Image Control of the Inverted Pendulum

Nicholas Tan (ntan2012@stanford.edu) | Sean Fitzgerald (seantom@stanford.edu) | Erik Augustine (eraugust@stanford.edu)

Introduction:

In this project we aim to control and balance an inverted pendulum through image recognition. We tried two methods. The first, using a mixture of a neural net and reinforcement learning. The second method was a Deep Q Network. We had mixed results with both techniques.

Neural Net and Reinforcement Learning:

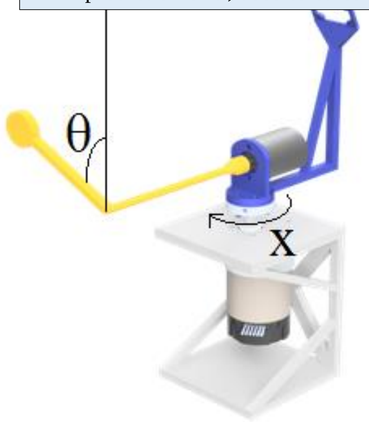
The two techniques we use to control our system are neural networks and reinforcement learning.

1. The image generated from our model is fed into a neural network that predicts the angle.
2. The estimated angle is fed into the FSM that gives the RL algorithm its state.

Data Acquisition and System Model

- System Model:
  - Our system is the "rotary" variant of the inverted pendulum. We created a physics model of this pendulum, and feed the system the motor torque.

    The model outputs state $s = \left(\theta, \frac{d\theta}{dt}, x, \frac{dx}{dt}\right)$
- We create images representative of the system state to train and test our NN model, which outputs a prediction of θ, used to calculate state.



Neural Net:

Input: $z^{[0]} \in \mathbb{R}^{3072}$

Calculate 3 hidden layers:

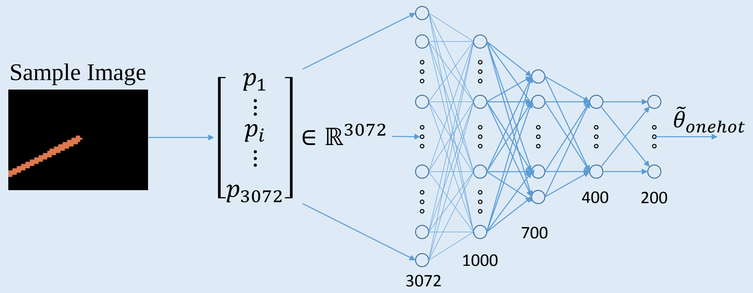
$z^{[l]} = W^{[l]} z^{[l-1]} + b^{[l]}$

$a^{[l]} = \sigma\left(z^{[l]}\right)$

Calculate output:

$y = softmax\left(z^{[3]}\right) = \tilde{\theta}_{onehot}$

Network is trained via mini-batch gradient descent with B = 10.



Reinforcement Learning:

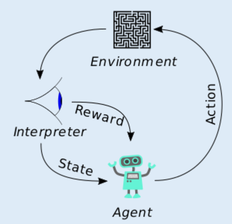
State $s = \left(\theta, \frac{d\theta}{dt}, \frac{dx}{dt}\right) \approx \left(\tilde{\theta}, \frac{d\tilde{\theta}}{dt}, \frac{d\tilde{x}}{dt}\right) \in \mathbb{R}^{241}$

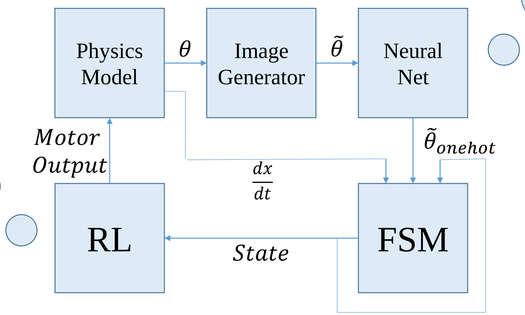
Actions $a = (-2, -1, 0, 1, 2) \in \mathbb{R}^5$

Reward $R: s, a \to \mathbb{R}$

Maximize Value: $V^{\pi}(s,a) = E\left[\sum_i \gamma^i R(s_i)\right]$

$= R(s) + \gamma \sum_{z \in S} P_{s\pi(s)}(z) V^{\pi}(z)$

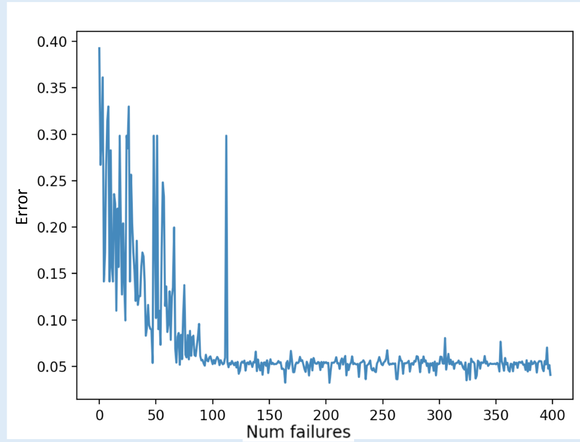


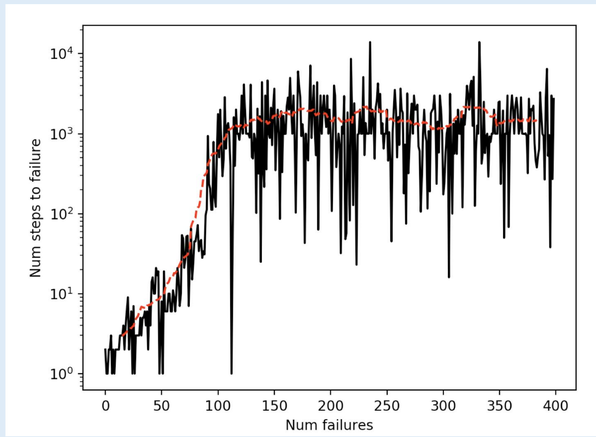


Results:

Control Method	Mean Time to Failure	Mean Error
Observe θ through NN	150 seconds	2.96º
Observe θ directly	232 seconds	2.71º
Discrete P Controller*	5 seconds	8.09º

*Discrete P controller had identical action space to RL model. A P controller with continuous action space was able to stay up indefinitely





DQN (Deep Q Network) is a reinforcement learning algorithm that is used when there is only very abstract information state information. The algorithm is an adaptation of Q-learning with the following features:

- Experience Replay: Every time a decision is made, the state transition, action, and observed reward are stored in "replay memory." A mini-batch is then randomly sampled from all of replay memory for gradient descent. This reduces the contextual bias of the neural network.
- Separate networks for learning and prediction: DQN maintains two separate neural networks, $Q(s,a)$ and $\hat{Q}(s,a)$. Q is used for prediction and is updated at every gradient descent. $\hat{Q}$ is used as the truth reference for the gradient descent steps on Q. Periodically, the parameters or Q are copied to $\hat{Q}$

We further modified this algorithm for our swing-up problem:

- Instead of neural network that predicted only reward from combined state and action input, we used used softmax to predict a reward for all actions from only the input state. This reduced the number of times we ran forward propagation.
- To fully explore the state space, we used an epsilon-greedy strategy, annealing epsilon over some fraction of the initial time steps down to some nominal probability.

Results: We applied DQN to solving the inverted pendulum swing-up problem on a physical model and did not achieve convergence. We should have used a less general method (such as the one shown above) and slowly generalized that algorithm into a DQN configuration that is able to achieve convergence.

Future Work:

- Further work on the neural net could help performance. Observing θ through an image caused a decrease in performance due to errors in estimation.
- The design could be made far more robust – correcting disturbances at the output would be essential in many applications
- The "swing up" problem could potentially be solved with a different training algorithm.
- Our DQN was unsuccessful in balancing the pendulum, although direct observation of its learning and performance showed that there was potential to make our system functional. Further research and debugging is needed.

Conclusions:

- State space and action space discretization can play a large role in system performance. In particular, the feature state to state space needs to be optimized.
- "Compute" is hard. We had to downsample from 640x480 pixels to 64x48 pixels. Furthermore we reduce the number of hidden layers greatly in order operate on reasonable time scales.
- Performance was lost due to the error in the neural net estimate. Further optimizing the image (reducing pendulum width, increasing size, etc) could help improve performance.

References:

- RL Diagram: https://en.wikipedia.org/wiki/Reinforcement_learning#/media/File:Reinforcement_learning_diagram.svg
- Pendulum Model: "On the dynamics of the Furuta Pendulum" https://www.hindawi.com/journals/jcse/2011/528341/