CS229 Term Final Project Report

Hasan Unlu

June 12, 2019

Controlling Two Wheels Balancing Robot Using Reinforcement Learning

The purpose of the project is implementing reinforcement learning which is going to replace PID controller loop in self balancing robot. Before starting the implementation, PID controller shown in Figure 1 is implemented and operated on the robot.

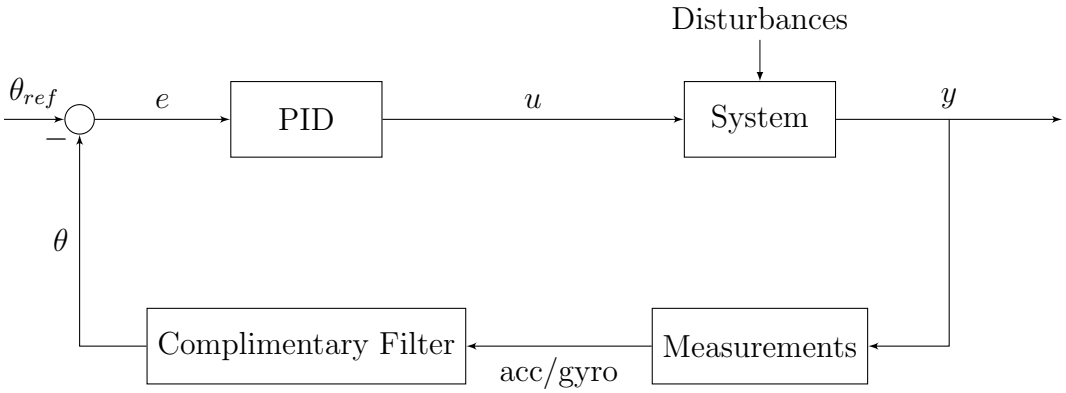

Figure 1: Control Diagram

PID controller tries to control robot comparing the measurements with reference angle. In this design, reference angle is 0 (vertical position). To be able to feed vertical angle, sensor readings need to be filtered. The sensor(Invense MPU6050) gives raw acceleration and angular velocity of the

robot. One the way to filter raw acceleration is using Complimentary Filter [1]. Complimentary filter equation is shown in below. Calibrated α is 0.98 for this sensor.

$$\theta = \alpha(\theta + gyro * dt) + (1 - \alpha)accelerometer$$

After the controller loop, PID results are converted into PWM by linearly proportional. 0-100% percent PMW is mapped to PID output. This part needs to be revisited, because discrete outputs are going to be used in reinforcement learning like backward and forward with fixed PWMs.

Binary PID Outputs

Before implementing the machine learning algorithm, discretized binary outputs tested on robot to see dynamics of the robot is applicable for binary control. PID output is passed through sign function and depending on the results -60% or +60% of PWM are applied to motors. It has been implemented and tested on the target robot platform, it is working as expected, but movements are drastic. Video link `https://www.youtube.com/watch?v=OYPt29tfekA`

Simulation Platform

Cart-Pole-v1 is used as a simulation platform [2]. This platform has 20 miliseconds sampled inverted pendulum dynamics. It doesn't have sensor models and the measurements(angular velocity and position) are ideal.

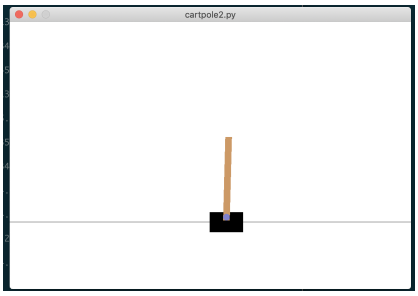

Figure 2: Cart-Pole-v1 platform in open.ai gym

Implementing Q Learning

The proposed architecture using machine learning is shown in Figure 3. Reinforcement learning will decide 0(backward) and 1(forward) depending on interpreting state of the pole. For state representation, θ and $\dot{\theta}$ are used. These continuous states are discretized into 4 bit representation for each.

s = [8-bit state] = [4-bit — 4-bit] = [θ — $\dot{\theta}$] = [0..255]

State 256 is terminal state.

Action space is only backward($a = 0$) and forward($a = 1$).

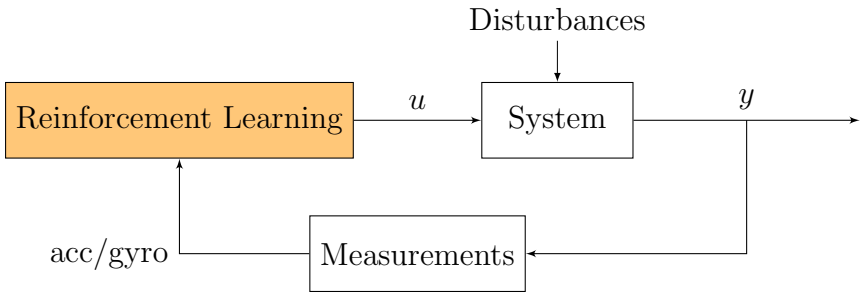

Figure 3: Proposed Architecure

Q-Learning equations:

$\pi(s) = a' = argmax_a \sum_{s'} Q(s, a)$

$Q(s, a) = Q(s, a) + \alpha * (R(s) + \gamma * Q(s', a') - Q(s, a))$

Where α is learning rate and γ is discount factor. a' is next best action to choose. s' is the state after action a happens.

In addition to traditional Q-Learning algorithm, rewarded table elements are still emphasized even if current step doesn't update them. This trick decreased learning time. But, as a disadvantage of this, it is not possible to change previously learnt structure quickly.

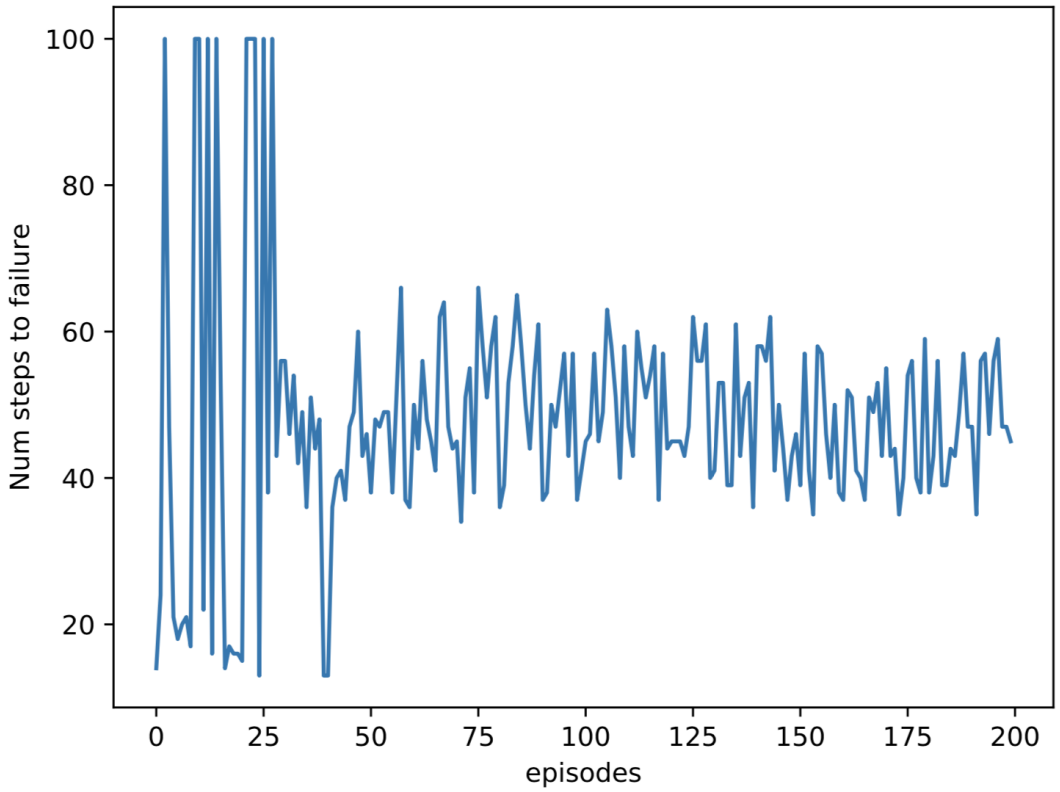

Figure 4: open.ai gym simulation results

Number of steps to failure is getting increase and learning happens. After episode 50, it always survive more than 40 steps. In this simulation, the limits on linear position and velocity of cart are removed. The operation(or rewarded) states are between -24 and 24 degrees for θ.

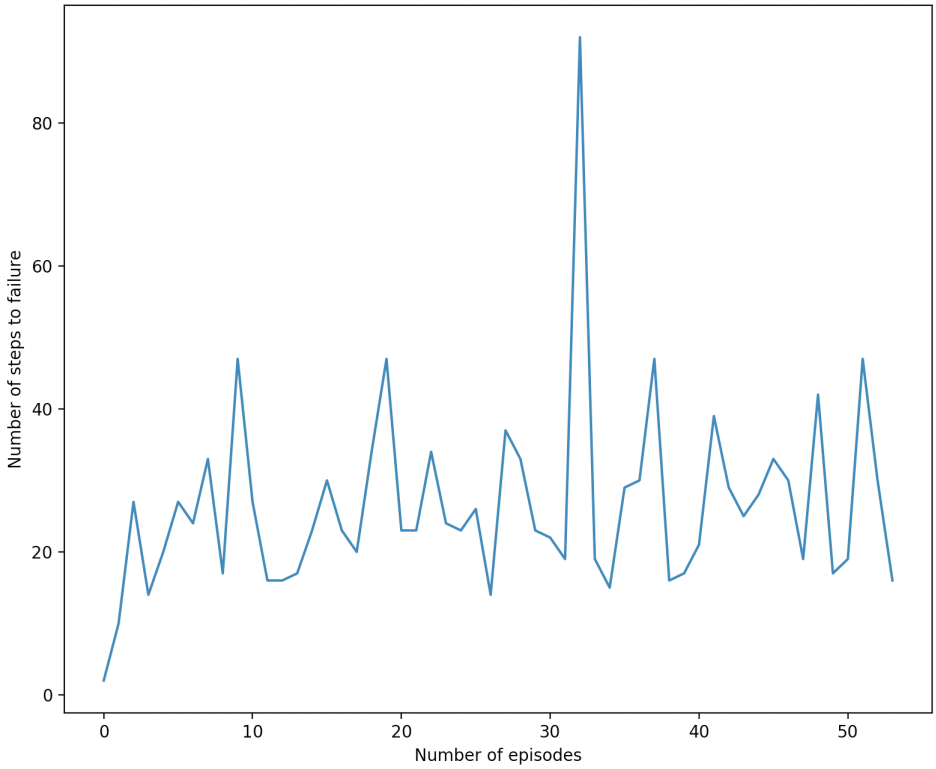

Figure 5: Data collected from the platform

The code running on the platform is relatively different on step start condition. Motors start acting when robot is brought to operation range in vertical angle position which is between -24 to 24 degrees. If it goes beyond, motors stop and episode ends. Data on the platform shows learning happens. Because, after a couple episode, it keeps itself average of 20 steps for each episode. But gyro noise is causing lots of different state transition and the robot couldn't make it more than 50 consistent steps before the fail. The code on the platform still uses complimentary filter to get angular position of the robot. In the next section, raw data measurement fed model will be discussed. Video link for implemented Q-Learning on the platform `https://www.youtube.com/watch?v=bV1DGn6hkEA`

Modeling Accelerometer in open.ai gym

Raw accelerometer measurement is composition of cart acceleration, gravity and centrifugal force created by pole movement. Sensor on the robots measures sum of those quantities. I added a raw accelerometer model to open.ai gym model and simulate the system. This approach is challenging for defining terminal state. Unfortunately raw accelerometer and gyro data itself has no way to define terminal state. Accelerometer model created but still vertical angle in simulation used for guiding the terminal state. It is not completely actual model but still shows there is capability of learning.

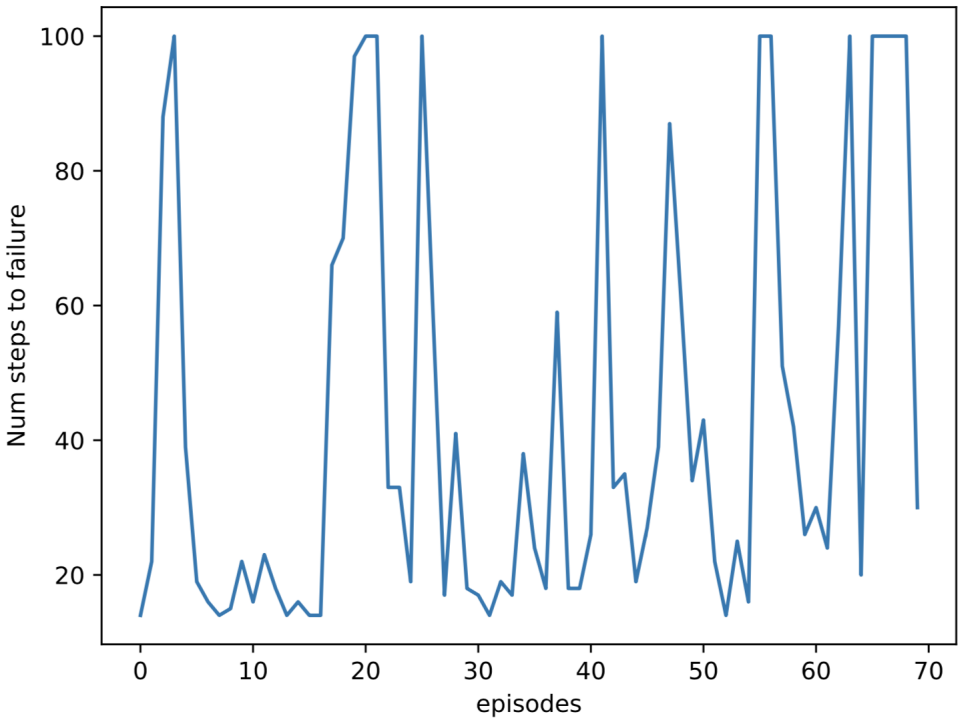

Figure 6: Learning curve using raw accelerometer model

Simulation and platform codes have been uploaded here. `https://github.com/hasanunlu/cs229`

Robotic Platform

The platform is composed of the following equipments,

-Two wheeler robot chassis from OSEEP.

-MPU-6050 Invensense 6-axis accelerometer and gyro.

-Raspberry Pi 3 Model B+ compute unit

-L298 motor driver

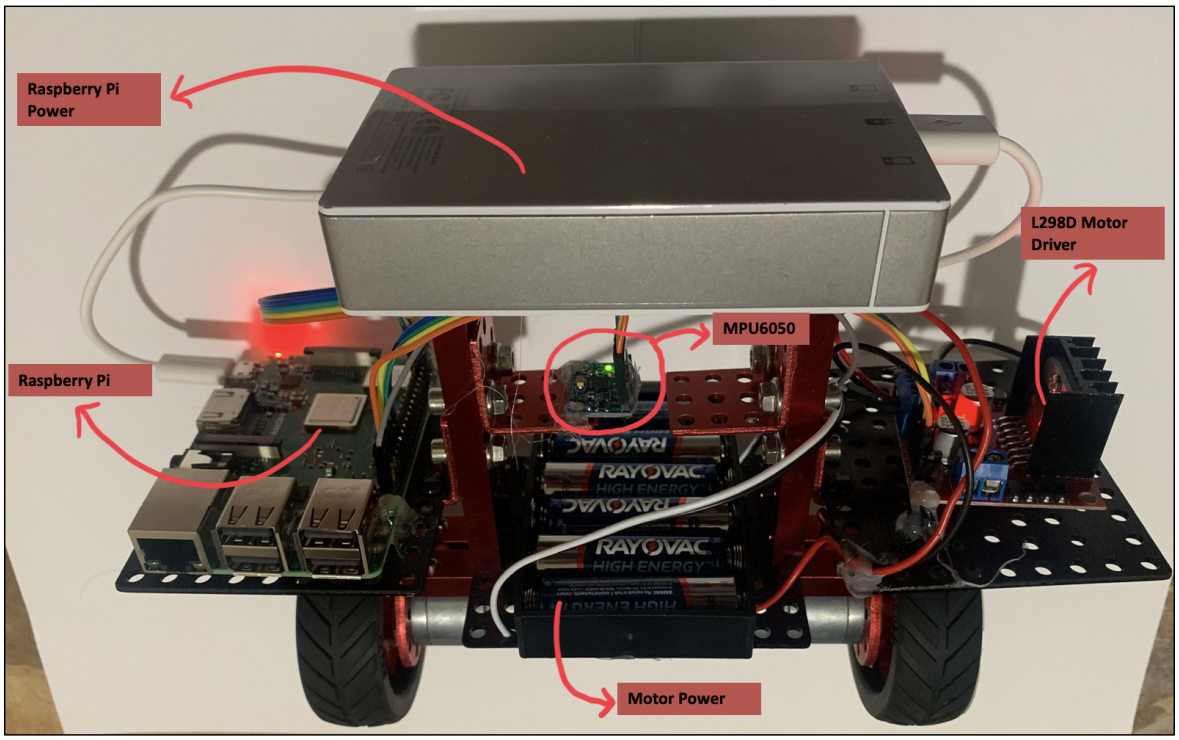

Figure 7: Robot Platform

References

[1] Aircraft Stability and Control 16.333, Lecture #15, MIT
[2] `https://gym.openai.com`
[3] Watkins & Dayan (1992) Q-Learning, Machine learning, 8(3), 279-292.