2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Improved Robustness and Safety for Autonomous Vehicle Control with Adversarial Reinforcement Learning

Xiaobai Ma, Katherine Driggs-Campbell, and Mykel J. Kochenderfer

Abstract— **To improve efficiency and reduce failures in autonomous vehicles, research has focused on developing robust and safe learning methods that take into account disturbances in the environment. Existing literature in robust reinforcement learning poses the learning problem as a two player game between the autonomous system and disturbances. This paper examines two different algorithms to solve the game, Robust Adversarial Reinforcement Learning and Neural Fictitious Self Play, and compares performance on an autonomous driving scenario. We extend the game formulation to a semi-competitive setting and demonstrate that the resulting adversary better captures meaningful disturbances that lead to better overall performance. The resulting robust policy exhibits improved driving efficiency while effectively reducing collision rates compared to baseline control policies produced by traditional reinforcement learning methods.**

I. Introduction

Robust control has a long history of providing guaranteed stability despite disturbances and modeling errors [1]. Such methods aim to achieve robust performance, given bounded disturbances or errors. In a learning setting, the controls community has also used reachability tools to synthesize robust controllers [2]. These safe learning approaches provide guarantees on safe interaction and efficiency, but tend to be overly conservative [3], [4].

The reinforcement learning community has also addressed robustness through adversarial learning [5], [6]. Instead of optimizing for expected reward, the system is optimized to reduce risk (i.e., maximize the rewards associated with the worst case trajectories [7]). Traditional reinforcement learning methods sample trajectories from the simulator's natural distribution, meaning that the probability of dangerous rollouts is very low. Thus, even with a large negative reward on failures, the expected reward could still be high due to averaging. The rareness of such trajectories makes it hard to train effective polices, which is undesirable for risk-sensitive tasks.

One solution is to bias the rollouts towards the worst case trajectories (e.g., trajectories from the tails of the disturbance distribution), and incorporate these samples in the training process. Rajeswaran et al. propose to sample a collection of trajectories from a source distribution and use a subset of low percentile trajectories to update the policy [8]. This method effectively biases the training towards effectively handling extreme disturbances, but is not sample efficient. Another approach is to formulate the training as a two-player game between the system we wish to control and the environment disturbance. This method is referred to as Robust Adversarial Reinforcement Learning (RARL) [9].

There is a trade-off between expected efficiency and robustness for safe driving. Intuitively, if the worst case scenario is assumed in the design process, then the resulting controller will be overly cautious [3]. In this work, we examine both worst-case performance, as is typically assessed for robust methods, as well as average reward to quantify this trade-off and compare different learning methods.

While these methods address some safety problems, reinforcement learning approaches require simulation and therefore often fail to generalize to the real world. This drawback is apparent in adversarial settings where the adversary may not capture real-world disturbances.

There has been a great deal of recent research focusing on how to model and train the adversary from a learning and game theoretic perspective. One method called Fictitious Self-Play (FSP) solves for the Nash Equilibrium in this two players game using a reservoir buffer to train the adversary [10]. Not only does this approach address some of the problems of overly cautious behavior and unrealistic disturbances by "averaging" the adversarial disturbances used in simulation, but also provides theoretical guarantees.

The RARL and FSP approaches do not necessarily capture real-world disturbances, as we will demonstrate in this paper. Moreover, these methods do not necessarily improve safe transfer for model mismatch and disturbances [11].

In this work, we extend these adversarial learning methods to a semi-competitive game setting, where the adversary in the two-player game is incentivized to adjust the disturbance magnitude according to the current capability of the control policy. We show that this not only improves safety, but also improves robustness under different environment models. We compare this trade-off between expected efficiency and safety for different game formulations and algorithms.

Finally, we apply our method to a transfer learning setting, where the training and testing are executed in different environments or with different dynamical models. To do this, we propose a framework for training the system against an adversary that accounts for modeling errors and mismatch between the simulation used in training and real-world dynamics. These real-world dynamics are derived from a test vehicle, which differs significantly from the model used to train the controller in simulation. In doing so, we not only improve the performance, but also the computational efficiency of the learning and training process. In summary,

This material is based upon work supported by SAIC Innovation Center, a subsidiary of SAIC Motors.

X. Ma, K. Driggs-Campbell, and M.J. Kochenderfer are with the Aeronautics and Astronautics Department, Stanford University, Stanford, CA, USA (e-mail: {maxiaoba,krdc,mykel}@stanford.edu).

this paper presents the following contributions:

1) We implement robust learning methods and assess the efficiency and safety trade-off for autonomous driving;
2) We augment and improve existing game theoretic frameworks by adding non-competitive incentives to actively adapt the disturbance magnitude; and
3) We demonstrate how these robust methods can account for model errors to improve transfer learning from simulation to real-world dynamics.

II. Problem Formulation

Suppose we would like to control a vehicle governed by a dynamical equation:

$$x[k+1] = f(x[k], u[k]) + g(x[k], u[k], d[k]) \qquad (1)$$

where f is the dynamics of the vehicle, $x[k] \in \mathbb{R}^n$ is the vehicle state at time $k \in \mathbb{N}$, $u[k] \in \mathbb{R}^m$ is the control input, and g is a function that captures the uncertainty in the system and the external environment as a function of the state, input, and disturbances $d[k]$.

A precise dynamical model may be unknown or difficult to compute. We approximate the vehicle dynamics with a simple discrete time bicycle model and noise model:

$$x[k+1] = \hat{f}(x[k], u[k]) + v(x[k], u[k], d[k]) \qquad (2)$$

where $\hat{f}$ is an approximate model of the true dynamics, v is a function that attempts to capture modeling error and environment disturbances, and all other variables are as previously described. Section III provides further details.

One approach to train a policy that is robust to model mismatch and noise is by introducing a two-player Markov game [9], which is a special case of a stochastic, dynamic game, for which one or more players have probabilistic transitions between states. In the Markov game, it is assumed that the distribution of next state of the game only depends on the current state and action. In this formulation, the modeling error and uncertainties are modeled as external disturbances. Player 1 (the protagonist) is trained to effectively control the vehicle, and the external disturbances are generated by player 2 (the adversary).

A. Game Theoretic Formulation

This two player game can be expressed as a tuple $(S, A_1, A_2, P, r_1, r_2, \gamma, s_0)$, where S is the state space of the vehicle initialized at s_0; A_1 is the action space for protagonist, which will be the acceleration and steering commands ($u[k]$ in eq. (2)); A_2 is the action space for the adversary, which will be the disturbances ($v(\cdot)$ in eq. (2)); $P : S \times A_1 \times A_2 \times S \rightarrow \mathbb{R}$ is the transition probability density defined by equation eq. (2); $r_1 : S \times A_1 \times A_2 \rightarrow \mathbb{R}$ and $r_2 : S \times A_1 \times A_2 \rightarrow \mathbb{R}$ are the rewards for protagonist and adversary; and γ is a discount factor.

The reward function determines the type of game and has a significant impact on equilibria, and therefore performance of the resulting policy. We consider the following two types of games: strictly competitive and semi-competitive.

1) Strictly Competitive Games: In the existing literature on robust adversarial learning, the game is generally posed as a zero-sum or strictly competitive game, where the protagonist and the adversary are in direct opposition [9], [10]. Specifically, the reward for the adversary is the opposite of the protagonist: $r_1 = -r_2$, where the protagonist is maximizing γ-discounted rewards while the adversary is minimizing them. With this setup, the adversary is helping to sample worst case trajectories for the protagonist. In RARL, the adversary's role is to sample trajectories from the worst α-quantile. The α parameter is determined by the magnitude of the disturbance available to the adversary [9]. One drawback of a zero-sum game is that a disturbance at maximum magnitude is almost always the best output for the adversary, as it disrupts the system most.

In our experiments, as will be presented in Sections IV and V, we observed three key different failures caused by training in a strictly competitive setting: (1) The protagonist becomes too conservative, thus exhibiting poor performance.; (2) The protagonist is unable to recover from failures early in the training processing and falls into a local minimum at a low performance region; and (3) The adversary becomes predictable by consistently outputting maximal disturbance, causing the protagonist to overfit and thus poorly generalizes over different disturbance distributions.

2) Semi-Competitive Games: We propose augmenting the traditional zero-sum game approach to a semi-competitive or nonzero-sum setting, where the adversary has both a competitive and a cooperative component in its reward function. Formally, we change the adversary reward to $r_2 = -r_1 + r_c$, where r_c is a cooperative reward that actively regulates the disturbance magnitude.

In the early stages of training when the protagonist is relatively weak, disturbances with different magnitude receive similar competitive reward $-r_1$. To earn some cooperative rewards, the adversary would prefer weak disturbances. As the protagonist iteratively learns and becomes more robust, a weak disturbances receives less competitive reward, while a stronger disturbance with higher competitive reward but lower cooperative reward is preferable. Thus, the adversary would need to trade-off between the competitive and cooperative rewards. The desired disturbance distribution would aim to cause a failure while using the minimal disturbance magnitude. This disturbance is also desired for the protagonist since it is easier for the protagonist to recover. By adding a cooperative reward, we make the adversary adaptively adjust its disturbance magnitude such that it allows the protagonist to effectively learn, while being adversarial.

While there are many forms which r_c may take, for the adversary that outputs disturbances following a Gaussian distribution, we found that a simple rectangle function $r_c = r_a \cdot \{|d[k]| < d_{max}\}$ works well, where $d[k]$ is the disturbance to be applied, d_{max} is the maximum allowed disturbance, and r_a is a positive constant. The magnitude of r_a is a weighting between the competitive and cooperative reward, and is inverse related to the α-percentile that would be sampled.

Algorithm 1 RARL Training Procedure [9]

```
Input: Environment $E$; Stochastic policies $\pi_P^{(0)}$ and $\pi_A^{(0)}$
for $i = 1, \dots, N_{iter}$
    $\pi_P^{(i,0)} \leftarrow \pi_P^{(i-1)}$
    for $j = 1, \dots, N_1$
        $paths \leftarrow$ ROLLOUT$(E, \pi_P^{(i,j-1)}, \pi_A^{(i-1)})$
        $\pi_P^{(i,j)} \leftarrow$ POLICYOPTIMIZER$(\pi_P^{(i,j-1)}, paths)$
    $\pi_P^{(i)} \leftarrow \pi_P^{(i,N_1)}$
    $\pi_A^{(i,0)} \leftarrow \pi_A^{(i-1)}$
    for $k = 1, \dots, N_2$
        $paths \leftarrow$ ROLLOUT$(E, \pi_P^{(i)}, \pi_A^{(i,j-1)})$
        $\pi_A^{(i,j)} \leftarrow$ POLICYOPTIMIZER$(\pi_A^{(i,j-1)}, paths)$
    $\pi_A^{(i)} \leftarrow \pi_A^{(i,N_2)}$
```

Algorithm 2 NFSP Training Procedure [12]

```
Input: Environment $E$
Initialize: Reservoir buffers $M_P$ and $M_A$; best response policies for protagonist and antagonist: $\pi_{P,br}^{(0)}$, $\pi_{A,br}^{(0)}$; average policies for protagonist and antagonist: $\pi_{P,avg}^{(0)}$, $\pi_{A,avg}^{(0)}$
for $i = 1, \dots, N_{iter}$
    $\pi_{P,br}^{(i,0)} \leftarrow \pi_{P,br}^{(i-1)}$
    for $j = 1, \dots, N_1$
        $paths \leftarrow$ ROLLOUT$(E, \pi_{P,br}^{(i,j-1)}, \pi_{A,avg}^{(i-1)})$
        POPULATE$(M_P, paths)$
        $\pi_{P,br}^{(i,j)} \leftarrow$ POLICYOPTIMIZER$(\pi_{P,br}^{(i,j-1)}, paths)$
    $\pi_{P,br}^{(i)} \leftarrow \pi_{P,br}^{(i,N_1)}$
    $\pi_{P,avg}^{(i)} \leftarrow$ FIT$(\pi_{P,avg}^{(i-1)}, M_P)$
    $\pi_{A,br}^{(i,0)} \leftarrow \pi_{A,br}^{(i-1)}$
    for $k = 1, \dots, N_2$
        $paths \leftarrow$ ROLLOUT$(E, \pi_{P,avg}^{(i)}, \pi_{A,br}^{(i,j-1)})$
        POPULATE$(M_A, paths)$
        $\pi_{A,br}^{(i,j)} \leftarrow$ POLICYOPTIMIZER$(\pi_{A,br}^{(i,j-1)}, paths)$
    $\pi_{A,br}^{(i)} \leftarrow \pi_{A,br}^{(i,N_2)}$
    $\pi_{A,avg}^{(i)} \leftarrow$ FIT$(\pi_{A,avg}^{(i-1)}, M_A)$
```

B. Solution Methods

To solve this high-dimensional, complex game and train a policy with reinforcement learning, we implement two methods to find the equilibria in the game: Robust Adversarial Reinforcement Learning (RARL) and Neural FSP (NFSP).

1) RARL: As originally proposed by Pinto et al., this approach uses neural network policies to represent each of the players and iteratively trains the protagonist and the adversary against each other, alternating at each training epoch [9]. The adversarial network learns to perturb the vehicle trajectories to maximize its reward. Then, by optimizing the protagonist against this evolving adversary, hypothetically, the protagonist learns to perform robustly over the entire disturbance space. The training algorithm is shown in algorithm 1 assuming the protagonist and adversary follow stochastic policies π_P and π_A.

2) NFSP: Fictitious Self-Play is a method for finding the Nash equilibrium in two-player, multi-step games [10]. Instead of iteratively training against an optimized opponent, players choose best responses to other players average behavior. Neural Fictitious Self-Play (NFSP), listed in algorithm 2, combines FSP with neural network polices to minimize divergence in common multi-agent reinforcement learning methods [12]. An agent consists of two neural networks: one network to learn an approximate best response to the average response of other agents, and a second network that averages the agent's own historical strategies. The average response is determined by using a reservoir buffer to store the agent's past behavior.

III. EXPERIMENTAL METHODS

This section presents the experiments designed to train and test the different robust adversarial reinforcement learning algorithms as well as the implementation details.

A. Experimental Setup

We hypothesize that using robust learning techniques will not only lead to safer control policies, but will also improve performance when transferring this policy to the real-world system, which will likely operate with different dynamics.

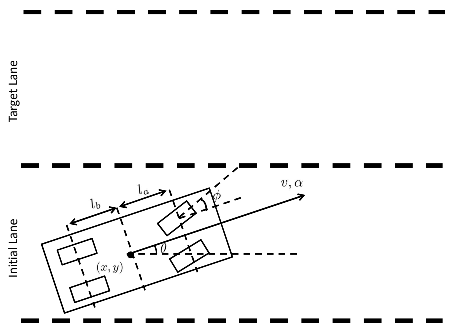


Fig. 1: Scenario representation of lane changing experiment setup.

To study this, we created a training environment where the protagonist controls the ego vehicle for lane keeping or lane changing. The ego vehicle drives on a straight three-lane roadway with infinite length, and there are no other vehicles on the road. The simulation environment is built upon a 2D traffic simulator, AutomotiveDrivingModels.jl.[1] The vehicle is initialized in the center lane with an initial velocity of 10 m/s. At the beginning of each trial, a target lane is randomly assigned. The time step of the simulation environment is 0.05s, and the maximum time duration of each path is 10s. This experiment is illustrated in fig. 1.

The Markov decision process used in the Markov game is defined as follows:

The state of the vehicle $s_t \in S$ is represented as $s_t = [x_t, y_t, v_t, \theta_t]$, where (x_t, y_t) is the position, v_t is the speed,

[1]https://github.com/sisl/AutomotiveDrivingModels.jl

and θ_t is the orientation of the vehicle.

The action of the protagonist $a_{1,t}$ is a two dimensional vector representing the acceleration and steering of the ego vehicle at each timestep, where $a_{1,t} = [\alpha_t, \phi_t]$. The acceleration is limited to -3 to $3\,\mathrm{m\,s^{-2}}$, and the steering is limited to $-20°$ to $20°$.

The action of the adversary $a_{2,t}$ is the disturbance on the acceleration and steering, $a_{2,t} = [\delta\alpha_t, \delta\phi_t]$. The magnitude of the disturbance is limited to 20% of the maximum acceleration and steering.

The state transition model, as generally stated in eq. (2) and visualized in fig. 1, follows a simple discrete time bicycle model, where $l_a = l_b = 1.5\,\mathrm{m}$ are distances between the vehicle's center of mass and its front and rear axle, $\Delta t = 0.05\,\mathrm{s}$ is the time step, and all other variables are as previously described.

The reward function of the protagonist $r_{1,t}$ is a weighted sum of the rewards on vehicle state and action:

$$r_{1,t} = 0.5r_t^v + 0.5r_t^\theta + 0.1r_t^\alpha + 0.2r_t^\phi + 1.0r_t^y \qquad (3)$$

where the reward on the velocity is given by a smooth, increasing function with range $[-1, 1]$:

$$r_t^v = \log_{\frac{v_{max} \quad v_{min}}{2}}(100 \cdot \frac{v_t - v_{min}}{v_{max} - v_{min}} + 0.99) - 1$$

The reward associated with orientation is given by:

$$r_t^\theta = \begin{cases} -1.0 & \text{if } |\theta_t| > \pi/4 \\ 0.0 & \text{otherwise} \end{cases}$$

The reward associated with the acceleration is given by:

$$r_t^\alpha = -\frac{|\alpha_t|}{\alpha_{max}}$$

The reward associated with the steering is given by:

$$r_t^\phi = -\frac{|\phi_t|}{\phi_{max}}$$

The reward associated with the vehicle position is determined by the lateral offset:

$$r_t^y = \begin{cases} 3.0 & \text{if } |y_t - y_{goal}| < 0.05 \\ 1.0 - \frac{|y_t - y_{goal}|}{l_w} & \text{otherwise} \end{cases}$$

where y_{goal} is the lateral position of the center line of the target lane and $l_w = 3\,\mathrm{m}$ is the lane width. If the ego vehicle goes off the road, meaning that a failure and collision has occurred, $r_{1,t} = -5.0$ and the trajectory ends.

In the zero-sum setting, the adversary's reward is strictly competitive, meaning $r_{2,t} = -r_{1,t}$. In the nonzero-sum setting, the adversary's reward is given by $r_{2,t} = -r_{1,t} + r_{c,t}$, where $r_{c,t}$ is the additional adversary reward taking the form:

$$r_{c,t} = r_{\delta\alpha,t} + r_{\delta\phi,t} \qquad (4a)$$

$$r_{\delta\alpha,t} = \begin{cases} r_a & \text{if } |\delta\alpha_t| < \delta\alpha_{max} \\ 0 & \text{otherwise} \end{cases} \qquad (4b)$$

$$r_{\delta\phi,t} = \begin{cases} r_a & \text{if } |\delta\phi_t| < \delta\phi_{max} \\ 0 & \text{otherwise} \end{cases} \qquad (4c)$$

where the maximum disturbance limit, $\delta\alpha_{max}$ and $\delta\phi_{max}$, is predefined. Note that $r_a = 0$ means a zero-sum setting. In the experiment, r_a is set to 3.[2]

B. Implementation Details

To compare the performance of different policies, we train five different networks. First, we train a baseline policy using traditional reinforcement learning. Then, we train policies using the RARL and the NFSP methods in a strictly competitive and a semi-competitive settings. We used the following parameters for training.

1) Baseline Policy: The baseline policy is trained with Trust Region Policy Optimization (TRPO) as implemented by RLLab with no disturbances in the environment [13], [14]. The baseline uses a Gaussian Multilayer Perceptron architecture with hidden layer sizes of 256, 128, 64, and 32 with ReLu nonlinearity activation functions. The training is conducted over 15000 iterations with a batch size of 4000 and a step size of 0.01.

2) RARL: For training the policies with RARL, we use the same policy structure as the baseline for both the protagonist and the adversary. The protagonist is initialized with a baseline policy trained with 5000 iterations, and the adversary is randomly initialized. The training is conducted over 4000 iterations, and $N_1 = N_2 = 5$, which is the number of policy updates performed for the protagonist and adversary policies in each iteration. The batch size is 400. The policy optimizer and other parameters are consistent with the baseline. The training parameters are chosen so that the training time for the baseline (starting from 5000 iterations) and RARL is approximately the same.

3) NFSP: For NFSP training, we use the same network structure as the baseline for the protagonist and adversary's best response policy, P_{br} and A_{br}. The average response policies, P_{avg} and A_{avg}, use RLLab's Gaussian Multilayer Perceptron Regressor implementation. The hidden layer sizes and nonlinearities are consistent with the baseline. The reservoir buffers M_P and M_A have size 2×10^4. Other parameters are consistent with RARL.

C. Validation Dynamics

We hypothesize that using the proposed robust methods will improve performance when the policy is executed on a real vehicle, despite modeling errors and noise. This hypothesis is validated using a dynamical model closer to the test vehicle provided by the SAIC Innovation Center. Figure 2 shows the test vehicle.

While testing on the vehicle, we observed that much of the model mismatch was caused by the steering response of the vehicle. In training, we assume that the steering of the vehicle perfectly tracks the the command steering, which is not true in practice. The limits on the steering and angular rate is an important constraint that is often tuned and modified in the low-level controller on the vehicle.

[2]The magnitude of r_a is inversely related to the α-percentile that would be sampled. We ran several experiments on different r_a values and found that 3 gives the best result.

Fig. 2: Test vehicle provided by SAIC Innovation Center.

For vehicular control, there are many vehicle specific model parameters that significantly impact the dynamics, such as the axle distance (l_a and l_b shown in fig. 1). The resulting policy should be robust to model mismatch with respect to changes in these parameters so that it is applicable on different vehicles.

D. Evaluation Metric

To test the control policy, we run each policy for 500 rollouts. We need two metrics to evaluate the efficiency and the safety of the polices. For efficiency, we use the the average undiscounted path reward introduced in section III-A, which is a combined metric on difference aspects of the driving. We subtract the collision part of the reward to make it orthogonal to the safety metric. For safety, we use the collision rate which is the number of failures divided by the number of rollouts.

IV. Robustness to Adversarial Disturbances

To validate our methods and demonstrate robustness, we conduct several tests with different disturbance distributions and vehicle dynamics to evaluate the performance of the trained protagonists. First, we test the robust policy with a disturbance that follows a Pareto distribution. Second, we show robustness to an adversarial disturbance.

A. Pareto Disturbance Test

To validate the systems' robustness, we test the policy using a heavy tailed distribution. The disturbances on acceleration and steering are sampled from a Pareto distribution with shape parameter β that varies from 1 to 10 and scale parameter $x_m = 1$.[3] The probability density function is $\frac{\beta x_m^\beta}{x^{\beta+1}}$ if $x > x_m$ and 0 otherwise. As β increases, the disturbance is more concentrated on boundaries. This heavy-tailed distribution captures extreme disturbances, giving insight into the worst case performance.

Figure 3 gives the results under the disturbance described above. The horizontal-axis is the shape parameter β and the vertical-axes are the average reward and collision rate. As the disturbance concentrates more on the boundary (β increases), the reward of baseline drops more rapidly than other policies, which shows the robustness of game theoretic

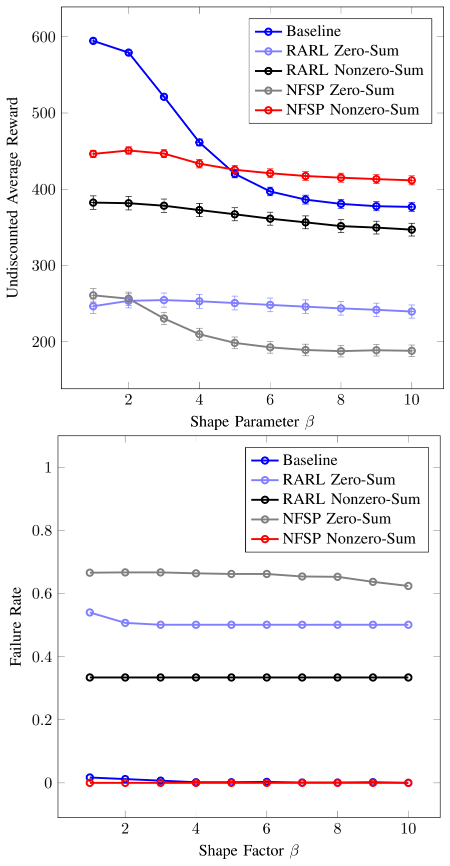


Fig. 3: Average path reward (top) and failure rate (bottom) of different policies under Pareto disturbance with increasingly heavy-tailed shape parameter, β.

methods. Policies trained with nonzero-sum rewards show better performance than ones trained with zero-sum, demonstrating the influence of the additional adversary rewards. The NFSP policy for the nonzero-sum game implementation demonstrates the most robust performance and highest reward when the variance is high.

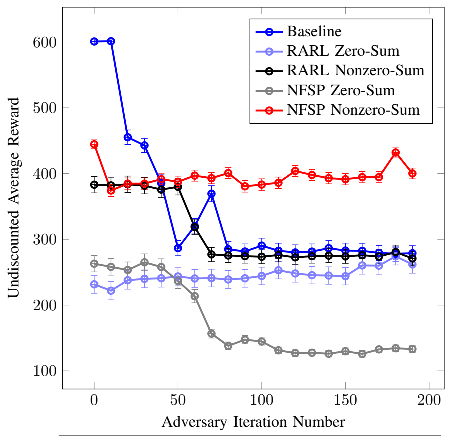
Further, the NFSP nonzero-setting policy exhibits the safest performance as well; the resulting policy always results in no collisions. The baseline policy performs nearly as well, despite more rapidly degrading performance in terms of expected reward.

B. Adversarial Disturbance Test

Inspired by RARL [9], we test the robustness of the policy with adversarial disturbances. We train an adversary to apply disturbance on acceleration and steering while holding the protagonist's policy constant (for NFSP, this is the best response policy of the protagonist). The training setup is the same as RARL. We then test the protagonist under the disturbance generated by this adversary.

Figure 4 shows the average path rewards and collision rate as a function of the adversary's training iterations. The policy trained by NFSP with $r_a = 3$ in the nonzero-sum setting shows strong robustness to the adversarial disturbance. The

[3]This distribution gives the normalized disturbance magnitude. In testing, this value is scaled to 20% of the maximum acceleration or steering. Additionally, half of the tests use the opposite of the sampled disturbance.



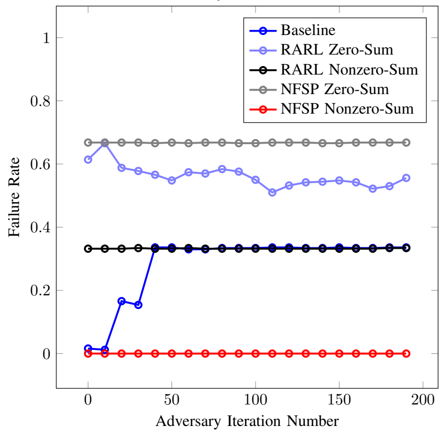


Fig. 4: Average path reward (top) and failure rate (bottom) of different policies under adversarial disturbance in which an adversary iteratively trains against the protagonist.

failure rate remains zero, while the expected reward is approximately 400. The baseline policy initially has a high average reward and zero collision rate, but its performance drops rapidly as the adversary becomes stronger. Policies trained with RARL result in rewards similar to the baseline, while the policy with non-zero formulations exhibit better performance with respect to the failure rate. These results illustrate how reformulating adversarial learning as a nonzero-sum game improves overall performance.

V. Robustness to Model Mismatch

We apply the policy to the real-world dynamics described in the previous section. Using the same metrics as before, we quantify the trade-offs between the policies. We test the policies on a limited steer change model that approximates the real vehicle behavior (LSC test), and a setting where we change the axle distance for the vehicle (DA test).

A. Limited Steer Change Test

In this test, we limit the steer change of the ego vehicle in one time step ($0.05\,\mathrm{s}$) to $4.5°$. We also add a uniformly distributed disturbance in the same limit as previous tests. This test shows the policies robustness against model error.

As shown in fig. 5, the proposed methods shows a reduced average reward compared to the baseline. While many of these adversarial learning methods show degraded safety, the NFSP solution in the nonzero-sum formulation improves safety with only a slightly degraded expected reward.

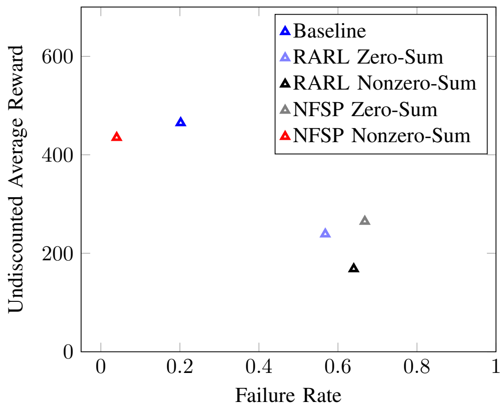


Fig. 5: Scatter plot showing the trade-off between average path reward and failure rate of different policies under limited steer change constraint and uniform disturbance.

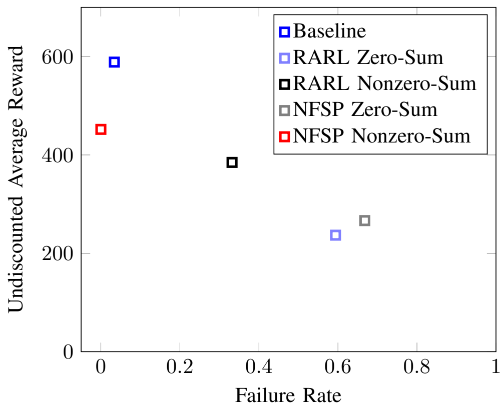


Fig. 6: Scatter plot showing the trade-off between average path reward and failure rate of different policies under different axle distance and uniform disturbance.

B. Different Axle Distance Test

In this test, we vary the vehicle's axle distance l_a and l_b uniformly from 0.5 to $2.5\,\mathrm{m}$. Recall that during training, the axle distance is fixed to $1\,\mathrm{m}$. Again, uniformly distributed disturbance is also added to the inputs. The results are shown in fig. 6.

This test exhibits similar results as was previously observed. While many of the adversarial methods do not improve performance relative to the baseline, the NFSP nonzero-sum formulation improves safety with only a slight sacrifice in expected reward.

C. Discussion

To summarize the findings of this work, fig. 7 illustrates the efficiency and safety trade-off for tested methods across all validation tests. The color indicates the policy and formulation and the shape indicates the robustness test.

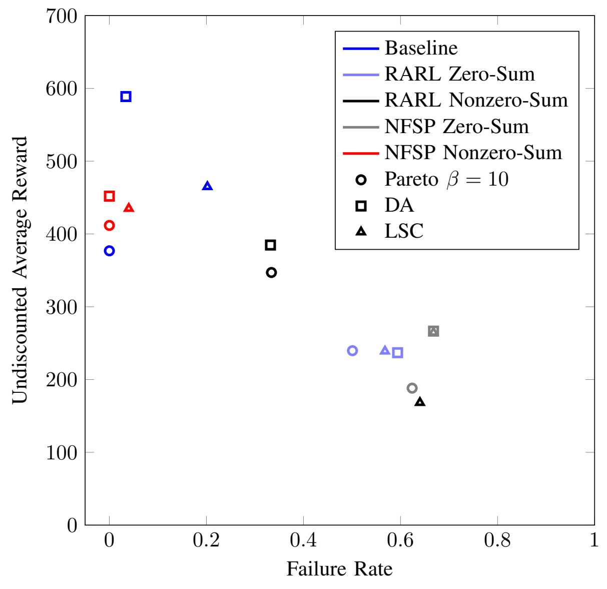


Fig. 7: Scatter plot showing the trade-off between average path reward and failure rate of different policies for all tests.

Generally, the performance variance of policies trained with robust learning methods is smaller than that of the baseline. We also observe that augmenting previous solution methods with a nonzero-sum formulation significantly improves robustness with respect to efficiency and safety. Overall, the NFSP method in the nonzero-sum formulation results exhibit the best performance with regard to maintaining safety with minimal sacrifice in efficiency.

Interestingly, most robust learning methods degrade both efficiency and safety, exhibiting worst performance than baseline. This phenomenon is likely a result of the game theoretic formulation and solution methods converging on suboptimal equilibria. Thus, our hypothesis that using robust learning methods improves robustness and efficiency is confirmed, but only by using semi-competitive games and the NFSP solution method.

VI. Conclusions

In this paper, we tested a variety of the two player Markov game formulations for robust reinforcement learning, varying the training methods (RARL and NFSP) and reward structure (zero-sum and nonzero-sum with additional adversary reward). The resulting policies were validated under different disturbance distributions and different vehicle dynamic models. Under all tests, the NFSP with nonzero-sum rewards shows overall strong robustness, safety, and more conservative performance than the TRPO single agent learning baseline. This method is also significantly more consistent between tests than the other approaches, which implies that the policy is more generalizable.

The other approaches (RARL variants and NFSP solving a zero-sum game) exhibited poor performance in both safety and efficiency metrics. This behavior is likely induced by the adversary becoming too "strong" for the current protagonist to recover from, and thus converges to a low performance point. However, with the addition of the cooperative reward and the averaging from fictitious self play, the protagonist and the adversary policies evolve more smoothly and converge more closely to the game's equilibrium. We will expand this work to a multi-player setting to capture a wider variety of disturbances and test more solution methods that can be applied to multi-player games like Policy-Space Response Oracles [15]. We will also explore the application of this robust training method on the test vehicle.

Acknowledgements

We thank Alex Kuefler and Kunal Menda for inspiring this work and the members of the Stanford Intelligent Systems Laboratory for their help on the implementation.

References

[1] T. Başar and P. Bernhard, *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer, 2008.
[2] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *IEEE Conference on Decision and Control (CDC)*, 2014, pp. 1424–1431.
[3] J. H. Gillula and C. J. Tomlin, "Reducing conservativeness in safety guarantees by learning disturbances online: iterated guaranteed safe online learning," *Robotics: Science and Systems VIII*, p. 81, 2013.
[4] K. Driggs-Campbell, R. Dong, S. S. Sastry, and R. Bajcsy, "Robust, informative human in the loop predictions via empirical reachable sets," in *arXiv: 1705.00748*, 2017.
[5] D. Lowd and C. Meek, "Adversarial learning," in *ACM International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 641–647.
[6] P. Laskov and R. Lippmann, "Machine learning in adversarial environments," *Machine Learning*, vol. 81, no. 2, pp. 115–119, 2010.
[7] A. Tamar, Y. Glassner, and S. Mannor, "Optimizing the CVaR via Sampling," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2015, pp. 2993–2999.
[8] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," *arXiv: 1610.01283*, 2016.
[9] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," *arXiv: 1703.02702*, 2017.
[10] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *International Conference on Machine Learning (ICML)*, 2015, pp. 805–813.
[11] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
[12] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," *NIPS Deep Reinforcement Learning Workshop*, 2016.
[13] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning (ICML)*, 2015, pp. 1889–1897.
[14] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning (ICML)*, 2016, pp. 1329–1338.
[15] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, J. Perolat, D. Silver, T. Graepel, *et al.*, "A unified game-theoretic approach to multiagent reinforcement learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 4193–4206.