Project#4: Safe Exploration in Continuous Action Spaces

Rajarshi Dutta[1], **Udvas Basak**[2], **Divyani Gaur**[3]
[1]200762, [2]201056, [3]200348
[1]MSE, [2]AE, [3]CHE
{rajarshi20, udvasb20, divyanig20}@iitk.ac.in

Abstract

The given report talks about safe exploration in continuous dynamics which primarily focuses on avoiding constraints violation. The problem statement also aims to prove that the smooth dynamics of the system can facilitate the RL algorithms to avoid constraints violation during training. possible solution is adding a safety layer over the DDPG layer that essentially solves an action correction formula per state. Also, the paper deliberately proves a closed-form solution due to the linearized approximation of the safety layer which guarantees convergence. This particular hybrid approach has been tested on two Mujoco game environments, primarily the 3-D Ball and Spaceship collider environments. All the codes are provided in https://github.com/Rajarshi1001/CS780_Project

1 Introduction

Reinforcement learning algorithms have been applied to various games, whether 2D or 3D. These algorithms also show promising applications in industries which also require precise constraint-free operations. Unless, the algorithms ensure safety measures thoroughly starting from the first moment of deployment, those algorithms would be deemed incompatible. Therefore, the paper (Dalal et al., 2018) majorly focuses towards zero-constraints violations during the entire learning process. It has been observed that this task is particularly easy for the case of discrete action spaces by essentially pretraining a constraint violation classifier on offline data which eliminates undesired actions but this paper focuses on continuous action spaces.

2 Problem Defintion

In most real world problems, the desired mechanism is to run a high efficiency risky operation until safety flags arise and these might trigger a preemption mechanism and redirect the given system to a conservative safe operation scheme. The primary focus is on first order or second order systems while handling one constraint at a given time step. The author also proposes about the case of multiple constraints where a joint model can be learned, treating all the constraints as a single combined one.

The given problem can be modelled as **CMDP** (Constrained Markov Decision Process) which consists immediate-constraints functions and per state observations in addition to the elements of an MDP. Considering a deterministic policy $\mu : S \to A$, the last relation (3) holds.

$$C = \{c_i : \mathcal{S} \times \mathcal{A} \to R | i \in [K]\} \tag{1}$$

$$\bar{\mathcal{C}} = \{\bar{c}_i : \mathcal{S} \to R | i \in [K]\} \tag{2}$$

$$c_i(s, a) \stackrel{\triangle}{=} c(\bar{s}') \tag{3}$$

The given problem boils down to solving a state-wise constrained policy optimization problem (4) where at each state, all safety signals $\overline{c_i}(.)$ are upper bounded by corresponding constants $C_i \in R$. For the types of physical systems we consider it is indeed plausible that safety constraints can be ensured by adjusting the action in a single (or few) time step(s).

$$\max_{\theta} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mu_\theta(s_t))\right] \tag{4}$$

$$\text{s.t.} \quad \bar{c}_i(s_t) \leq C_i, \quad \forall i \in [K] \tag{5}$$

3 Related Work

The entire work focuses on control problems with policy optimization. One of such examples were (Achiam et al., 2017) where similar constraint policy optimization was solved using a modified trust-region policy gradient where the policy was updated to safe set on each iteration during the learning process. A second approach (Berkenkamp et al., 2017) essentially devised an appropriate Lyapunov

function for policy attraction regions under certain control conditions. Although safe exploration was guaranteed under the given conditions, the algorithm demanded knowledge of the specific system. Another approach utilized a Quadratic Programming solver (first mentioned in (Amos and Kolter, 2021)) which essentially solved an optimization problem on policy level or state-wise safety basis. A complete manual approach (Pham et al., 2018) was mentioned in where algorithm required exert knowledge to hand-design physical constraints in a robotic arm actuation.

4 Methodology/Algorithm Description

It is impossible for the model to start correcting actions to avoid constraints in the initial phases of its learning. Shaping the reward as well to heavily penalize the model on violation of constraints also doesn't seem to work since it required enough training for the negative effect of violation to be incorporated in the DP scheme. The author suggests to use single step dynamics in contrast to a known behaviour policy to overcome this issue.

The approach lies in a one-time initial pretraining of a lightweight neural network that essentially predicts the changes in safety signal over periods of time. The main simplicity is derived from its formulation;, a first order approximation of the changes in the safety signal with respect to actions using state features calculated from a state fed neural network.

$$c_i(s,a) \triangleq c(\bar{s}') \approx c_i(s) + g(s;w_i)^\top a \quad (6)$$

The $g(s;w_i)$ is trained using the above neural network. The training scheme proceeds via generation of an policy-oblivious set $D = (s_j, a_j, \bar{s}'_j)$ created by randomly initializing the agent in uniform random locations in the environment, letting it perform random actions for multiple episodes. The episodes are essentially terminated on reaching time limit or constraint violations.

The linear safety layer is placed on top of the original policy network [1]. The safety layer essentially performs action corrections after every policy query; it solves an quadratic optimization problem to calculate the minimal change in actions such that the safety conditions are always satisfied in every part of the learning process.

$$a^* = \arg\min_\theta \frac{1}{2}||a - \mu_\theta(s)||^2 \quad (7)$$

$$\text{s.t.} \quad \bar{c}_i(s) + g(s;w_i)^\top a \le C_i, \quad \forall i \in [K]$$

The linearity of the approximation also ensures a closed form solution 8 and an easy implementation of the safety layer.

$$\lambda_i^* = \left[\frac{g(s;w_i)^\top \mu_\theta(s) + \bar{c}_i(s) - C_i}{g(s;w_i)^\top g(s;w_i)^\top}\right]^+ \quad (8)$$

$$a^* = \mu_\theta(s) - \lambda_{i^*}^* g(s;w_{i^*}) \quad (9)$$

5 Experiments, Results and Discussion

The given implementation was tested on two environments, namely Ball(n-D) and spaceship.

1. For the Ball setting(1), a cube is constructed on d-dimensions, represented as $B^d_{[a,b]} = \{x | a \ge x_i \le b, i = 1, \ldots, d\}$. The goal is to bring the ball (green) to the target (purple) by setting the velocity of the ball at every 4^{th} time step. Episode runs for 30s and the target appears at a uniformly sampled different location ever 2s. The feasible region kept for the ball $B^d_{[0,1]}$ and target $B_{[0.2,0.8]^d}$. The system dynamics is governed by Newton's laws with considerable amount of damping. State of the ball is represented as $s = (x_B, v_B, x_T + \epsilon_d)$ where $\epsilon_d \sim \mathcal{N}(0, 0.05 \cdot I_d)$, action as $a = v_B$. The reward function is chosen as $R(s,a) = \max(\left[1 - 10 \cdot \|x_B - x_T\|_2^2, 0\right])$ and discount factor γ is chosen as 0.99. For the action correction to initiate, C_i's are set to effectively constrain the ball's feasible region to $B^d_{[0.1,0.9]}$, represented by the red region in the figure ((1)). Action correction is applied by the linear safety layer as the ball steps out of this region.

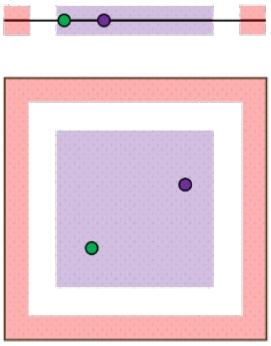


Figure 1: Ball Environment

2. The second environment ((2)) represents a spaceship bounded by a closed region. There

[1] **DDPG** network which directly outputs actions)

are two variants of the environment in terms of walls, first having two infinite parallel walls and second having walls in form of a diamonds. The termination occurs when target is reached, spaceship's bow touches the wall or when time limit is exceeded (15s for corridor, 45s for Arena). The state space consists of spaceship's location and velocity, the action space $a \in [-1, 1]^2$ consists of two thrust engines in front, back, right and left directions. The reward structure is sparse and set to 1000 points on reaching the target, 0 elsewhere with γ set to 0.99. The system dynamics is second-order with position being controlled by force. Similarly, there is a safety region placed at a distance of 0.05 from every wall which executes the action correction during learning.

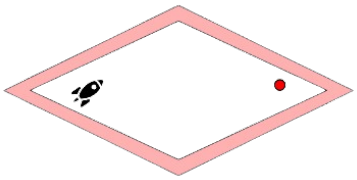


Figure 2: Spaceship Environment

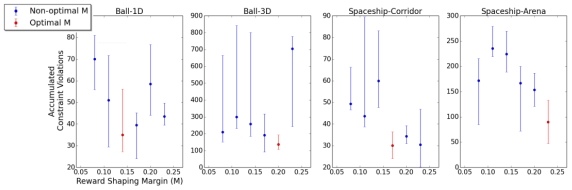


Figure 3: Constraints violations for different M values

The general techniques for this problem would be to shape the reward near the boundaries such that the agent is penalized heavily on crossing those regions. The paper has proposed a highly negative reward (-1 for Ball-nD environment and -1000 for spaceship) and in a M fraction. The algorithms are executed (DDPG) as the base model for producing actions, with varying $M \in [0.08; 0.11; 0.14; 0.17; 0.2; 0.23]$ and it was observed (3) that there were no clear structural patterns in the optimal M obtained from the plots.

Using these optimal M for reward shaping, the performance of DDPG, DDPG + reward shaping(using optimal M), and DDPG + safety layer, is compared in Fig 4. In Ball-nD environments, DDPG + reward shaping performs better than DDPG with respect to both convergence and lesser

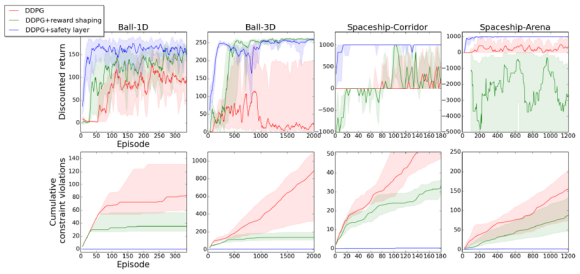


Figure 4: Comparison of performance of DDPG, DDPG + reward shaping, and DDPG + safety layer

constraint violations. DDPG + safety layer outperforms both, with 0 constraint violations. DDPG cannot perform well enough in the Spaceship Environment, with no convergence. Reward shaping has a negative impact. DDPG + safety layer proved the best, with a convergence which was very early, and 0-1 constraint violations.

6 Reproducibility

The algorithms presented in the paper are reproduced, with inspiration from (Lukas Koller, n.d.) and (Agrawal, n.d.).

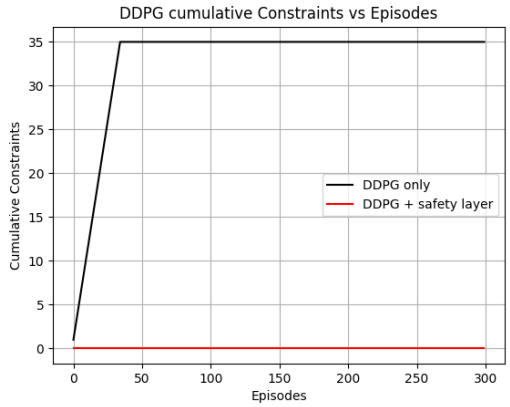
The plots(5) below illustrates the episodic reward and cumulative constraints violations observed while using a DDPG (Deep Deterministic Policy Gradient) on the Ball and spaceship environments without using any safety layer. The figure shows that the agent is able to attain convergence but at the cost of violating the constraints which defeats the entire purpose.

The next set of figure (6) illustrates the same for the spaceship environment. The rewards plots are extremely noise for the spaceship environment. There are some minor issues regarding the rewards starting from a very large negative value and converging towards 0 and the agent doesn't seem to violate all the constraints with the addition of the safety layer as seem in the Cumulative constraints violation plot (6b). These issues needs to be addressed in the future.

7 Future Directions

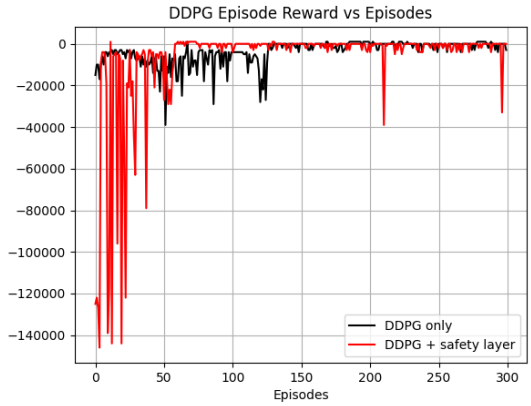
1. Our first areas of improvement would be to look and develop the environments, especially where the ideas fail. These might arise due to two major issues, firstly either through the neural network not being able to learn the $g(s; w_i)$ function properly and the other through the introduction of multiple

(a) Rewards vs Episodes for Ball environment

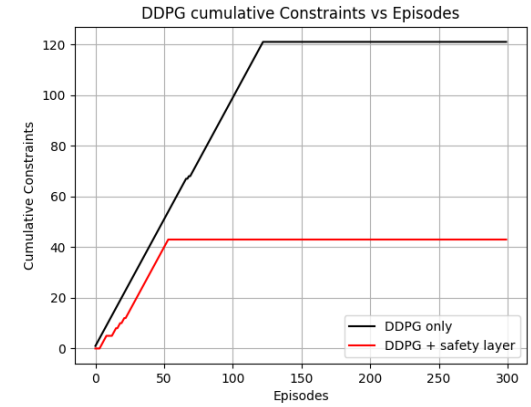


(b) Cumulative Constraints vs Episodes for Ball environment

Figure 5: Comparison between two approaches for Ball environment



(a) Rewards vs Episodes for spaceship environment



(b) Cumulative Constraints vs Episodes for spaceship environment

Figure 6: Comparison between two approaches for spaceship environment

constraints being considered at the same time step.

2. Another possible work could be trying to actively trying this proposed solution in other real and physical world environments (like data center cooling, or robotic actuation, bioreactors) where safety critical conditions are deemed mandatory.

3. The proposed timeline is shown in Fig. 7

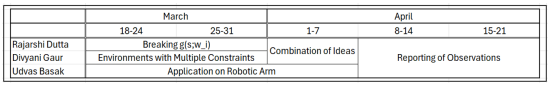


Figure 7: Proposed Timeline

8 Individual Contribution

Member	Contribution
Rajarshi Dutta	Finding relevant repositories, understanding the code and its implementation.
Udvas Basak	Mathematical formulation and proofs
Divyani Gaur	Looking for related research papers for other physical environments, presentation slides

Table 1: Individual Contributions

9 Conclusion

The paper provides a foolproof way for ensuring safe explorations in continuous action spaces. The environments defined are unique, and provide good benchmarks for testing and comparison. The implementation of the code is also carried out, and the results are presented.

References

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

Amey Agrawal. n.d. safe-explorer. https://github.com/AgrawalAmey/safe-explorer.

Brandon Amos and J. Zico Kolter. 2021. Optnet: Differentiable optimization as a layer in neural networks.

Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. 2017. Safe model-based reinforcement learning with stability guarantees.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerík, Todd Hester, Cosmin Paduraru, and Yuval Tassa. 2018. Safe exploration in continuous action spaces. *CoRR*, abs/1801.08757.

Lukas Koller. n.d. safe-explorer. https://github.com/kollerlukas/safe-explorer.

Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. 2018. Optlayer - practical constrained optimization for deep reinforcement learning in the real world.

10 Appendix

Hyperparameters	**Values**
DDPG Actor Layers	[100, 100]
DDPG Critic Layers	[500, 500]
Actor Learning Rate	0.0001
Critic Learning Rate	0.001
Safety Layer Neurons	100
Agent Slack for Ball-nD environment	0.1
Agent Slack for Spaceship Environment	0.05
Discount Factor	0.99
Memory Buffer Size	1000000
Actor Weight Decay	0
Critic Weight Decay	0.01

Table 2: Hyperparameters for DDPG + safety layer