Controlling Two-Wheels Self Balancing Robot Using Q-Learning

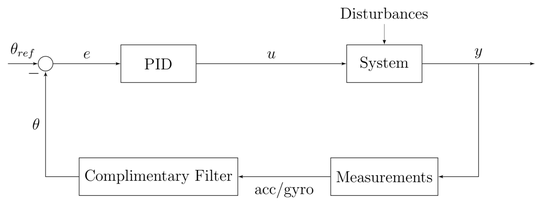
Hasan Unlu, hunlu@stanford.edu, Project ID: 94

Problem

Implementing Q-Learning to control two wheels self balancing robot. Mainly PID control loop will be replaced by reinforcement learning.

PID Loop

Deterministic control loop is shown below. Based on robots vertical angle PID controller produces PWM to control motors. In this design there is no linear position control. Only vertical angle is being tried to keep in reference range.



Complimentary Filter

It combines raw accelerometer and angular velocity data and calculates angular position. Basic intuition is that if there is less movement or static, trust accelerometer result, otherwise integrate measured angular velocity and add it to last angular position **[1]**.

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

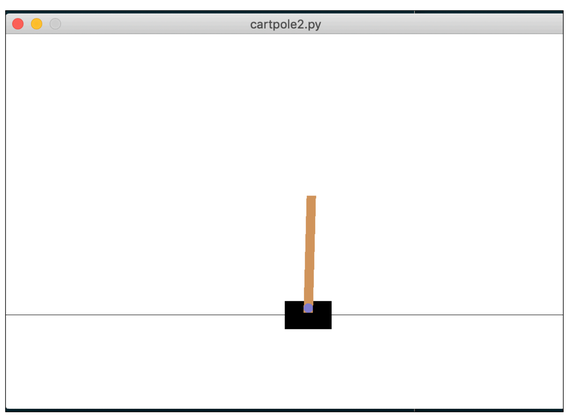
Binary PID output

The first step to prove before implementing Q-learning is action selection. Only **backward(a=0)** and **forward(a=1)** are going to be used in Q-Learning. Before implementing Q-Learning, regular PID output is converted into fix PWM rated motor outputs. In this video binary PID output produces only -1 and 1 to control robot. %70 PWM is selected per 1 or -1.

https://www.youtube.com/watch?v=OYPt29tfekA

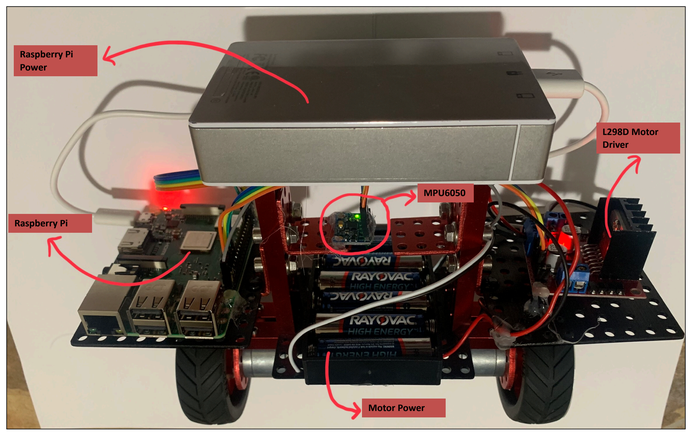
Simulation Platform

Cart-Pole-v1 in ***gym.openai.com*** is used to simulate the system **[2]**.



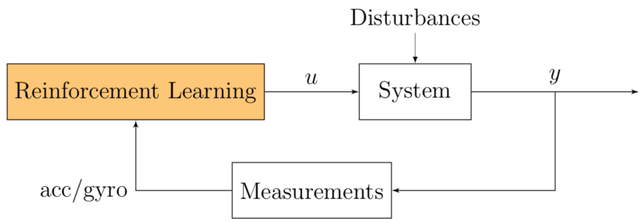
Robotic Platform

- Two wheeler robot chassis from OSEEP.
- MPU-6050 Invensense 6-axis accelerometer and gyro.
- Raspberry Pi 3 Model B+ compute unit.



Machine Learning Architectures

- Raw measurements only (angular velocity and unfiltered accelerometer)
- Calculated angular position from Complimentary filter and angular velocity.



Q-Learning

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

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

Where $\boldsymbol{\alpha}$ is learning rate and $\boldsymbol{\gamma}$ is discount factor. $\boldsymbol{a'}$ is next best action to choose. $\boldsymbol{s'}$ is the state after action $\boldsymbol{a}$ happened **[3]**.

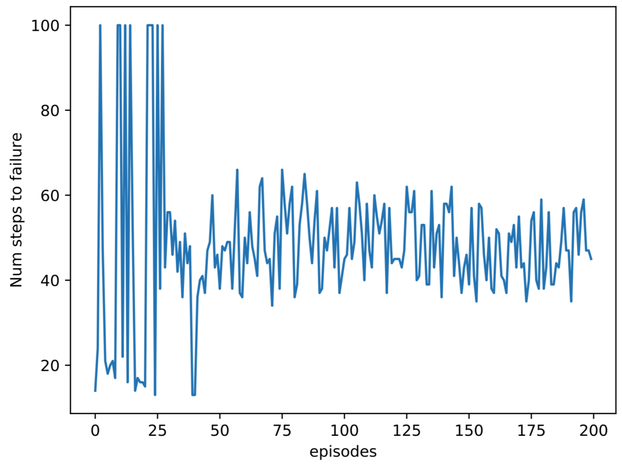
Only angular speed($\dot{\boldsymbol{\theta}}$) and angular position($\boldsymbol{\theta}$) of the robot are used for discreting state generation. 257 total states are used. State 0 to 255 are for rewarded states and 256 is for terminal or exit state.

[8-bit state definition] = [4-bit for $\boldsymbol{\theta}$ | 4-bit for $\dot{\boldsymbol{\theta}}$]

Operating $\boldsymbol{\theta}$ is between -24 and 24 degree.
Operating $\dot{\boldsymbol{\theta}}$ is between -200 and 200 degree/seconds.

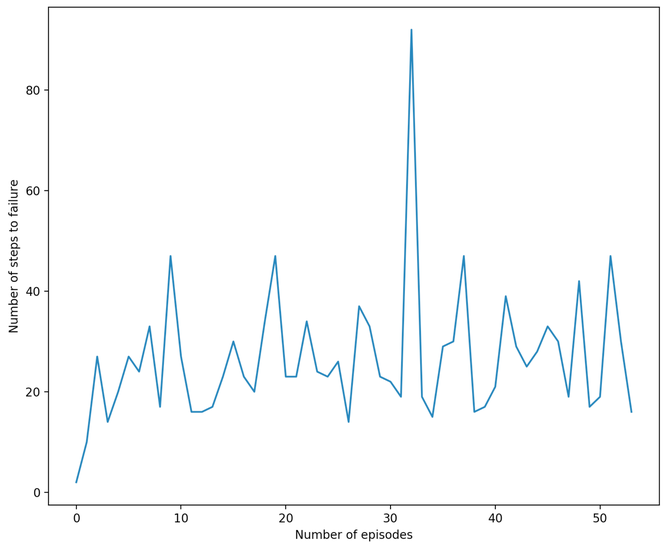
Simulation Performance

The proposed setup using angular position and angular velocity gave the following learning curve.



Robotic Platform Demo

Same algorithm tested on the simulation is implemented on robotic platform. The platform runs 52 episodes and it stops updating Q table. Each episodes start when robot positioned in operating $\boldsymbol{\theta}$ range and stop when it reaches terminal state. At first, robot tries continuing backward and forward then it quickly tries different set of backwards and forwards back to back.



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.