A Hybrid Approach for Reinforcement Learning Using Virtual Policy Gradient for Balancing an Inverted Pendulum

Doctoral Consortium

Dylan Bates

Department of Mathematics
Center for Research in Scientific Computation
North Carolina State University

Table of Contents

1. Research Problem
2. Outline of Objectives
3. State of the Art
4. Methodology
5. Expected Outcome
6. Stage of the Research

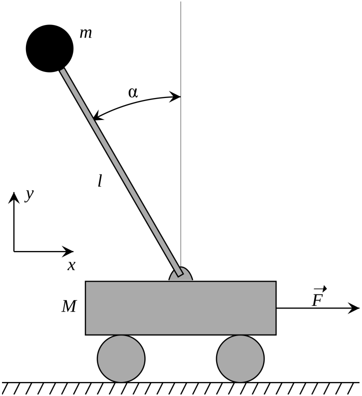
Introduction

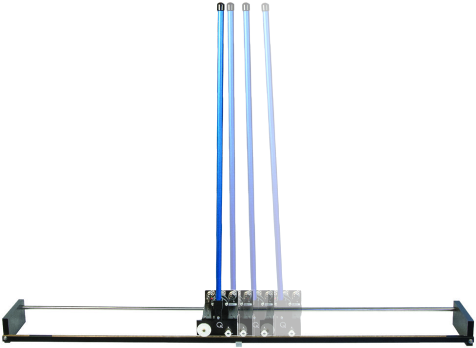
Reinforcement Learning is the process of using trial-and-error with finite rewards.

- Input state of the environment
- Output optimal action
- (Optional) Rewards

Our agent is an underactuated single-inverted pendulum on a one-dimensional track. The nonlinear equations of motion result in an unstable equilibrium that can be difficult to maintain.

Environment and Agent





Goals

- Use RL to train a realistic simulation of a virtual pole to balance itself.
- Use the trained neural network to balance a real pole in the real world.

Ultimately, training through simulation can speed up training time and increase robustness.

Literature Review

- Traditional control: Linearized systems limit the angle of deflection. Robost performance is only achieved with properly tuned weights.
- Virtual RL: Usually implemented in a simulation like OpenAI's Cartpole. Physics here are usually questionable.
- Real RL: “Nothing works; I mean, the robots will break down - they'll break down all the time.” - Dr. Tim Lillicrap

Environment

- Started with OpenAI Cartpole
- Adjusted physics, improved EOM
- Continuous action-space
- State: $[x, \alpha, \dot{x}, \dot{\alpha}]$

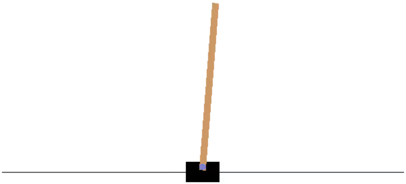
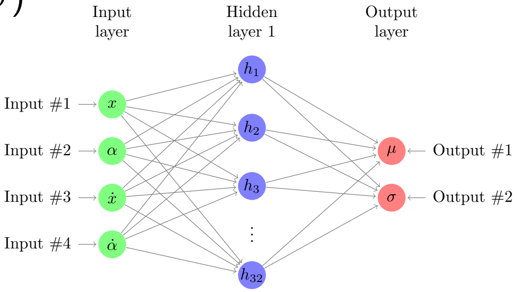


Figure: Modified Cartpole environment.

Neural Network

- Input state vector
- `ReLU` activation
- 2 outputs: $\sim N(\mu, \sigma)$

Figure: Artificial NN approximates actions.



Discounted Rewards

Rewards: $r_t = 1,\ 0 \leq t \leq T$

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

- $0 < \gamma < 1$ is a discount factor, intended to prioritize actions now over actions in the future.
- Larger values of γ takes a long term approach.

Loss

$$L = -R \cdot \tilde{a} + \epsilon H$$

- $R =$ normalized discounted rewards
- $\tilde{a} =$ log probability of actions
- $H =$ optional entropy parameter

This can be modified to promote optimal behaviour.

Neural Networks - Policy Gradient

Unlike supervised learning, we cannot calculate an explicit error between the neural network's output and the "correct" answer. Goal: max $\sum_i \ln p(y_i|x_i)$

- Maximize expected return:
  $J(\pi_\theta) = \int_\tau P(\tau|\theta)R(\tau) = \underset{\tau\sim\pi_\theta}{\mathbb{E}}[R(\tau)]$
- Use gradient descent:
  $\nabla_\theta J(\pi_\theta) = \underset{\tau\sim\pi_\theta}{\mathbb{E}}[\nabla_\theta \ln P(\tau|\theta)R(\tau)]$
- Normalized discounted rewards encourage and discourage half of actions.

Virtual Policy Gradient

$$\nabla_\theta J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{\mathbb{E}} \left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t | s_t) A^{\pi_\theta}(s_t, a_t) \right]$$

The algorithm is incredibly generalizeable, allowing the same code to complete a variety of tasks.

- Balance a pole
- Play Pong
- Drive a car

You only need an updated simulation, and the appropriate number of neurons, hyperparameters...

Results

The inverted pendulum learned to balance for at least 10 seconds 91% of the time

- takes an average of 807 trials
- high variance among gradients and training time
- optimizing hyperparameters reduced this to 355
- minimum trials was only 54 ($\sim$1 minute to train)

Due to the high variance among the gradients, even the best hyperparameters had inconsistent performance.

New Work - Actor Critic and PPO

- Actor: ANN predicting the best action to take
- Critic: new head of the same ANN estimating the value of that action, $V^{\pi}(s_t)$
- Subtracting this baseline reduces the variance of Policy Gradient

Given $Q^{\pi}(s,a) = \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a]$,
set $A^{\pi}(a_t|s_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$

Then we use the same equation as before:

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)\right]$$

Applications

Ultimately, training in simulation could make other machine learning applications more efficient, speeding up the development of control systems that can be implemented in the real world.

- Self-driving cars
- Reusable rockets
- Complex robotics
- Healthcare treatment

Domain randomization and cross-modal learning can help make virtually trained models more robust.

Questions and Contact Info

- Email: `dwbates@ncsu.edu`
- LinkedIn: `linkedin//dwgb93`
- Videos/Slides/Paper: `https://go.ncsu.edu/icaart-videos`
- Happy Birthday, Mom!