Deep Reinforcement Learning for Spacecraft Trajectory Optimization

PHYS 416, Final Project Proposal

Ben Harris

Description

In the past decade, reinforcement learning has exploded in popularity. Following the release of Deepmind's AlphaGo project, this technique has become the go-to tool for training AI on . Unlike traditional neural networks which require massive datasets and train by minimizing loss functions , deep reinforcement learning (DLR) works through interaction with a model environment and trains the to maximize a rewards function. This makes the technique ideal for environment exploration and optimization with limited data.

Spacecraft Trajectory Optimization is an extremely difficult task in which scientists attempt to identify a path in space from their source to a target which minimizes a certain factor (fuel, time, etc). While there are known optimum trajectories in certain situations (primarily the Hohmann Transfer Orbit), these problems are incredibly difficult and become increasingly complex with the addition of more gravitational bodies.

For my final project, I would like to apply deep reinforcement learning as a method for solving trajectory optimization problems. Given the game-like nature of trajectory choices and semi-supervised requirements of DRL, I think these two methods are uniquely aligned to work well together.

Starting Plans

Reinforcement learning is centered around the interactions between two key components: the agent (neural network) and the environment (solar model). Rather than aiming to minimize a loss function (as most machine learning does), the agent learns by exploring the environment and aiming to maximize a reward function. A well-designed agent will split time between testing new ideas and refining known successful paths.

For the model, I'm going to start with the solar system I created in Chapter 3 Exercise B. I plan to extend the original capabilities to include a rocket object and plan to model the planetary and rocket motion using the Adaptive Runge Kutta. Next, I'll build a simple 2D model of the earth and mars and calculate the Hohmann Transfer Orbit between the planets. Once I have that baseline, I'll be ready to connect an agent and begin training. Hopefully, the agent will be able to find a trajectory connecting earth and mars.

For the agent, I plan to start with tutorial level neural networks and potentially explore more complicated architectures as I progress. My reward function will begin as the minimum distance a rocket gets from the target but will later incorporate a minimization of fuel factor.

I plan to code the project in Python. For the neural network aspects, I plan to use TF Agents: a TensorFlow extension which provides specialized functionality for deep reinforcement learning.

Goals

This project has the potential to be incredibly complex. However, I've identified five components that one can adjust to simplify the problem. They are as follows:

- Mathematical Space -- Is the model built in 2D or 3D? I plan to start in 2D. If all goes well, I will explore building the model in 3D environments.
- Solar System Difficulty -- Does the solar system include lots of moving planets that screw up the rocket? I plan to start with two planets revolving in sync. If that works, I'll gradually increase the complexity adding more planets, retrograde and out-of-sync orbits, and making the targets harder.
- Solar System Consistency -- Does the environment vary or is the model trained on a consistent solar system? When we keep the solar system consistent, the model learns how to solve that specific optimization, however, it will be completely useless if the target changes. If the environment varies in training, theoretically, the model will be able to adapt to variable targets and novel solar systems. I plan to start with consistent systems as proof of concept and then move to variable environments.
- Launchpoint -- Who controls the launch timing? Is the rocket control the only thing that the agent must decide or does it also have to find a valid launch window? I plan to start with fixed launch points and later give the rocket control of launch timing.
- Rocket thrust -- Does the rocket have instantaneous change in velocity or is there limitation on acceleration? I will almost exclusively work with a high-thrust model (no acceleration limitations). Low thrust modeling is incredibly difficult and seems to be beyond the scope of this class.
- Rewards -- How do we score each attempt? Is it a single factor or an aggregate of many? I plan to start with scoring by the minimum distance the attempt was from the target. I then plan to use minimum distance as the primary reward with a secondary

By slowly increasing the level of difficulty for these factors, one can morph a rudimentary system into the Global Trajectory Optimization Problem. A possible pipeline to do so is as follows:

1. Consistent simple solar system in 2D space with 3DOF HT rocket, stationary launch, and single reward
2. Consistent simple solar system in 2D space with 3DOF HT rocket, stationary launch, and aggregate reward
3. Consistent simple solar system in 2D space with 3DOF HT rocket, variable launch, and aggregate reward
4. Varied simple solar system in 2D space with 3DOF HT rocket, variable launch, and aggregate reward
5. Varied complex solar system in 2D space with 3DOF HT rocket, variable launch, and aggregate reward
6. Consistent simple solar system in 3D space with 6DOF HT rocket, variable launch, and aggregate reward
7. Varied simple solar system in 3D space with 6DOF HT rocket, variable launch, and aggregate reward
8. Varied complex solar system in 3D space with 6DOF HT rocket, variable launch, and aggregate reward
9. Varied complex solar system in 3D space with 6DOF LT rocket, variable launch, and aggregate reward

10. Global Trajectory Optimization Problem

For my project, I am confident in my ability to complete Steps 1-3. I could see myself getting as far as Step 8, but that is the absolute dream (everything works scenario). Realistically, I think Step 5 would be an incredible achievement and should be the goal. Two PhD thesis have been written attempting Steps 9 and 10 (see Resources 3 and 4), so those are just there to show the potential of the pipeline.

Resources

1. Machine Learning and Evolutionary Techniques in Interplanetary Trajectory Design -- Dario Izzo, Christopher Sprague, and Dharmesh Tailor

2. A Survey on Artificial Intelligence Trends in Spacecraft Guidance Dynamics and Control -- Dario Izzo, Marcus M¨artens, and Binfeng Pan

3. A Reinforcement Learning Approach to Spacecraft Trajectory Optimization -- Daniel Kolosa

4. Low-thrust Spacecraft Guidance and Control using Proximal Policy Optimization -- Daniel Miller

5. Guidance for Closed-Loop Transfers using Reinforcement Learning with Application to Libration Point Orbits -- Nicholas LaFarge, Daniel Miller, Kathleen Howell, Richard Linares

6. Reinforcement Learning for Low-Thrust Trajectory Design of Interplanetary Missions -- Alessandro Zavoli, Lorenzo Federici