CS221 project proposal: AlphaZero for Autonomous Driving

xxx

September 27, 2019

1 Problem statement and candidate techniques

1.1 Problem to solve

We want to solve a Decision Making problem.

- Use Case: Motion Planning, controlling acceleration on a predefined path, with many other objects crossing our path
- No loss of generality as quite frequently motion planning is done in 2 steps:
  - Compute a path to follow: a path that is kinematically feasible and collision free w.r.t. static objects (checked against an occupancy grid)
  - Define a velocity profile on this path: that is collision free w.r.t. dynamic objects
- We have to plan a trajectory that is:
  - Collision free
  - Comfortable: we want to avoid too strong accelerations and decelerations
  - Efficient: go as fast as possible from point A to point B
- Sources of uncertainties:
  - Sensor uncertainty: state vector of other vehicles may not be known exactly
  - Driving Model uncertainty: CA, CV, IDM
  - Even without uncertainty the problem is already difficult to handle when there are many objects around

1.2 Input-Output Model

State Space is continuous, Action Space is Discrete

- Inputs: state description $[x, y, \dot{x}, \dot{y}]$ or $[x, y, \dot{x}, y, \dot{\ddot{x}}, \ddot{y}]$ for all objects
- Ouput: longitudinal acceleration every 20 ms, discret set of actions considered $\{-4, -2, -1, 0, 1, 2\} m.s^{-2}$

1.3 Candidate Techniques to solve the problem

Let's first consider the problem without sources of uncertainty. Known state vectors, known driving models

- Hard coded rule: to be used as a Baseline
- Planning (or Model Based RL): MCTS tree search with progressive widening to handle the continuous state space
- Learning (or Model Free RL): DQN or Policy gradient algorithm that is trained in simulation
- Combining planning and learning: a MCTS tree search that is guided for more efficient search by a trained NN.
- Others: CSP ???

The expected benefit of combining planning and learning is to make the search much more efficient, so that in less iterations we find a good solution suitable for real-time online planning.

Sources of uncertainty are handled with POMDP models (more in the scope of AA228/CS238 than CS221)

2 CS221 Project Setup

All coding done in Python. Deep learning part: pytorch or TF/Keras ??

- simulator: uses standard open ai gym interfaces. We can use an existing custom simulator gym-act. Or check if there are other existing simulators (if possible compatible with open ai gym interfaces) dealing with a similar task
- Baseline: a simple hard coded rule producing acceleration command every 20 ms, a simple velocity profile generation
- Oracle: MCTS tree search with many/many iterations (not suitable for realtime online use).
- Algo1 MCTS: planning based, MCTS tree serach with progressive widening cf [3]

- Algo2 NN: learning based, CNN network for RL (DQN or Policy Gradient ??) cf [1]
- Algo3 MCTS-NN: combine algo1 and algo2 in a similar way than AlphaZero and [2]

3 CS221 Project schedule

- ... of October:
  - Reproduce AlphaZero simple setup
  - Outline differences between Chess game and Autonomous Driving use case
- 24^{th} of October:
  - Simulator running
  - Baseline: results available
  - Oracle: Branch and Bound search results (cf [3] chapter 4.6.2)
- 14^{th} of November:
  - 1st version of Algo1 MCTS
  - 1st version of Algo2 NN
- $2^{nd}$of December:
  - 1st version of algo3 MCTS-NN
- 13^{th} of December:
  - Project report due

References

[1] Carl-Johan Hoel, Krister Wolff, and Leo Laine. "Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning". In: *CoRR* abs/1803.10056 (2018). arXiv: `1803.10056`. URL: `http://arxiv.org/abs/1803.10056`.

[2] Carl-Johan Hoel et al. "Combining Planning and Deep Reinforcement Learning in Tactical Decision Making for Autonomous Driving". In: *CoRR* abs/1905.02680 (2019). arXiv: `1905.02680`. URL: `http://arxiv.org/abs/1905.02680`.

[3] Mykel J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.

[4] David Silver et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". In: *CoRR* abs/1712.01815 (2017). arXiv: `1712.01815`. URL: `http://arxiv.org/abs/1712.01815`.