Train and compare RL methods for optimal control of a vehicle on a constant curvature

Zachary Chan and Rohith Nibhanupudi

A standard task in control theory is to design a controller to track a reference signal by commanding actuators to perform actions. In a car, this can be thought of as a motor controlling the steering wheel of a car to direct the car's motion to the center of a road lane – known as lateral control or lane centering. Due to the state-action nature of control problems and reinforcement learning's (RL) optimization of rewards over state-action spaces, using reinforcement learning to perform lane centering is feasible. Our project will evaluate the use of reinforcement learning for lateral control of a vehicle to follow a curve with constant radius. We will evaluate the use of 3 reinforcement learning algorithms: deep deterministic policy gradient (DDPG), soft actor critic (SAC), and adversarial inverse reinforcement learning (AIRL) with a model predictive controller (MPC) as an expert. DDPG and SAC are both off policy actor critic methods. The difference is that DDPG uses a deterministic policy actor network while SAC uses a stochastic policy actor network. They are both algorithms that can learn a continuous state and action space which makes them capable of continuous time control of a system. AIRL aims to learn a policy actor that imitates an existing policy, with adversarial learning techniques introduced to reduce bias and promote exploration. The algorithms will be evaluated using the maximum reward obtained in a testing environment. The models will be trained and evaluated in an online learning setting where models only have access to and predict from historical data at any given round. This setting also mirrors real-world application, since a vehicle does not know every optimal action for an entire trip at every point.

We will develop the project in Matlab. We will first set up an environment with continuous action and state spaces to simulate lateral dynamics. This includes using first principles equations for vehicle motion and determining vehicle parameters which will be fixed throughout the project. Next, we will create reinforcement learning models in Matlab. Matlab has a Reinforcement Learning toolbox to create and train DDPG and SAC agents. We will also leverage Matlab's RL toolbox to build the AIRL model, with a backup model developed in PyTorch and imported into Matlab if development is too difficult in Matlab. After developing model architecture, we will integrate our environment into the training loop and monitor training and validation metrics to prevent overfitting. Next, we will evaluate the models in a noisy testing environment like the training environment and observe the total reward of each model to determine the best performing approach. After each stage of development is completed, we will build the corresponding sections of the paper and poster to ensure that there are no delays.

The timeline is summarized as follows: From November 6th to November 12th, both members will develop the environment, with future weeks used to fine-tune the environment for any missing parameters. Then, from November 13th to November 30th, we will develop and train the models. Rohith will focus on the Imitation Learning model, and Zach will focus on the actor critic methods. Next, from December 1st to December 5th, we will build the environment and run inference to evaluate the models. All remaining edits to the poster and presentation will be completed in the week of December 5th to December 13th ahead of all final deadlines for project submission.

References:

1. M. Brasch, I. S. Heinz and A. Bayer, "Lateral Control of a Vehicle using Reinforcement Learning," 2022 8th International Conference on Control, Decision and Information Technologies (CoDIT), Istanbul, Turkey, 2022, pp. 451-456, doi: 10.1109/CoDIT55151.2022.9804101.
2. https://www.mathworks.com/help/reinforcement-learning/ug/imitate-nonlinear-mpc-controller-for-flying-robot.html
3. https://www.mathworks.com/help/reinforcement-learning/ug/train-ddpg-agent-for-path-following-control.html