

3 Our Algorithm

We used code that we had implemented in the Inverted Pendulum Lab. We are given M (the mass of the cart), m (the mass of the rod), l (the length of the rod), q_1, q_2, q_3, q_4 (which are nonnegative weights used in the diagonal of the matrix Q), and r (which is the nonnegative weight making up the matrix R). Along with these parameters, we use the function `linearized_init` to construct and return the matrices A, B, Q, and R. The formulas for the matrices were found from these equations:

$$F = M\ddot{x} + b\dot{x} + N \tag{1}$$

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{2}$$

$$\tag{3}$$

Which become:

$$(I + ml^2)\ddot{\theta} - mgl\theta = ml\ddot{x} \tag{4}$$

$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\theta} = u \tag{5}$$

$$\tag{6}$$

With $I = \frac{1}{3} * m * l^2$. So the matrices are:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix}$$

$$Q = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 1 \\ 0 & 0 & 0 & q_4 \end{bmatrix}$$

$$R = \begin{bmatrix} r \end{bmatrix}$$

Next, we implement a function called `rickshaw` which takes in parameters tv (an array of time values), $X0$ (initial conditions on the state variables), A, B, Q, R (which we get from `linearized_init`), and a matrix P. P is calculated using `scipy.linalg.solve_continuous _are` which solves the algebraic Riccati equation given A, B, Q, and R. `rickshaw` is a Linear Quadratic Regulator which will calculate and return the optimal control U and the optimal state vector Z. This is all done inside our function `stabilize`.

To stabilize the rod on the cart vertically, we first initialize $X0$ using `env.reset()`. We give it our optimized qs and r that will be explained in the next section. We get Z and U using our `stabilize` function. Then, for each step in time (.02 seconds), we take the observation from `env.step()` on the first element of U, use it to contruct a new $X0$, then call `stabilize` for a new U and Z. U actually calculates the expected optimal controls for the rest of time, however we update after taking each next step. So, we use the first element of U in each iteration.

4 What Makes it Optimal?

To optimize our method, we used the LQR algorithm. The algorithm is an automated method to find the state-feedback controller, which gives very efficient feedback gains. Choosing the best parameters for our algorithm was very time consuming and required implementing our algorithm as a sklearn object. After creating our method as a sklearn object, we were able to run GridSearchCV to optimize over all parameters, q1, q2, q3, q4, and r to get the parameters the yielded the least amount of iterations. q1 was left as zero, because this parameter determines where the pendulum is balanced, which is not a parameter that is important in our case. Because we are using the parameters the yield the least iterations, our method has been optimized.