# Summary on Robust Adversarial Reinforcement Learning

Siddharth Nayak

## 1 Introduction

Most of the RL agents are trained on simulations of the real world scenarios for example CARLA simulator for cars, Robots in MuJoCo. But there are differences in the simulation physics and the real world. Thus, sometimes the policies learned in simulations fail to transfer to real robots. Also, we cannot train the agent on real world data completely as this would lead to lesser number of data-points experienced by the agent. Thus to tackle these issues, this paper proposes the idea of robust adversarial reinforcement learning (RARL). This algorithm improves the training stability of the agent even in the presence of additional unknown disturbances. It also improves the the robustness of the agent towards differences in training and testing cases.

## 2 Model

The algorithm is a Generative Adversarial Network(GAN - by Goodfellow et.al.) inspired model where there are two agents: Protagonist and Adversary. The protagonist tries to increase the score whereas the adversary tries to decrease the score. The adversary is like a disturbance to the (protagonist)agent. The protagonist and the adversary play a zero-sum game with a minimax objective function.

### 2.1 Terminology for two-player zero-sum games

The game is an MDP which is expresses with the tuple: $(S, A_1, A_2, P, r, \gamma, s_0)$ where $A_1$ and $A_2$ are the continuous set of actions the players can take. $P : S \times A_1 \times A_2 \times S \to R$ is the transition probability density and $r : S \times A_1 \times A_2 \to R$ is the reward for both the players. If player 1 (protagonist) is playing strategy $\mu$ and player 2 (adversary) is playing the strategy $\nu$, the reward function is $r_{\mu\nu} = \mathbf{E}_{a^1 \sim \mu(.|s), a^2 \sim \nu(.|s)}[r(s, a^1, a^2)]$. A zero-sum two-player game can be seen as player 1 maximizing the $\gamma$ discounted reward while player 2 is minimizing it.

### 2.2 The equations governing the algorithm

The policies $\mu$ and $\nu$ are the only learnable parameters in the algorithm. The reward for the protagonist is $R^1 \equiv R^1(\mu, \nu)$. And similarly the reward for the adversary is $R^2 \equiv R^2(\mu, \nu) = -R^1(\mu, \nu)$. The optimal equilibrium reward for the system is $R^{1*} = \underset{\nu}{\min}\underset{\mu}{\max} R^1(\mu, \nu) = \underset{\mu}{\max}\underset{\nu}{\min} R^1(\mu, \nu)$

### 2.3 The pipeline:

To get a stable solution the authors propose a method to train the protagonist and the adversary in an alternating iterative fashion. In the first phase the protagonist policy is learned while holding the adversary policy fixed. Next, the protagonist policy is fixed and the adversary policy is learned. This alternative method is used until convergence.

## 3 Experiments

The authors have demonstrated the robustness of RARL by: (1) training with different initializations; (2) testing on different external conditions; (3) testing with adversarial disturbances. All the experiments are done in OpenAI gym with MuJoCo physics simulator on Inverted Pendulum, Half Cheetah, Swimmer, Hopper and Walker2D environments. RARL performs better than the vanilla TRPO(by Schulman et.al.) even when there are no disturbances in the system. The mass of the environment objects are changed(as a disturbance to the system). Even in this case the RARL performs better. The baseline TRPO has a much bigger standard deviation than RARL which clearly shows the robustness of the algorithm.

## 4 Possible Improvements

I feel that one of the main drawbacks of this paper is that even though it tries to tackle the problem of policy transfer from simulation to real-world robots, the authors have not evaluated the performance of the algorithm on real-world robots. One of the possible extensions that we could try out is, combining the Adversarially Robust Policy Learning (ARPL-by Mandlekar et. al) . In ARPL the states of the system are perturbed adversarially according to the gradient of the policy with respect to the states. I feel that combining this may result in a much better robust policy as it may have the advantages of both the algorithms. Also, RARL can be combined with the Residual Reinforcement Learning (by Johannink et.al.) where the control is made by both a classic controller as well as an RL agent. RARL which generalises better towards irregularities can be used to control the RL part of the controller for more robustness. Also, one can use model uncertainty estimates as done in Safe Reinforcement Learning with Model Uncertainty Estimates (by Lutjens et. al.) can be used to get to know what the protagonist and the adversary know and do not know. This will give insights about the vulnerabilities of the protagonist and thus a suitable adversary can be designed to train the protagonist and make it more safe than before.

## 5 Conclusion

The use of a GAN-like structure in the algorithm was quite creative. The algorithm has been proven to be robust to variations in the environment like addition of friction, changes in the mass of the interacting objects, etc. The addition of the adversary agent makes the system sample hard trajectories and thus makes it learn on those hard paths. This makes the agent more robust to the variations in the environment.

## 6 References

L.Pinto, J.Davidson, R.Sukthankar, A.Gupta. Robust Reinforcement Learning . Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

J.Schulman, S.Levine, P.Moritz, M.Jordan, P.Abbeel. Trust Region Policy OPtimization. 32nd International Conference on Machine Learning 2015.

T.Johannink, S.Bahl, A.Nair, J.Luo, A.Kumar, M.Loskyll, J.Ojea, E.Solowjow, S.Levine. Residual Reinforcement Learning for Robot Control. Arxiv Preprint: https://arxiv.org/abs/1812.03201

B.Lutjens, M.Everett, J.How. Safe Reinforcement Learning with Model Uncertainty Estimates. International Conference on Robotics and Automation 2018.

### 6.1 Note:

Paper Implementation of RARL can be found here