Reinforcement Learning with Hard Constraints for Autonomous Driving

Vaishali Kulkarni & Philippe Weingertner
Ramtin Keramati (Project mentor)

STANFORD UNIVERSITY

Abstract

We consider a problem of RL agent providing safety guarantees along with efficient strategy for autonomous driving. We define hard constraints applicable to real life that the RL agent has to enforce, not just in expectation but all the time. We explore policy gradient algorithms and propose modifications to deal with complex, non-differentiable hard constraints.

Introduction

For an autonomous vehicle, the real state of the world is partially observable. Also the models of the surrounding cars are not known so that creates a non-stationary uncertain environment. Most of the prior work have dealt with this problem either by including the constraints in the objective function, thereby minimizing this in expectation or constraining the exploration to only safer actions or overriding actions of policy with safer actions. We propose to modify the RL optimization to include hard constraints. One of the most important safety criterion for an autonomous vehicle is avoiding collisions which means we need to compute an early collision indication. We define a new safety constraints which calculates the time to collision (TTC) with all objects in the scene. We set a constraint on TTC to be above certain margin.

We define the RL objective taking into account hard constraints

$$\max_{\theta} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi_\theta(s_t))\right]$$

$$\text{s.t. } (minTTC(s_t, a_t)) \geq \text{Margin}_i \; \forall i \in [\![1, K]\!]$$

Note that the computation of TTC is a complex task and is non-differentiable function. We explore and modify the policy gradient algorithms to satisfy this constraints during the gradient search, allowing it work on both maximizing rewards while making sure that safety constraints are satisfied.

Anti Collison Test (Act) environment

We developed an openAI Gym compatible test setup for experimentation, namely ACT. We have support for 3 driver models: CV, Basic, IDM. The env calculates the minTTC constraints based on sampled parameter for each car at each time step. The ego car has to drive from point A to B. Other cars are instantiated randomly at different positions, with different speeds. The RL agent has to get from point A to B safely (avoiding collisions) and efficiently.

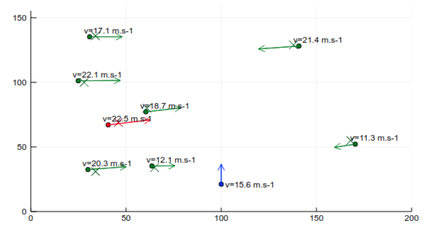

Figure 1: Anti Collision Test Environment

The problem is setup as a MDP with:

- States: different variants will be tested, but as a starting point consider $\{(x, y, v_x, v_y)_{ego}, (x, y, v_x, v_y)_{obj_{1..n}}\}$
- Actions: longitudinal accelerations $a_{longi} \in [-2ms^{-2}; +2ms^{-2}]$
- Rewards: $-1$for every timestep,-1000 for a collision terminal state, $+1000$ when goal is reached terminal state
- Discount factor: 0.99
- Model-Free setting: the ego-vehicle does not know the driver model of other cars, it could be CV, Basic or IDM.
- We found that using relative coordinates gives better results than using absolute coordinates.

Approach

The primary challenges posed are

- Due to inherent uncertainties in the IDM models, sampling of parameters for other cars is needed
- Very large state space $R^n_{obj}x4$
- Continuous action space
- constraints such as bounded acceleration, no backward motion allowed
- An iterative, non-differential hard constraint that we want to impose
- Non-stationary environment, need to move from unsafe states to safe states quickly.

The approach taken to study/address the problem is-

1. Define and code a safety cost associated with a batch
2. Backtracking linesearch at every policy update s.t minimize the objective loss function and linsearch to improve the safety cost
3. Benchmark with PG algorithms from openAI
4. Experiment with reduced state space to study the impact

Algorithm

TRPO updates policies by taking the largest step possible to improve performance, while satisfying KL-Divergence (distance between probability distributions). Conceptually, for the problem at hand, the goal is to allow largest step to improve performance as long as the new policy is better in terms of minTTC (minimize the number of collisions).

The modified-TRPO-like algorithm uses a "safety cost function" aka penalty measure to compare the old and new policy. The algorithm is described below.

0.1 Modified-PG-Penalty algorithm

Algorithm 1 Policy Gradient Algorithm with Penalty for minTTC

procedure POLICY GRADIENT(α)
  Initialize policy parameters θ and baseline values $b(s)$ for all s, e.g. to 0
  for iteration = $1, 2, \ldots$ **do**
    Collect a set of m trajectories by executing the current policy π_θ
    for each time step t of each trajectory $\tau^{(i)}$ **do**
      Compute the *return* $G_t^{(i)} = \sum_{t'=t}^{T-1} r_{t'}$
      Compute the *advantage estimate* $\hat{A}_t^{(i)} = G_t^{(i)} - b(s_t)$
    Re-fit the baseline to the empirical returns by updating to minimize

$$\sum_{i=1}^{m} \sum_{t=0}^{T-1} \|b(s_t) - G_t^{(i)}\|^2$$

  Update policy parameters θ_k using the policy gradient estimate $\hat{g}$

$$\hat{g} = \sum_{i=1}^{m} \sum_{t=0}^{T-1} \hat{A}_t^{(i)} \nabla_{\theta k} \log \pi_{\theta k}(a_t^{(i)} | s_t^{(i)})$$

  Calculate the "penalty" from the env by sampling actions from π_{theta_k} over a batch and Update the policy parameter θ_{k+1} using backtracking line search

$$\theta_{k+1} = \theta_k + \alpha^j \hat{g}$$

  where $j \in 1, 2, ...Iteration$ is the smallest value that minimizes the policy loss and the "penalty from collision"
  return θ and baseline values $b(s)$

Experiment Results

The results of the policy gradient using backtracking linesearch with safety cost constraint is attached below.

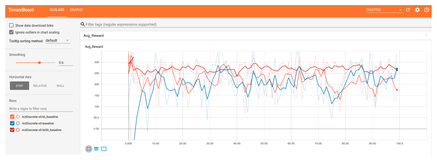

Figure 2: backtracking linesearch rewards

This is run over a batch size of 50000. The linesearch with 50 iterations gives reasonable results. The results are stable, low variance but nonetheless, the algorithm do not reach the optimal rewards of 1000 which represent best safety/efficiency metric.

The setup with reduced state (2 cars) is used to run PG algorithms from openAI. The results for the reduced state are shown below. The Soft Actor Critic (SAC) algorithm is able to show good results. Increasing the state space, all the below algorithms fail.

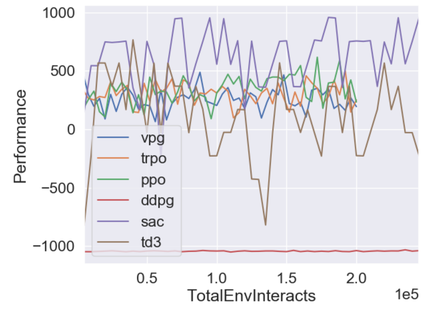

Figure 3: Figure caption

Conclusions

The Act environment with multiple objects along with non-differentiable constraints created challenging tasks that most state of the art PG algorithm could not handle. The proposed algorithm performed better with 10 objects and provided reduced collision rate. Dealing with scalability and IDM driver models remains a major challenge.

Future Work

For addressing the scalability wrt n_{obj}, images look promising and more work is needed to explore this approach (DQN tricks). Different variations of the constraints need to be considered that are computationally less complex and possibly differentiable.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. 2017.

[2] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. 2018.

Acknowledgements

We would like to thank Ramtin Keramati for being our project mentor and his valuable suggestions on our project.