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].

\[
\theta = \alpha(\theta + \text{gyro} \times dt) + (1 - \alpha)\text{accelerometer}
\]

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=03PlC29fF0kA

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.

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

References

Q-Learning
\[
\pi(s) = a' = \arg\max_a \sum_{n=0}^{N} Q(s,a)
\]

\[
Q(s,a) = Q(s,a) + \alpha (R(s) + \gamma \times Q(s',a') - Q(s,a))
\]

Where \(\alpha\) is learning rate and \(\gamma\) is discount factor. \(a'\) is next best action to choose. \(s'\) is the state after action \(a\) happened [3].

Only angular speed(\(\dot{\theta}\)) and angular position(\(\theta\)) of the robot are used for discretizing 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 \(\theta\) | 4-bit for \(\dot{\theta}\)]

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