Safe, Multi-Agent, RL for Autonomous Driving

Shai Shalev-Shwartz, Shaked Shammah, Amnon Shashua

Presented by: Ashish Gaurav

29th June, 2018

Outline

What is the paper about?

- shows how to do policy gradient without the Markovian assumption
- shows why traditional objectives pose a functional safety problem for autonomous driving
- proposes a new architecture for safe reinforcement learning (options graph)

Other work

- Learning in a multi-agent setting has been tried through game theoretic approaches, like minimax Q-learning [3], Nash Q-learning [2]
- Safe RL is also studied in [1], where they project the hard constraints on the NN weights

Problems with ML approaches

1. The algorithm performs better if it has seen more examples. So to ensure safety we need to train a lot.
2. The behavior of other agents is unpredictable. Our next state in the environment depends not just on our action, but on the behavior of other agents too.

Common solutions: POMDP, game-theoretic approaches

Categories of RL Algorithms

1. Algorithms that estimate Q or V functions
2. Policy gradient algorithms
3. Model based algorithms
4. Behavior cloning, i.e. supervised learning

Policy Gradient without Markovian Assumption

- states S, actions A, policies $\pi_\theta : S \times A \to [0, 1]$
- differentiable policies
- trajectories defined as sequence of state-action pairs

$$\bar{s} = \{(s_1, a_1), (s_2, a_2) \cdots (s_T, a_T)\}$$

- sub-trajectories

$$\bar{s}_{i:j} = \{(s_i, a_i), (s_{i+1}, a_{i+1}) \cdots (s_j, a_j)\}$$

- reward is defined over a trajectory $R(\bar{s})$, could be discounted
- Markovian assumption does not hold

The learning problem is finding the optimal policy π_θ such that the following is maximized

$$\mathbf{E}_{\bar{s}}[R(\bar{s})]$$

Policy Gradient without Markovian Assumption

Theorem 1:

Even without the Markovian assumption, the policy gradient can be found through

$$\nabla_\theta \mathbf{E}_{\bar{s}} R(\bar{s}) = \mathbf{E}_{\bar{s}} \left[R(\bar{s}) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right] = \mathbf{E}_{\bar{s}} \hat{\nabla}(\bar{s})$$

We can guarantee faster convergence through baseline subtraction.

Variance Reduction through Baseline Subtraction

We can redefine the estimator $\hat{\nabla}(\bar{s})$ such that

$$\hat{\nabla}(\bar{s}) = \sum_{t=1}^{T} (R(\bar{s}) - b_t) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Variance Reduction through Baseline Subtraction

Lemma 1:

For all t, let b_t be a scalar that doesn't depend on $a_t, \bar{s}_{t+1:T}$, then

$$\mathbf{E}_{\bar{s}}\left[\sum_{t=1}^{T} b_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right] = 0$$

Using Lemma 1,

$$\mathbf{E}_{\bar{s}}\hat{\nabla}(\bar{s}) = \mathbf{E}_{\bar{s}}\left[\sum_{t=1}^{T}(R(\bar{s}) - b_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] = \nabla_\theta \mathbf{E}_{\bar{s}} R(\bar{s})$$

So it is still an unbiased estimator, but the variance changes based on how b_t are chosen.

Solving for optimal baseline

We can rewrite the estimator in matrix form, assuming R is a matrix of size T with all $R(\bar{s})$, $b = \{b_t\}_{t=1}^{T}$ and $F = \{\nabla_\theta \pi_\theta(a_t|s_t)\}_{t=1}^{T}$

$$\hat{\nabla}(\bar{s}) = \sum_{t=1}^{T}(R(\bar{s}) - b_t)\nabla_\theta \log \pi_\theta(a_t|s_t) = (R - b)F$$

Solving for minimum variance $\mathbf{Var}\hat{\nabla}(\bar{s})$ is the same as solving to minimize $\hat{\nabla}(\bar{s})^2$ [4], so

$$\frac{\partial}{\partial b}(R - b)F(R - b)F := 0$$

This can be used to solve for b, and R, F can be approximated over a minibatch of episodes.

Other ways: Online linear regression

Policy Gradient Variants without Markovian Assumption

The authors note that there are some papers [5] that use another function instead of $R(\bar{s})$, called the Q function, defined as

$$Q_\theta(\bar{s}_{1:t}) = \sum_{\bar{s}_{t+1:T}} P_\theta(\bar{s}_{t+1:T}|\bar{s}_{1:t})R(\bar{s})$$

Lemma 2:

Even without the Markovian assumption,

$$\nabla_\theta \mathbf{E}_{\bar{s}} R(\bar{s}) = \mathbf{E}_{\bar{s}}\left[\sum_{t=1}^{T} Q_\theta(\bar{s}_{1:t})\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

Safe Reinforcement Learning

Objectives that involve expectation are common in machine learning. Even for RL, our objective is to maximize $\mathbf{E}_{\bar{s}} R(\bar{s})$. The authors **argue** that this poses a functional safety problem.

Suppose that

1. $R(\bar{s}) = -r$ with probability p (accident)
2. with probability $1 - p$, $R(\bar{s})$ sums to something in $[-1, 1]$

The contribution of accident trajectories on $R(\bar{s})$ is the term $-pr$. Hence, this term should not be negligible.

$$pr \gg 1, \text{ or } r \gg \frac{1}{p}$$

Lemma 3:

With the requirement

$$r \gg \frac{1}{p}$$

we have

$$\mathbf{Var}R(\bar{s}) = \Omega(pr^2)$$

This makes finding the estimator for the objective harder.

New Architecture for Safe Reinforcement Learning

The authors decompose the policy π_θ into two parts:

1. π_θ^D, maps S to desires, learnable
2. π^T, maps desires to final trajectory, non learnable

So, first we learn a policy π_θ^D from experience by maximizing an expected reward, and then, we find a trajectory keeping in mind hard constraints on functional safety, using π^T.

$$\pi_\theta = \pi^T \circ \pi_\theta^D$$

Example: Double Merge Scenario

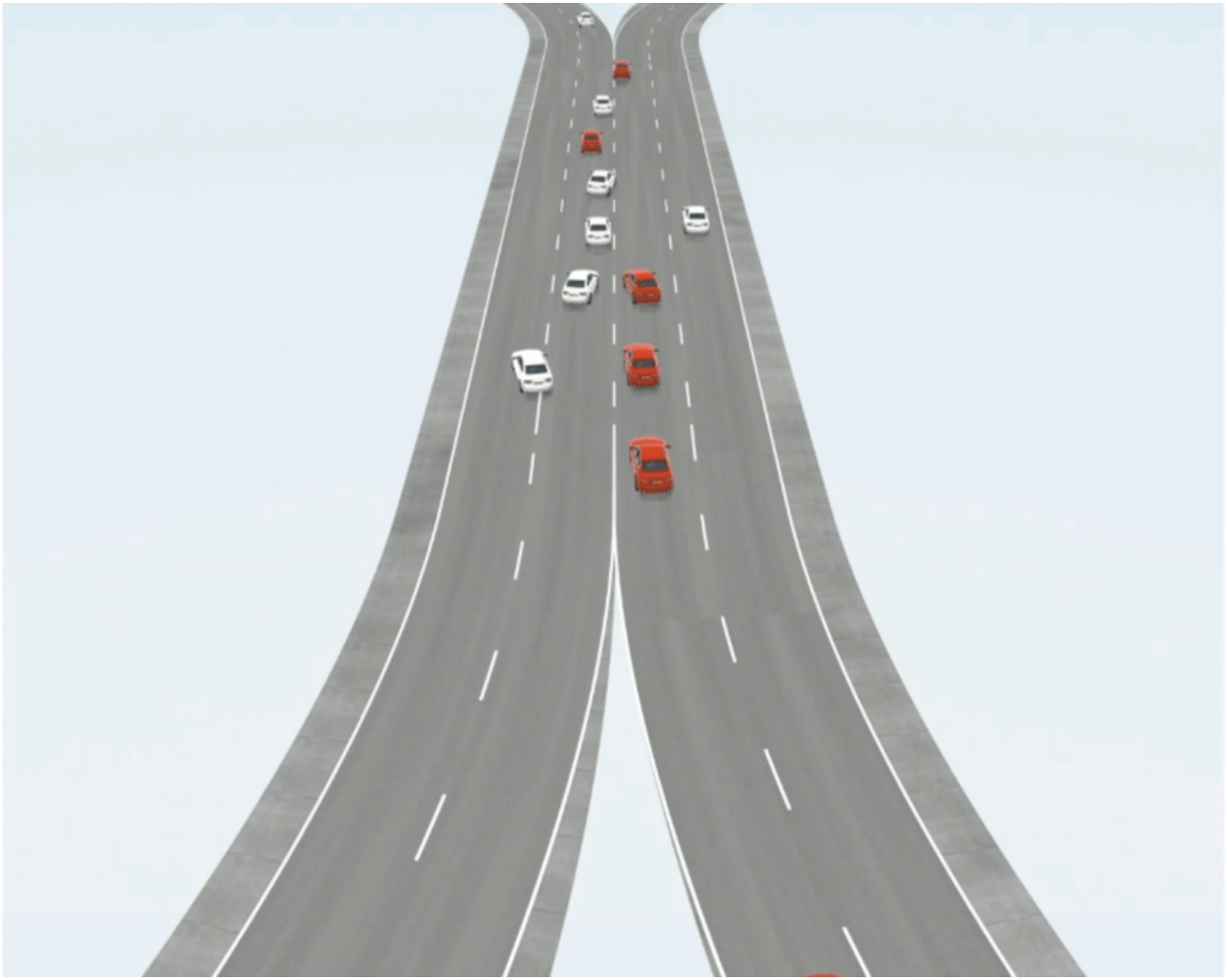


Figure 1: Source: [6]

Example: Double Merge Scenario

Let's define the desires through 3 parameters

1. Our speed, in $[0, v_{max}]$
2. Our lateral position in the lanes, $L = \{1, 1.5, 2, 2.5, 3, 3.5, 4\}$
3. Our approach with respect to any other vehicle, $\{g, t, o\}$

$D = [0, v_{max}] \times L \times \{g, t, o\}^n$, where there are n other vehicles.

Example: Double Merge Scenario

The 3 costs are:

1. Cost for speed

$$C_s = \sum_{i=2}^{k} (v - \frac{1}{\tau} ||(x_i, y_i) - (x_{i-1}, y_{i-1}||)^2$$

2. Cost for desired lateral position

$$C_l = \sum_{i=1}^{k} d(x, y, l)$$

3. Cost due to other vehicles: If the car is tagged g, we should arrive at least 0.5 seconds after it, $\tau i > \tau j + 0.5$, and the opposite if the car is tagged t. If an offset has to be maintained the trajectories to not intersection.

$$C_{v_i} = \phi(\tau(j - i) + 0.5)$$

Example: Double Merge Scenario

Total cost would be

$$C = \lambda_1 C_s + \lambda_2 C_l + \sum_i \lambda_{3i} C_{v_i}$$

This is the cost for π^T, the policy that maps desires to trajectories. It also uses some hard constraints like:

- Do not allow host vehicle to be off roadway
- Do not allow host vehicle to be arbitrarily close to any other vehicle

This part is not learned, and it is obtained by solving the optimization problem with the given cost C.

Temporal Abstractions

The part that's left is learning the policy π_θ^D, which maps the state space to desires.

The authors note that it is necessary to divide decision making into components because:

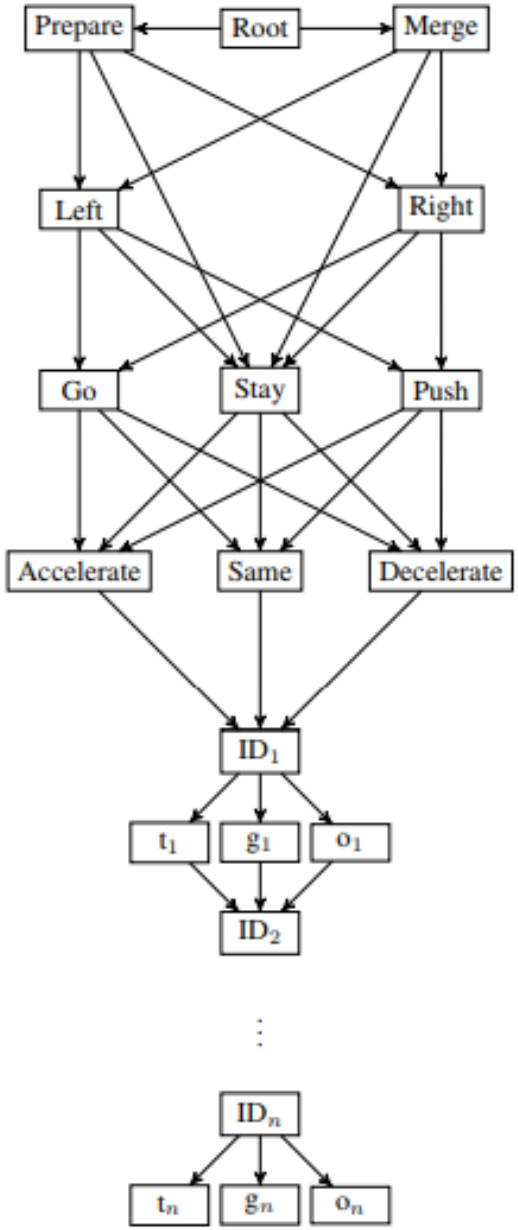
1. desires D might be too large, even continuous
2. the variance of policy gradient estimator grows with time horizon T

So, the authors use the options framework [7, 8].

Why?

1. interpretability of results
2. decomposing the stucture means smaller horizons and possibilities per node, smaller variance

Temporal Abstractions



Experiments

- proprietary software and data
- $R(\bar{s})$ has to take care of comfort, as well as whether merge succeeded
- each agent can observe location, velocity and heading of other agents within 100m
- use optimization based on dynamic programming to solve for π^T
- initialize all policies in option graph through imitation learning
- each policy represented by NN of 3 hidden layers
- for π_θ^D, authors use two sets of agents A and B (**self play**)
- use one set as reference agents, run policy gradient on other until convergence, then switch

Conclusions

- policy gradient doesn't really need the markovian assumption; we can initialize through imitation
- the variance of the gradient depends not just on $R(\bar{s})$, but also on time horizon
- policy π_θ can be decomposed into a learnable π_θ^D and a non learnable π^T
- the learnable π_θ^D can be learnt using an options graph (temporal abstraction); this reduces time horizon and complexity
- the non learnable π^T can be solved for a specified cost function using say dynamic programming

References I


H. B. Ammar, R. Tutunov, and E. Eaton.
Safe policy search for lifelong reinforcement learning with sublinear regret.
In *International Conference on Machine Learning*, pages 2361–2369, 2015.


J. Hu and M. P. Wellman.
Nash q-learning for general-sum stochastic games.
Journal of machine learning research, 4(Nov):1039–1069, 2003.


M. L. Littman.
Markov games as a framework for multi-agent reinforcement learning.
In *Machine Learning Proceedings 1994*, pages 157–163. Elsevier, 1994.


J. Peters and S. Schaal.
Reinforcement learning of motor skills with policy gradients.
Neural networks, 21(4):682–697, 2008.


J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel.
High-dimensional continuous control using generalized advantage estimation.
arXiv preprint arXiv:1506.02438, 2015.

References II


S. Shalev-Shwartz, S. Shammah, and A. Shashua.
Safe, multi-agent, reinforcement learning for autonomous driving.
arXiv preprint arXiv:1610.03295, 2016.


M. Stolle and D. Precup.
Learning options in reinforcement learning.
In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.


R. S. Sutton, D. Precup, and S. Singh.
Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning.
Artificial intelligence, 112(1-2):181–211, 1999.