Reinforcement Learning for Autonomous Racing in Simulation
CS229 Research Proposal

Adithya Ganesh, Matthew Das Sarma, Nancy Xu

August 18, 2019

Objective. Apply reinforcement learning to train an agent that can autonomously race in TORCS (The Open Racing Car Simulator). TORCS is a modern simulation platform used for research in control systems and autonomous driving. We will train learning algorithms to autonomously race based on variations of classical reinforcement learning techniques. The model will be used to race in TORCS, with the ultimate goal to achieve minimal race time and maximal robustness across tracks.

Initial Goal.

1. Use variations of classical RL methods to train a functional racing model
   (a) Metrics: completes circuit, constrained within track, baseline racing time, etc.
   (b) Compare deterministic and stochastic policies
   (c) Compare different reward functions (e.g. finite horizon vs. infinite horizon)

Stretch Goals / Ongoing Research.

1. *Deep learning.* Compare classical methods to modern deep learning approaches.
   (a) Fitted value iteration to solve an MDP
   (b) Deep Q-learning
2. *Transfer learning.* How effect can a model trained in simulation be in the real world?
   (a) Experiment with real-world autonomous driving – can we use a model trained in TORCS to control an RC car outfitted with cameras and ultrasonic sensors?

1 Reinforcement Learning Framework

1.1 Classical MDP Approach

In the traditional reinforcement learning model, an agent functions in its environment through perception and action. At each iteration, the agent receives input, i that describes the current state s, and then selects an action a. The action modifies the state of the environment, and the state transition is communicated to the agent through a reward R. The goal of reinforcement learning is to find a policy π that chooses actions a in in given states s that maximize the cumulative expectation of R.

Classical approaches are based on the assumption that the dynamics of our system can be modeled as a Markov decision process, which consists of the set of states S, actions A, rewards R, and transition probabilities T. In particular, the Markov assumption requires that the next state s' and the reward only depends on the previous state s and action a.

Importantly, π can either be deterministic or stochastic. There are also different models of the expected return J (mention the difference between finite-horizon and infinite horizon).

In the average-reward setting, we can frame the control problem as a constrained optimization problem (due to Bellman):

$$\max_{\pi} J(\pi) = \sum_{s,a} \mu^{\pi}(s)\pi(s,a)R(s,a),$$

such that

$$\mu^{\pi}(s') = \sum_{s,a} \mu^{\pi}(s)\pi(s,a)T(s,a,s'); \quad \forall s' \in S,$$

$$1 = \sum_{s,a} \mu^{\pi}(s)\pi(s,a),$$

$$\pi(s,a) \geq 0, \forall s \in S, a \in A.$$

1.2 Deep Q-Learning

Recently, much of the research literature has focused on deep Q learning, which tries to use a deep convolutional neural network to approximate the optimal action value function

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi],$$

which is the maximum sum of rewards r_t, discounted by γ at each time step t, achievable by a behavior policy $\pi = P(a|s)$.

2 Overview of Control Problem

The agent perceives the environment via a number of simulated sensor measurements.

Example sensor measurements (a total of 79 sensors exist):

Name	**Range (unit)**	**Description**
`angle`	$[-\pi, +\pi]$ (rad)	Angle between vehicle heading and heading of track axis.
`distFromStart`	$[0, +\infty)$ (m)	Distance of the car from the starting line along the track path.
`track`	[0, 200] (m)	Vector of 19 range finder sensors that return the distance between the track edge and the vehicle within a range of 200 meters.
`speed` 4	5	6
7	8	9

- trackSensors - Vector of 19 range finder sensors that return the distance between the track edge and the vehicle within a range of 200 meters.
- Angle - Angle between vehicle and track heading.
- distFromStart - Distance of the car from the starting line along the track path.
- focus - Vector of 5 range finger sensors that return the distance between the track edge and the car within a range of 200 meters.
- opponents - Vector of 36 opponent sensors: each sensor covers a span of 10 degrees within a range of 200 meters and returns the distance of the cloest opponent in the covered area.
- speed{X, Y, Z} - Speed of the vehicle along longitudinal, transverse, and Z axis of car.
- wheelSpeedVel - Vector of 4 sensors representing the rotation speed of wheels.

A vehicle controller must determine the value of five actuators - the steering wheel, the gas pedal, the brake pedal, and the gearbox.

- Accel - Virtual gas pedal
- Brake - Virtual brake pedal
- Clutch - Virtual cluth pedal
- Gear value

- Steering

Importantly, the control algorithm must rely on local track and dynamics information only (without have a representation of the full track state).

3 References