Targeted Query-based Action-Space Adversarial Policies on Deep Reinforcement Learning Agents

CS637 (2021-22-1) Project Presentation

Group Members:

Samarth Varma (180655)
Milind Nakul (180420)
Nakul Singh (180455)

Introduction

- Increase in complexity of Cyber Physical Systems due to advancement in Computation.
- The Shift from Traditional Control to Deep Reinforcement Learning based methods for control of these systems.
- Deep Learning as a learning based approach is prevalent in CPS which are capable of generating immense amount of data from monitor and sensor readings.
- Deep Learning : Fault and anomaly detection, system monitoring, and supporting human-machine interaction procedures.
- Deep Reinforcement Learning : inverse design, traffic resource management, recommendation systems and image processing
- sensor readings and states of the system as RL observations

Traditional Control Methods vs Reinforcement Learning

- Obtaining a highly accurate model for real-world and complex CPS is non-trivial
- Extensive knowledge of system's behavior is required for Traditional Control Methods
- Often Challenging to scale to complicated high-dimensional systems.
- Reinforcement Learning methods do not require knowledge of the system but only access to the actual system to learn a good control policy.
- Leveraging the abundance of data to learn a control policy which can be adapted to real system.
- Robotic Manipulation and Navigation

Drawbacks of a RL-based control

- Vulnerable to a host of malicious adversarial attacks.
- Injection of perturbations of various forms through the multiple channels in which the controller interacts with the environment.
- The cost of such a successful attack on the RL controller can be extremely high (safety-critical systems).
- Understanding the various mechanisms in which an RL controller can be attacked to mitigate such scenarios.
- Study of Query-based black-box DRL-based targeted attack model that perturbs another RL agent to an adversarially-defined goal.

Problem

- Difficulty of obtaining accurate models of complex CPS for traditional control.
- Focus shift from Traditional Control Methods to Deep Reinforcement Learning.
- To securely deploy DRL, it is essential to learn the weakness from a possible malicious adversary attack on the Cyber Physical System.
- Problem Focus is on the Action-Space Actuation Attacks which perturbs the output of a controller.
- Analyzing the characteristics of the nominal policies and find the policies that are robust towards attacks.
- Efficacy of different adversarial training schemes.

Related Works

- In the field of Deep RL based control system adversarial attacks have been broadly studied under the division of black box and white box adversarial attacks
  - A black box DRL requires only the sensory data to mount a targeted attack on another RL agent.
  - A white box DRL requires full state information of the targeted RL agent before mounting an attack.
- The susceptibility of RL agents to such adversarial attacks have recently been studied recently.
- Their vulnerability has been discussed in several papers where the authors proposed a method to attack a DRL agent's state-space uniformly at each every time step.
- Several other researchers have also proposed state-space attack and defence strategies for DRL agents. While the above studies focus on white-box attacks, attempts to find optimal black-box attacks found that smooth policies are naturally more resilient.

Related Works

- Attack models based on state-space attacks perturbs or modifies the input of the RL agent such that the agent is fooled into taking wrong actions whereas action-space attacks add perturbations to π_{nom}'s output such that the resultant actions lead the agent to the adversarial goal.
- In this paper a black box model is adopted which uses action-space attacks for adding perturbations to the nominal trained policy.
- Another approach which differentiates this work from others is the use of learning-based attack.
- Optimization-based attacks often require access to internal information of π_{nom}, such as network architectures and weights, which may be non-trivial to obtain. Additionally, the knowledge encoded within π_{nom} typically does not contain accurate information with regard to other arbitrary goals; hence mounting targeted attacks is not straight-forward.
- Learning based attacks overcome these problems associated with optimization-based attacks thus providing an efficient implementation.

Deep Reinforcement Learning

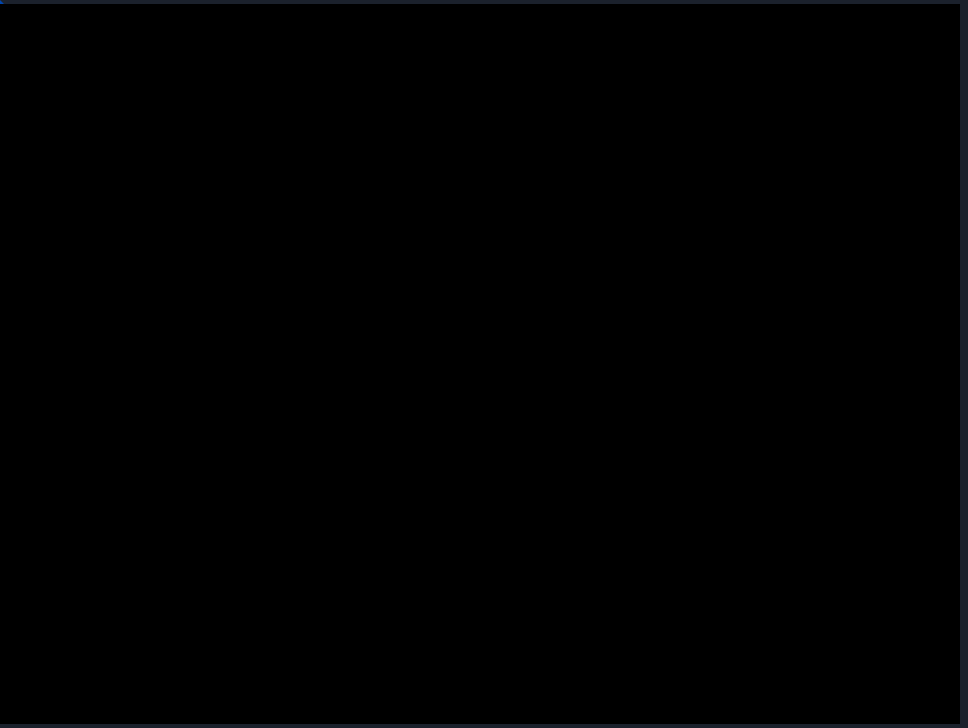
- We are provided with an environment which has its own dynamics and states. This environment of Reinforcement Learning Framework can be used to model different real world Cyber Physical Systems.
- We are provided with an agent which takes certain actions at each time step and correspondingly the environment returns the reward and corresponding observation.
- Depending on this reward and observation the agent decides to take further actions.
- Thus the agent learns to take optimal actions while interacting with the environment.
- The goal of the agent is to maximize the cumulative discounted future reward.

Policy Based Method in DRL

- We will be using model free, policy based Reinforcement Learning Framework.
- In the policy based methods, the agent directly learns some policy which maps the states to optimal actions to maximize the returns.
- This optimal policy returns a distribution over the actions via a representation of the parameterized distribution.
- To train our agent we will use the policy based method called Proximal Policy Optimization(PPO).

Description of Proximal Policy Optimization

- One of the most popular Policy Gradient Methods .
- It provides us with state of the art results for various Reinforcement Learning problems.
- We use the PPO algorithm which has been implemented in the chainerrl library.
- We use a DNN to represent the policy with some parameter.
- In this algorithm we use a ratio over the old policy and the new policy and use it to in the objective function to come up with an update.

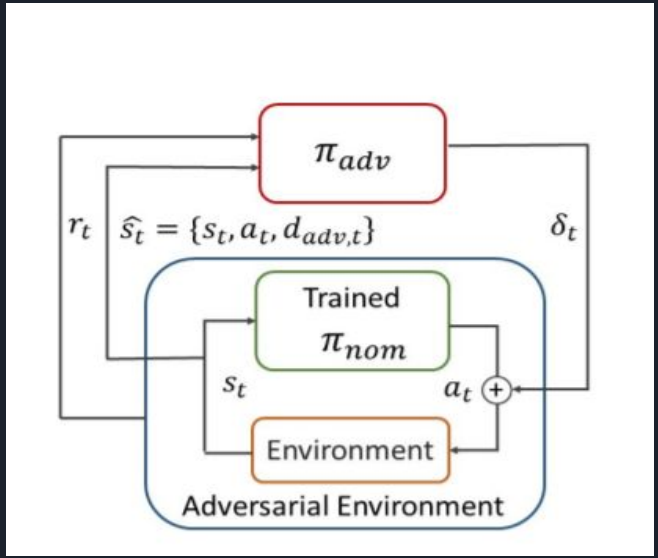


- PPO lets us train AI policies in challenging environments
- Policy Gradient methods are fundamental to recent breakthroughs in using deep neural networks for control (Video Games/ 3D locomotion)
- Getting good results is challenging as they are sensitive to the choice of step size.

Reference: https://openai.com/blog/openai-baselines-ppo/

Framework

- The RL agent which is subjected to targeted adversary attacks is called the nominal policy(π_{nom}).
- The RL agents which mounts the targeted attack on the nominal policy is called the adversarial policy(π_{adv}).
- In this implementation we do not require information about the internal architecture of π_{nom}
- We just need the readings of the sensors and actuators at the input and output channels of $\pi_{nom.}$



Objective of the Targeted Attack Model

- Our objective is to train π_{adv} to learn an optimal sequence of perturbations to add to π_{nom} actuations such that the nominal policy arrives at the adversarial goal.
- We assume that π_{nom} has been trained sufficiently and it produces action to reach the nominal goal. Hence it can be considered to be a stationary policy which does not change with time.
- We can cast the objective of the targeted attack model as another RL formulation, where the interactions of π_{nom} with the actual environment are combined into an adversarial training environment for $\pi_{adv.}$
- In this adversarial environment, the goal of π_{adv} is to maximize the reward with respect to the adversarial goal, subject to environment's state transitions and π_{nom}'s actions.

Mathematical Formulation of the objective

$$\max \boldsymbol{R}_{\text{adv}} = \sum_{t=0}^{T} \hat{R}(\hat{s}_t, \delta_t)$$

where :

$$\hat{s}_t = \{s_t, a_t, d_{adv}\},\ \delta_t \sim \pi_{adv}(\hat{s}_t)$$

$$s_t = E(s_{t-1}, a_{t-1} + \delta_{t-1}),\ a_t \sim \pi_{nom}(s_t)$$

- In the formulation above, $\hat{s}_t$ denotes the state observed by π_{adv} at time step t, which consists of concatenated vectors of the nominal state observation s_t, nominal action at sampled from π_{nom} and a distance measure to the adversary goal $d_{adv,t}$.
- δ_t denotes the perturbations sampled from π_{adv} and E(.) signifies state transitions of the environment due to the previous states, actions and perturbations.

Reward Function

$$\hat{R} = \begin{cases} \|d_{adv,t-1}\| - \|d_{adv,t}\| - I(\lambda) & \text{if } d_{adv} > 0 \\ 1 & \text{if } d_{adv} = 0 \end{cases}$$

- This is the reward function we use to train the adversarial agent.
- We use the difference between the distance to the adversarial target at successive time steps to come up with the reward function.
- We also use another penalty term, I(λ). This terms penalizes our adversarial policy if the agent reaches the nominal goal.

Our Reward Function

$$\hat{R} = \left\{ \begin{array}{ll} ||d_{adv,t-1}|| - ||d_{adv,t}|| - (||d_{nom,t-1}|| - ||d_{nom,t}||), & \text{if } d_{adv} > 0 \\ 1, & \text{if } d_{adv} = 0 \end{array} \right\}$$

- Instead of using a discrete Indicator function to denote whether the agent reaches the nominal goal we plan on using a continuous function . We use the difference between the distance to the nominal target at successive time steps to come up with the reward function.
- This helps us in providing a better estimate of whether the agent moves away from the nominal target and towards the adversarial target.

Training the Adversarial Agent

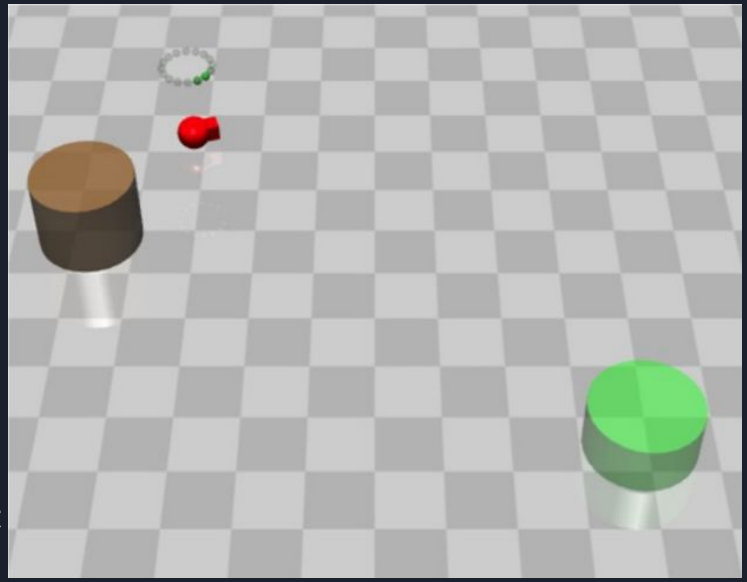
Initialize π_{adv}, trained π_{nom}, E N; **while** *steps* $< N$ **do**
  Initialize s_t;
  while $t < T$ **do**
    $a_t \sim \pi_{nom}(s_t)$;
    Construct $\hat{s}_t = (s_t, a_t, d_{adv})$ or $\hat{s}_t = (a_t, d_{adv})$;
    $\delta_t \sim \pi_{adv}\ (\hat{s}_t)$;
    $s_{t+1} = E(s_t, a_t + \delta_t)$
  if *time for update* **then**
    Update π_{adv}

Parameter	Value
Optimizer	Adam
Learning Rate	3E-4
Entropy Coefficient	0
Batch Size	1024
Update Interval	2048

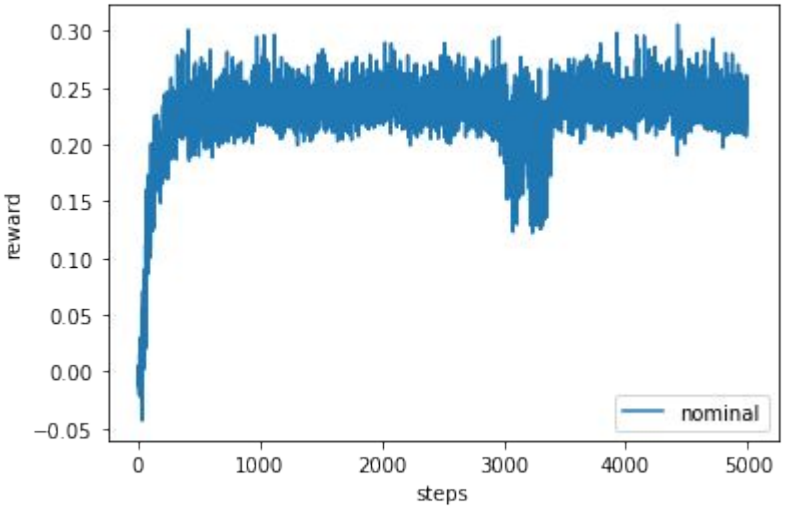
- This is the algorithm for training our adversarial agent and the values of various hyperparameters used.
- We initialize the state of the environment by resetting it.
- We select the action from the nominal policy depending on the current state.
- We construct the observation space for the adversarial agent by combining vectors of the nominal state observation s_t, nominal action at sampled from π_{nom} and a distance measure to the adversary goal $d_{adv,t}$.
- Then we obtain the perturbation provided by the current adversarial agent.
- Then based on the action and current state the environment reaches a new state .
- We update the adversarial policy depending on the update frequency.

Environment Description

- Two goals corresponding to Nominal and Adversarial Goals
- Green area, Brown area and Red object represents the nominal goal, adversarial goal and the agent respectively.
- Point Goal: RL agent is represented as a simple robot that can actuate forward/backward and rotate clockwise/anti-clockwise.
- Car Goal: RL agent is represented by a robot with two independent wheels and a free-rolling wheel with same translation and rotation ability.
- Sensor readings and LiDAR information enables the agent to observe its surroundings that represents the relative distance to its goal.
- Agent outputs an action bounded between [-1,1] for each actuation dimension.



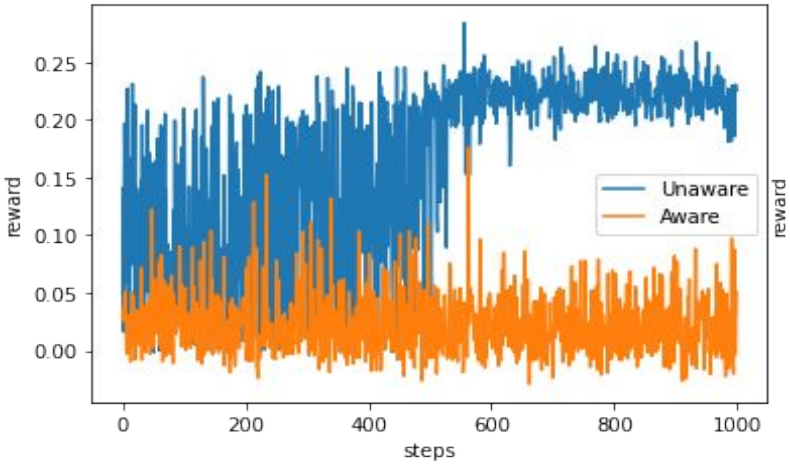
Results and Discussion- I PointGoal Environment



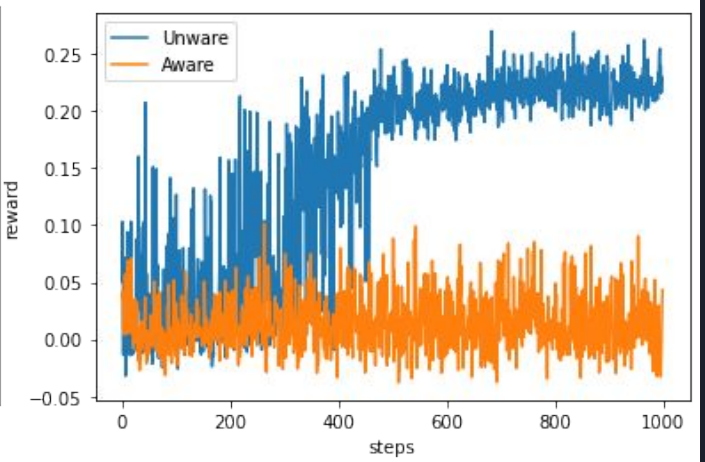
Nominal Agent trained using the PPO algorithm for PointGoal Environment.

Nominal Agent gets the reward directly from the environment

Adversarial Agent- I for PointGoal Environment

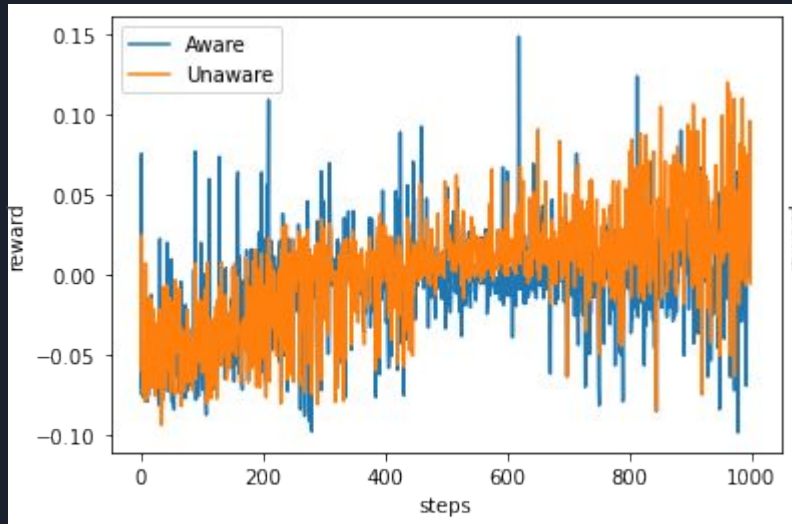


Old Reward Function

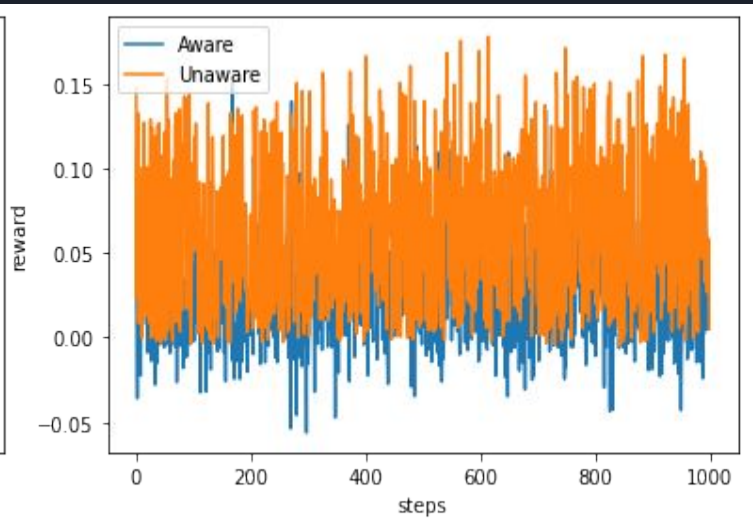


New Reward Function

Robust Agent- I for PointGoal Environment

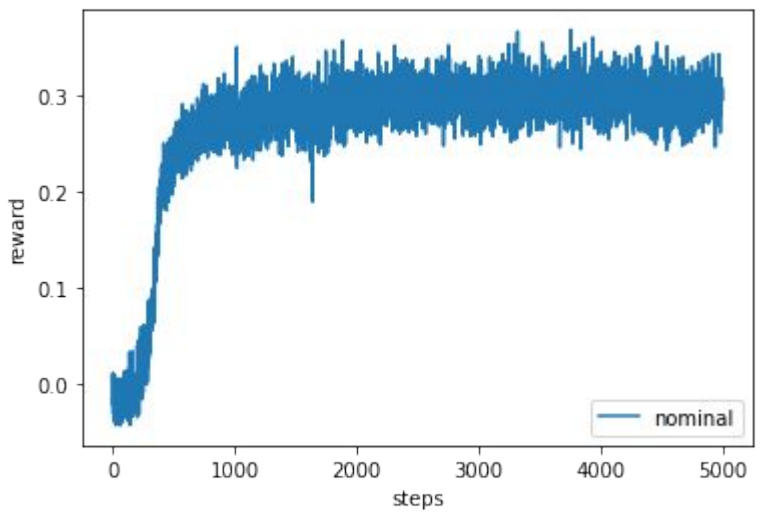


Robust Agent with new Reward Function



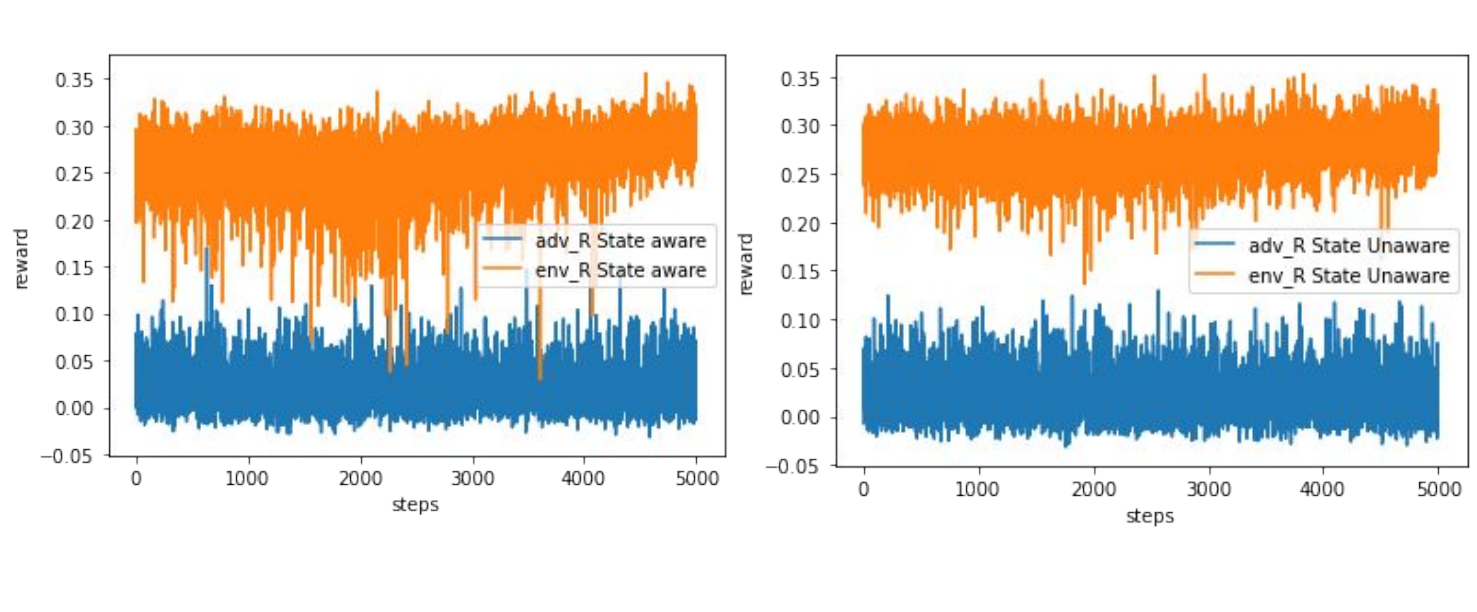
Robust Agent with the old Reward Function

Results and Discussion-Ⅱ for CarGoal Environment

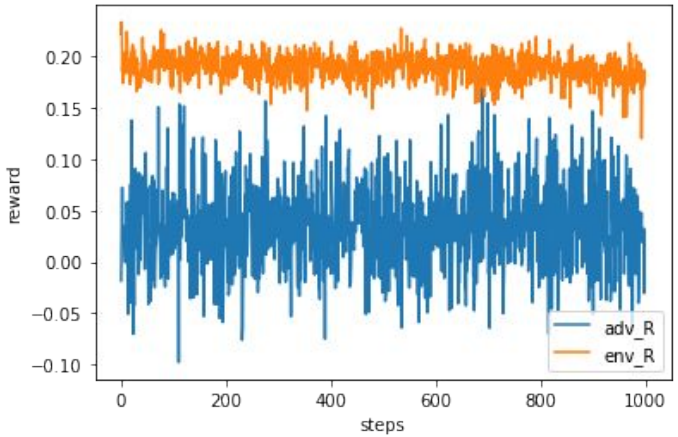


- Apart from PointGoal environment nominal agent was trained using PPO algorithm on CarGoal environment.
- The number of iteration to reach the optimal policy for CarGoal environment is slightly larger due to higher complexity.

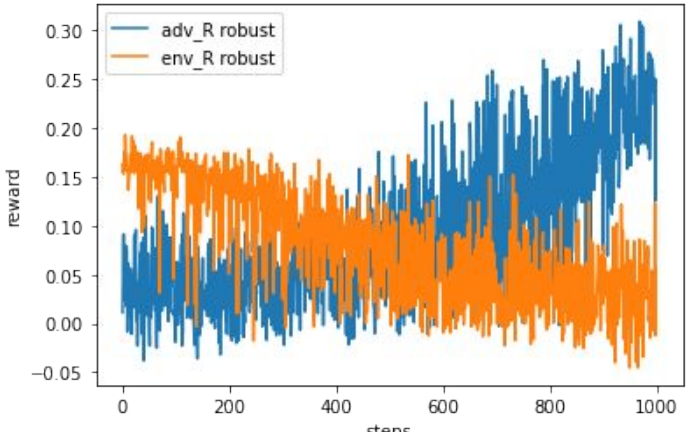
Adversarial Agent-Ⅱ for CarGoal Environment



Robust Agent-Ⅱ for CarGoal Environment

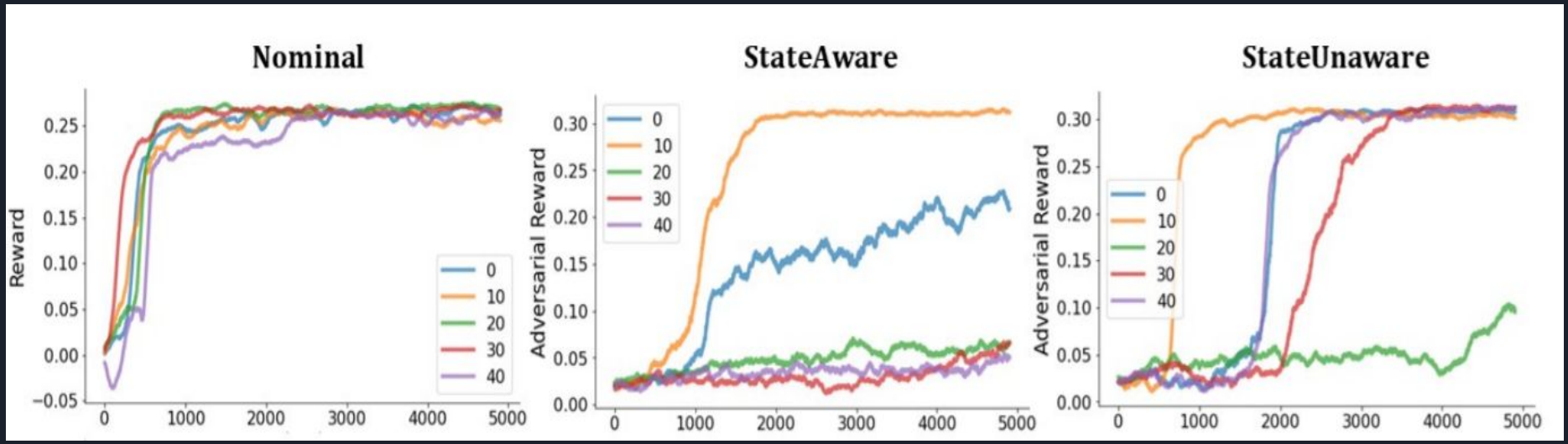


Robust Agent with the old Reward Function

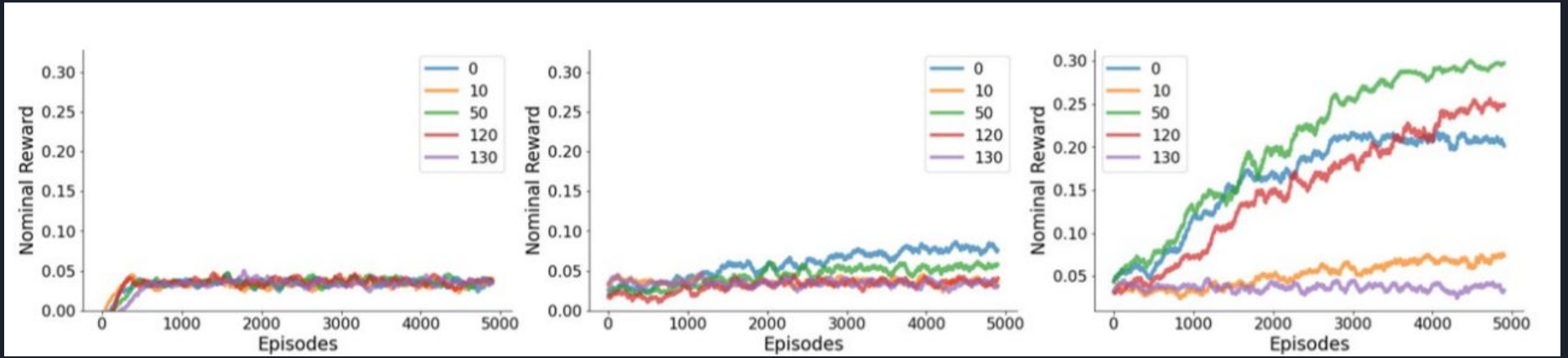


Robust Agent with the new Reward Function

Results from Paper for the Adversarial Agent



Results from Paper for Robust Agent



Untrained Nominal and Adversarial policy

Untrained Nominal But Trained Adversarial Policy

Trained Nominal and trained Adversarial Policy

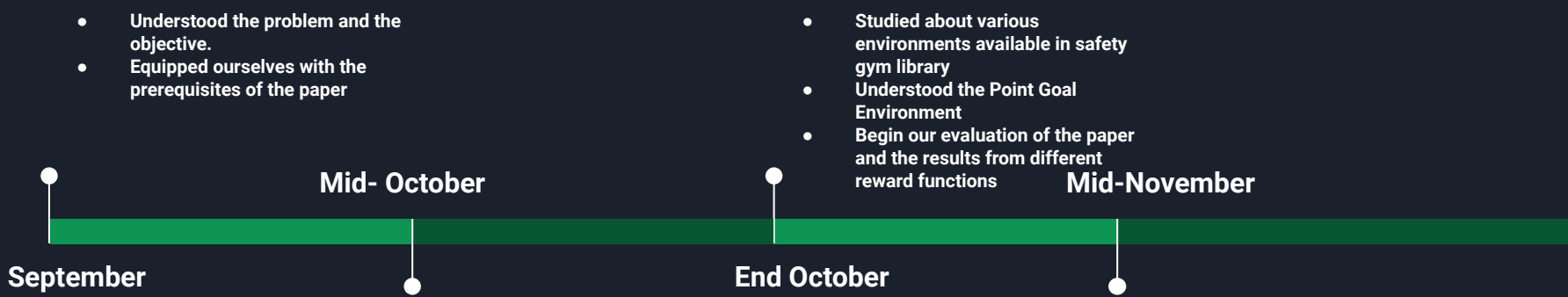
Additional Results

- Comparing π_{nom} without adversarial training and π_{nom} after adversarial training that is subjected to targeted attacks, we observe that the reward is significantly reduced in case of π_{nom} without adversarial training.
- Nevertheless, such adversarial training schemes also comes with a cost to the general performance of $\pi_{nom.}$
- This occurs in cases where there is no adversarial attack on the agent.
- Trade-off between a policy's performance and ability to generalize to more scenarios.

Questions and Conclusions

- During the presentation it was asked that intuitively state Aware control system should provide better training statistics compared to the state Unaware case, then why is a counter-intuitive result observed?
- Based on research articles and discussions one possible conclusion that we could draw was the curse of dimensionality.
- Ultimately Deep reinforcement learning is a machine learning problem, on adding State information to the given data set we are inadvertently increasing it’s dimension whereas the size of the data set remains same.
- This redundant state information therefore acts as a detrimental factor slowing down the learning process.

Timeline



- Understood the problem and the objective.
- Equipped ourselves with the prerequisites of the paper

- Studied about various environments available in safety gym library
- Understood the Point Goal Environment
- Begin our evaluation of the paper and the results from different reward functions

Contributions

- Nakul - 33.33%
- Samarth - 33.33%
- Milind - 33.33%

THANK YOU