0 Contents

Explain the assumptions behind the theoretical development you are using and the application of the theory to your particular problem. Any heavy algebra or details of computing work should go into an appendix. This section should describe the running of the experiment or experiments and what equipment was used, but should not be a blow by blow account of your work. Experimental accuracy could be discussed here.

0.1 The Inverted Pendulum (IP)

The Inverted Pendulum is an inherently unstable system with highly nonlinear dynamics and is under-actuated.

0.1.1 Dynamics

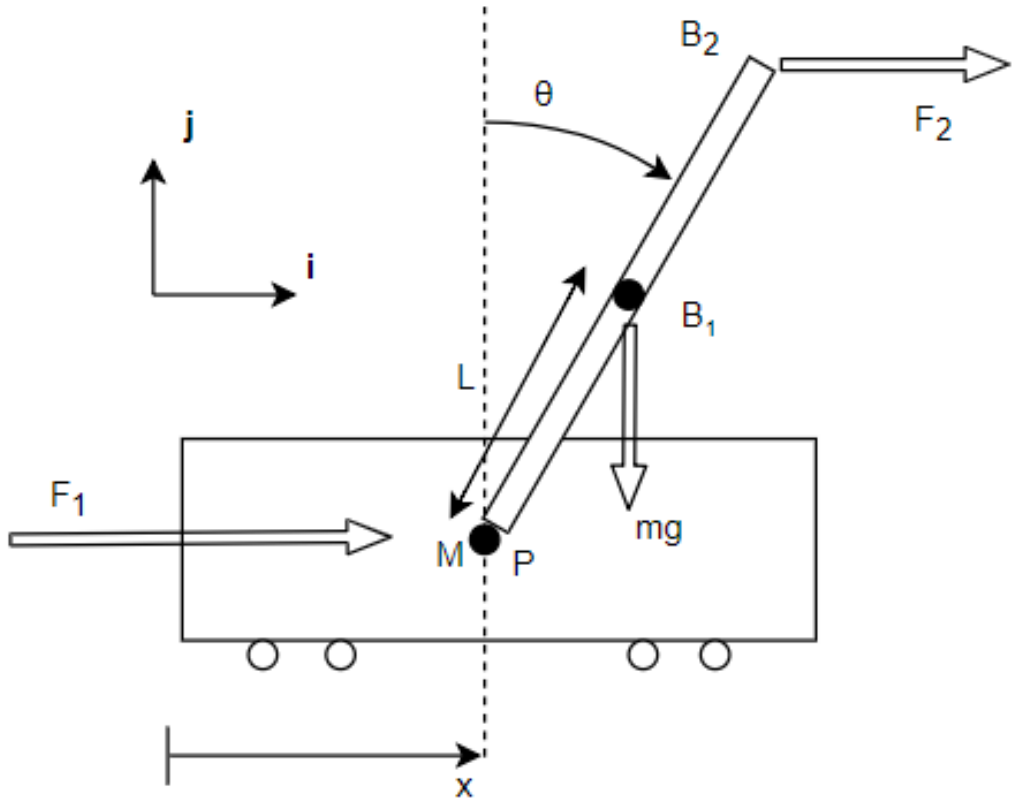


Figure 1: A free-body diagram of the inverted pendulum system. For the OpenAI IP the system is in discrete time with a time-step of $\tau = 0.02s$. The other constants are $l = 0.5m$, $m = 0.1kg$, $M = 1kg$, $F = \pm 10N$, $x_{max} = \pm 2.4m$, $\theta_{max} = \pm 12^o$.

The full state space equations for the inverted pendulum as defined in fig. 1 are given by:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{\left(\frac{2M-m}{m}F_2 - F_1\right)cos\theta + g(M+m)sin\theta - mL\dot{\theta}^2 sin\theta cos\theta}{(M+msin^2\theta)} \\ \dot{\theta} \\ \frac{F_1 + F_2 cos(2\theta) + msin\theta(L\dot{\theta}^2 - gcos\theta)}{L(M+msin^2\theta)} \end{bmatrix} \tag{1}$$

Ignoring second order terms and linearising about $\boldsymbol{x}_e = [x_e, \dot{x}_e, \theta_e, \dot{\theta}_e]^T = [0, 0, 0, 0]^T$:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{\frac{2M-m}{m}F_1 - F_2 + g(M+m)\theta}{M} \\ \dot{\theta} \\ \frac{F_1+F_2-gm\theta}{lM} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & g\frac{M+m}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{mg}{lM} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\frac{1}{M} & \frac{2M-m}{Mm} \\ 0 & 0 \\ \frac{1}{lM} & \frac{1}{lM} \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \tag{2}$$

Which, as expected, is unstable since $det(\lambda I - A) = 0 \implies \lambda^2(\lambda^2 + \frac{mg}{lM}) = 0$. Note, for small angles the natural frequency of a non-inverted pendulum is $\omega_n = \sqrt{\frac{mg}{lM}} = \sqrt{\frac{0.1 \times 9.81}{0.5 \times 1}} \approx 1.40 rad/s$. Therefore, the time constant for the system is $\tau \approx 0.70s$.

OpenAI's gym is a python package that supplies an inverted pendulum environment built-in. This environment was wrapped to use the dynamics above and other extra functionality, whilst providing a rendering function shown in figure 2.

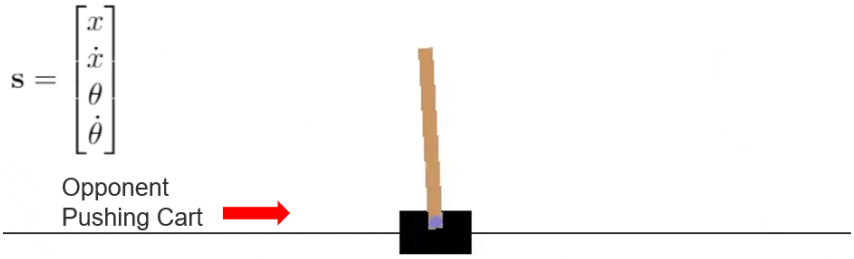


Figure 2: The OpenAI gym CartPole environment. The classical state representation is shown in the top left. Actions by the player and the adversary are taken as an impulse to the left or right as defined in 1.

0.1.2 Cost and Value Function

For each step/impulse, the 2D state is calculated and a cost, is calculated as:

$$c(x_t, u_t) = -\frac{1}{\sum_i w_i}\boldsymbol{w} \cdot \left[\left(\frac{x_t}{x_{max}}\right)^2, \left(\frac{\dot{x}_t}{\dot{x}_{max}}\right)^2, \left(\frac{\theta_t}{\theta_{max}}\right)^2, \left(\frac{\dot{\theta}_t}{\dot{\theta}_{max}}\right)^2\right]^T \tag{3}$$

Where $\boldsymbol{w}^T = [w_1, w_2, w_3, w_4] = [0.25, 0.1, 0.7, 1]$ and $0 \geq c(x_t, u_t) \geq -1$. The weights, $\boldsymbol{w}$, were chosen through empirical measurement of the the importance of each state ***. Weighting on the inputs was set to zero, as there are only two inputs for this problem, thus the cost can be written as $c(x_t)$. The max values can be approximated experimentally (note, $x_{max} = 2.4$ and $\theta_{max} = 12^o$ are given constraints):

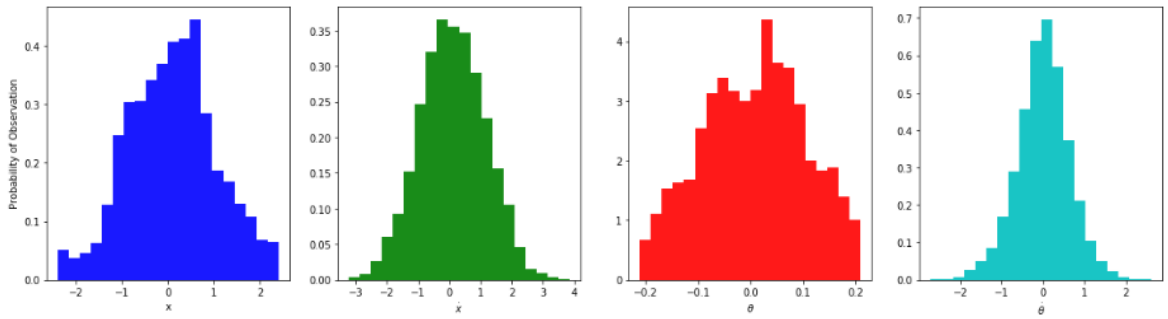


Figure 3: Histograms of typical state values. The frequencies greatly depend on the quality of the controller, with better controllers giving much narrower distributions. However, these are typical for a controller of medium efficacy over many episodes where the starting state is randomised (possibly in an uncontrollable position)

.

Suitable estimates for the the values of $\dot{x}_{max}$ and $\dot{\theta}_{max}$ are thus $\approx x_{max}$ and 2 respectively.

The value function is computed after an episode has completed as the discounted future losses at each state with the constraint that $\gamma^k < \frac{1}{20}$, where $\frac{1}{20}$ was chosen as it is a standard factor for insignificance. Since steps_beyonds_done (= k) must be defined in the CartPoleWrapper class, this is a constant, and therefore γ is calculated as $\gamma < \frac{1}{20}^{\frac{1}{k}}$. The discounted values are calculated using a geometric series:

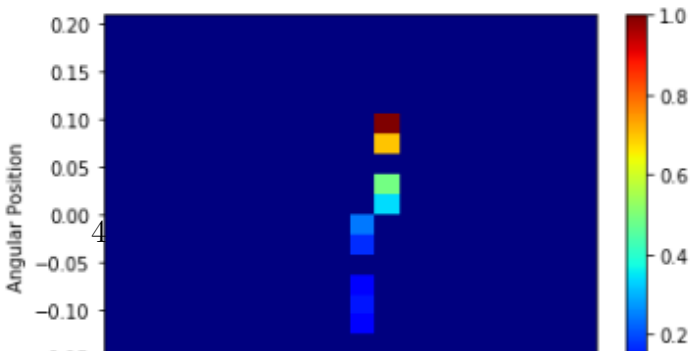
$$v_0 = \frac{\sum_{\tau=0}^{k} \gamma^\tau c(x_\tau)}{\sum_{\tau=0}^{k} \gamma_\tau}, \qquad \text{where } \gamma^k < \frac{1}{20} \tag{4}$$

Where for simplicity of notation, $v_0 = v(t)$, the state value at step t.

0.1.3 State Representations

The state can be represented in a number of ways, most simply this would just be feeding $\boldsymbol{x} = [x, \dot{x}, \theta, \dot{\theta}]$ into the neural network. This has a number of advantages such as lower computational cost, greater numerical accuracy (if the process is fully observable) and simpler implementation. Conversely, following Silver et. al, a 2D representation can be used. There are several possibilities for this, all of which first require binning $\boldsymbol{x}$:

(1) A matrix stack of x vs $\dot{x}$ and θ vs $\dot{\theta}$, both of which would only have one non-zero entry. This scales as b^n where b =



number of bins and n = number of states.

(2) A matrix stack of x_t vs x_{t-1} for all states. Similarly this scales as b^n, however the derivative states do not need to be plotted as these can be inferred. This has the advantage that, if the derivatives are not observable, we can build them into the 2D representation, however, if they are observable then this is less accurate than (1).

(3) A matrix of discounted previous states making a kind of motion history image. This is the formulation used, and shown in figure 4.

A 2D representation like this allows us to use a convolutional neural network,which has the benefit of various transformation invariances - these are particularly useful for the inverted pendulum since it is symmetric.

The state is a histogrammed and discounted function of previous state positions and angles, i.e. $NewState = Binned\ Current\ Position + \gamma * PreviousState$, where γ is a discounting factor, currently set at 0.7. The numpy library in python has a function histogram2d which allows the binning of two-dimensional arrays.

pseudocode, proof and explanation + cost/benefits, improving the binning near the centre.

0.1.4 Episode Execution

0.2 Self Play and Adversaries

0.2.1 Point of Action

symmetry, action choices and PMW

A discrete time step of 0.02s is 35x smaller than this and therefore we expect an impulse to cause $\sim 3\%$ change in the state values. This is far below the threshold for pulse-width modulation, i.e. the actions are fast enough for the input forces to be modelled as continuous **** Is this right?? experimentally, the largest velocities give even better results ****

0.2.2 Worst Possible Action

0.2.3 Adversarial Cost

representation with the cost function.

0.3 Neural Network

0.3.1 Loss Functions and Pareto

0.3.2 Architectures

Player vs Adversary Architectures? Combined?

0.4 MCTS

outline + pseudocode

0.4.1 State and Player Representation

0.4.2 Terminal States and Suicide***

0.4.3 Modified UCB

0.5 Player and Adversary Evaluation

0.5.1 Elo Scoring