Inverse Pendulum Simulation using Reinforcement Learning

Yuanjun “Dastin” Huang and Po Hsiang Huang

01

Introduction

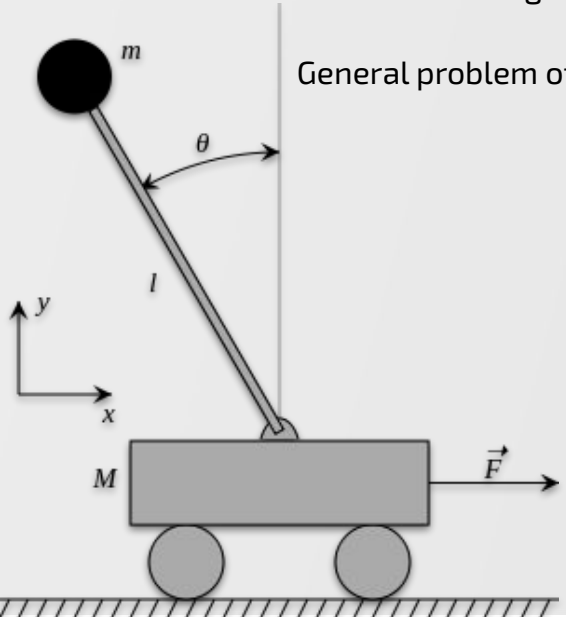
Background

Inverted Pendulum problem involves balancing a pendulum on a cart.

Widely adopted baseline problem for many control algorithms

Reinforcement learning is a subfield of machine learning

General problem of decision making



Background

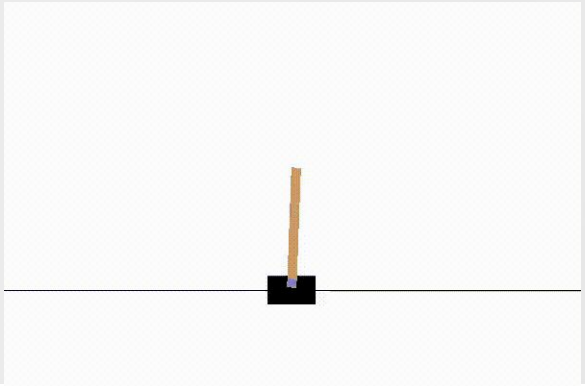
OpenAI developed Gym because:

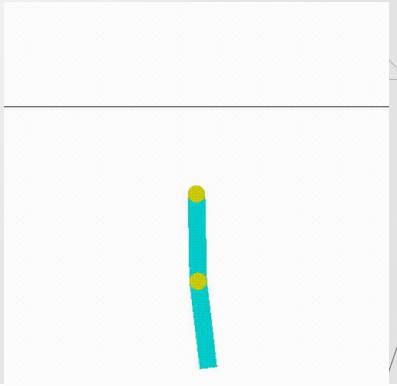
“The need for better benchmark”

“The lack of standardization of environments used in publications”

Classic pendulum control problems to tackle:

Cartpole-v1, Pendulum-v0, Acrobot-v1





Deep Learning Motivation

Traditional methods are not effective in more complicated problems

E.g. Acrobat-v1 with two inverted pendulums connected

Traditional methods are limited to be discrete inputs

Reinforcement learning using neural networks may be more effective at learning a function approximator for the Bellman equation

02

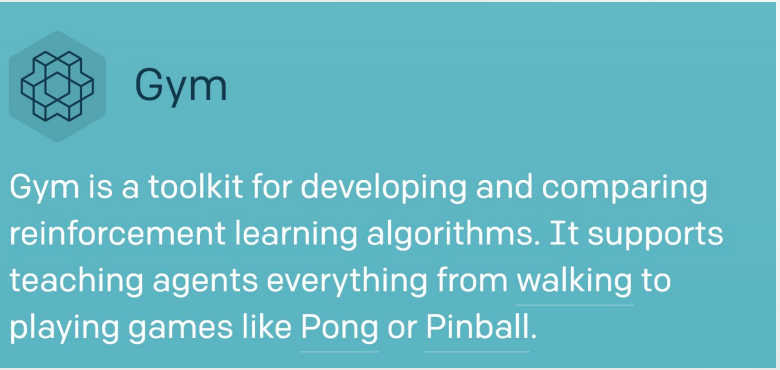
Related Work

Literature Survey

OpenAI Gym

OpenAI Gym Leaderboard

Balancing a CartPole System with Reinforcement Learning - A Tutorial



CartPole-v0

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

- Environment Details
- *CartPole-v0 defines "solving" as getting average reward of 195.0 over 100 consecutive trials.*
- *This environment corresponds to the version of the cart-pole problem described by Barto, Sutton, and Anderson [Barto83].*

User	Episodes before solve	Write-up	Video
Zhiqing Xiao	0 (use close-form preset policy)	writeup	
Hengjian Jia	0 (use close-form PID policy)	code/writeup	
Keavnn	0	writeup	
Shakti Kumar	0	writeup	Video
Nextgrid.ai 🔥	0	writeup	Video
iRyanBell	2	writeup	
Adam	3 (36)	writeup	
Daniel Sallander	4	writeup	
Kapil Chauhan	4	writeup	
Ritika Kapoor	7 (use genetic algorithm)	writeup	
Ben Harris	12	writeup	video
Tiger37	14 (0)	writeup	video
Blake Richey	20	writeup	
LukaszFuszara	22	writeup	video
MisterTea, econti	24	writeup	
Roald Brønstad	24	writeup	
yingzwang	32	writeup	
sharvar	33	writeup	

Literature Survey

Deep Reinforcement Learning with Double Q-learning

Comparison of reinforcement learning algorithms applied to the cart-pole problem

Fixed point controllers and stabilization of the cart-pole system and the rotating pendulum

Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems

03

Methods

Dataset

No external dataset was needed

Collect observations from the environment (continuous):

Cart Position, Cart Velocity, Pole Angle, and Pole Angular Velocity

Perform actions to the environment (discrete):

Push cart to the left and Push cart to the right

Feature Extraction

For Q Learning and Double Q Learning:

- Needed to discretize continuous observations into tables
- Assign bins for each observation in the order of its importance to the problem
  - Tested out different bin counts, (1, 1, 6, 12) works the best

For Deep Q Network:

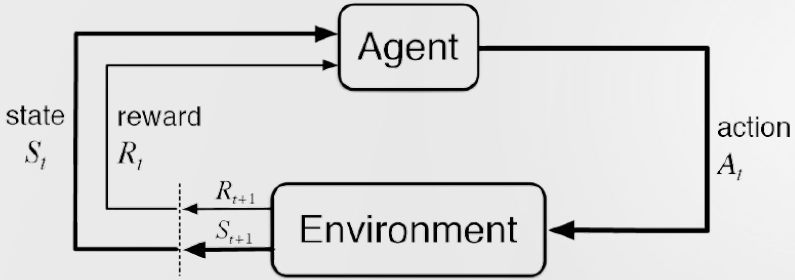
- Tried 2D Convolutional Neural Network on frames extracted from the environment
- No need for discretizing observations on kinematic state

Q Learning

Explore vs Exploit

Goal: maximize Q value, or future expected reward

Stored into a Q table matching observations to actions



$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}^{\text{temporal difference}}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

Double Q Learning

Double Q = Two Q Tables

Action is chosen based on the average value of the two tables

Each table is updated alternately at each iteration

Counters bias introduced in a single Q Table due to overestimation of future return value

Deep Q Learning

Deep neural network = function approximator

Q learning = learning the Q value of each state-action pair, i.e., Q(s, a) = f(s, a)

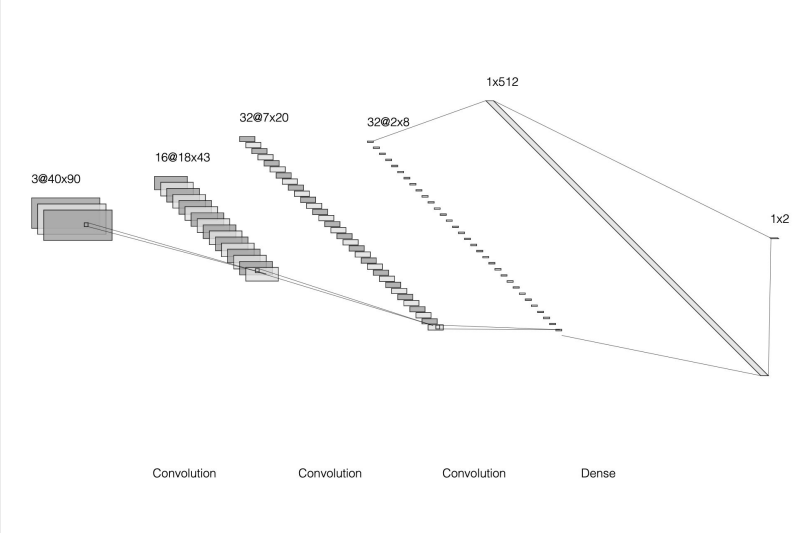
Deep Q learning = using DNN to approximate the function f

2 different pipelines

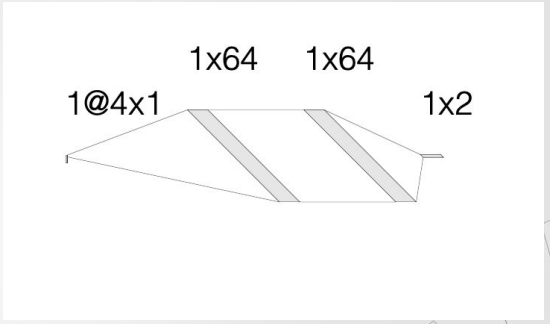
- Rendered image -> Q values -> control strategy
- System kinematics state -> Q values -> control strategy

DL Model Architecture

Pipeline 1: rendered image as input



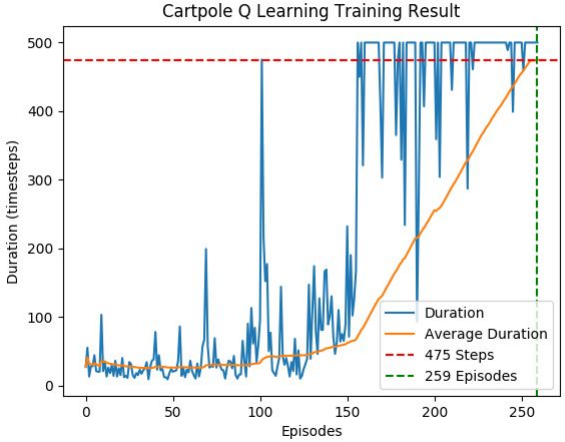
Pipeline 2: take $(x, \dot{x}, \theta, \omega)$ as input

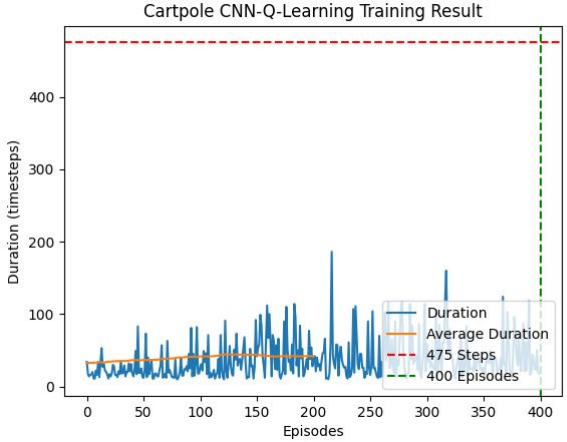


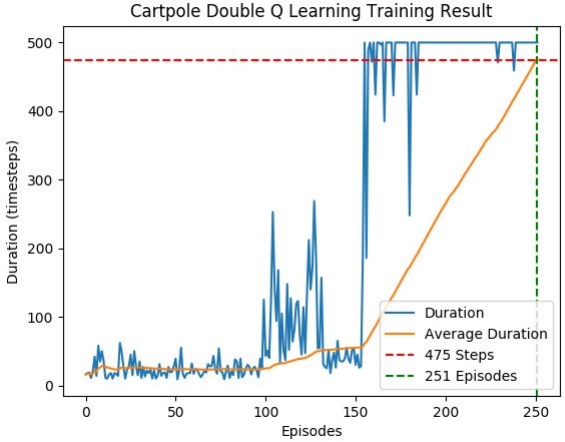
04

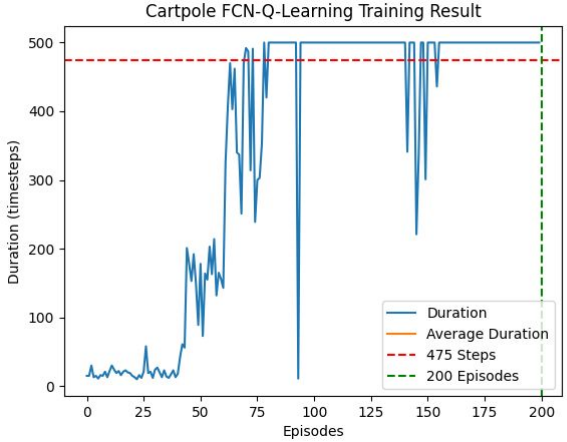
Experiments and Results

Results


Cartpole Q Learning Training Result
Duration (timesteps)
Episodes
Duration
Average Duration
475 Steps
259 Episodes


Cartpole CNN-Q-Learning Training Result
Duration (timesteps)
Episodes
Duration
Average Duration
475 Steps
400 Episodes


Cartpole Double Q Learning Training Result
Duration (timesteps)
Episodes
Duration
Average Duration
475 Steps
251 Episodes


Cartpole FCN-Q-Learning Training Result
Duration (timesteps)
Episodes
Duration
Average Duration
475 Steps
200 Episodes

Results Summary

Cartpole-v0
Maximum timestep = 200
Reach an average 195 steps over 100 episodes

	Q Learning	Double Q Learning	DQN (CNN)	DQN (FNN)
Episodes	300.6	279.6	200 [fixed]	200 [fixed]
Steps w/ perturb	183.9	180.2	9.3	200

Cartpole-v1
Maximum timestep = 500
Reach an average 475 steps over 100 episodes

	Q Learning	Double Q Learning	DQN (CNN)	DQN (FNN)
Episodes	323.0	314.2	200 [fixed]	200 [fixed]
Steps w/ perturb	451.0	454.3	9.4	500

*results are averaged over 10 runs

Old Results Summary

Cartpole-v0
Maximum timestep = 200
Reach an average 195 steps over 100 episodes

	Q Learning	Double Q Learning	DQN (CNN)	DQN (FNN)
Episodes	301.6	432.6		
Steps w/ perturb	181.3	182.1		

Cartpole-v1
Maximum timestep = 500
Reach an average 475 steps over 100 episodes

	Q Learning	Double Q Learning	DQN (CNN)	DQN (FNN)
Episodes	339.5	296.6		
Steps w/ perturb	451.7	433.0		

*results are averaged over 10 runs

05

Code Demo

06

Future Work

TODO

- Documentation
  - Code
  - README
- Test out different parameters
- Try out different network architectures
- Implement other variations of Q learning: Double DQN, Dueling DQN, Prioritized Experience Replay……
- Final report write up

07

References

References

[1] Nagendra, Savinay, et al. "Comparison of reinforcement learning algorithms applied to the cart-pole problem." *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2017.

[2] Olfati-Saber, Reza. "Fixed point controllers and stabilization of the cart-pole system and the rotating pendulum." *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*. Vol. 2. IEEE, 1999.

[3] Brockman, Greg, et al. "Openai gym." *arXiv preprint arXiv:1606.01540* (2016).

[4] https://gsurma.medium.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288

[5] https://blog.floydhub.com/an-introduction-to-q-learning-reinforcement-learning/

[6] https://gym.openai.com/docs/

[7] https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

[8] https://github.com/tensorflow/agents/blob/master/docs/tutorials/1_dqn_tutorial.ipynb

[9] https://engineering.purdue.edu/DeepLearn/pdf-kak/week16.pdf

The End

Thank you!