EFFECT OF JENSEN-SHANNON DIVERGENCE IN SAFE MULTI-AGENT RL

Rushikesh Zawar **Prabhdeep Singh Sethi** **Roshan Roy**

School of Computer Science
Carnegie Mellon University
{rzawar,prabdhes,roshanr}@andrew.cmu.edu

ABSTRACT

Reinforcement Learning (RL) has achieved significant milestones, however its safety remains a concern for real-world applications. Safe RL solutions focus on maximizing environment rewards while minimizing cost. In this work, we extend the Multi-Agent Constrained Policy Optimisation (MACPO) approach that maintains policy consistency using Kullback-Leibler (KL) divergence. We find that Jensen-Shannon (JS) Divergence, a symmetric measure, serves as a better alternative to KL divergence; its symmetric nature is more forgiving of extreme differences in policies. Our results demonstrate that JS divergence improves rewards and reduces costs, enhancing safety and performance in multi-agent systems.

1 INTRODUCTION

In Reinforcement Learning, the primary goal for agents involves maximizing a predetermined reward signal. However, if this reward signal lacks precision or clarity in its design, it can lead the agent to learn unintended actions, potentially resulting in undesirable or even harmful behavior. The challenge lies in the intricate nature of crafting these reward functions, which proves to be fundamentally difficult Gu et al. (2022). This difficulty in designing optimal reward functions serves as a pivotal driving force behind the adoption of constraints within RL frameworks when deploying it in real-world scenarios as you need safety within the environment as well as safety and coordination amongst agent in a multi-agent setup.

Constrained Policy Optimization (CPO) by Achiam et al. (2017) is a cornerstone work in Safe Reinforcement Learning. MACPO by Gu et al. (2021) extends it to the more complex Multi-Agent settings. It has a Local Policy Search for Constrained RL that builds upon TRPO by Schulman et al. (2015) and enforces safety in every policy update. It operates within the framework of Markov Decision Processes (MDPs) to ensure policy consistency via average Kullback-Leibler (KL) Shlens (2014). However, the rigidity of KL divergence and the need for fine-tuning its maximum value throughout training have limitations.

Since in MACPO each new policy has to be close to the old one in terms of average KL and drawing inspiration from the success of Wasserstein distance in Generative Adversarial Networks (GANs) by Arjovsky et al. (2017) and Weng (2019), known for its superior performance in capturing differences between low-dimensional manifolds, we propose a novel approach of incorporating JS to better capture policy divergence. Given the intractability of Wasserstein distance and its issues with continuous settings, we find JS to be a middle ground.

2 METHODOLOGY

The JS Divergence between two probability distributions P and Q is defined as the average of the KL Divergence of P from the average distribution M, and the KL Divergence of Q from M. If we denote the KL Divergence as D_{KL}, the JS Divergence D_{JS} can be mathematically represented as:

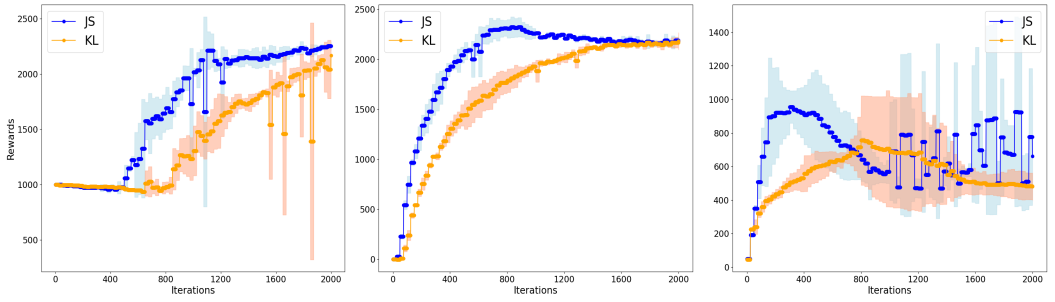
$$D_{KL}(P||Q) = \sum_x P(x) ln \frac{P(x)}{Q(x)} \tag{1}$$

$$D_{JS}(P,Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \quad where \quad M = \frac{1}{2}(P+Q) \tag{2}$$

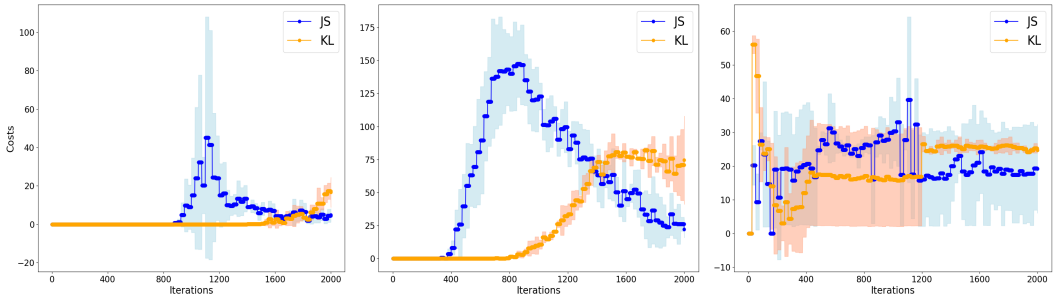
KL divergence (see 1) gives higher penalty when the probability of an event is high in one distribution and low in the other. The sensitivity of KL divergence to regions of near-zero probability suppresses exploration compared to JS. JS divergence being symmetric in nature (see 2) makes it more forgiving to extreme differences in probability distributions and also makes it smooth.(Ghasemipour et al. (2020), Miret et al. (2020)). This allows room for exploration for the new policy to be considered which helps multi-agent coordination as it requires broader exploration of policy spaces to discover complex dependencies. Since, JS Divergence is smooth during training it stabilizes the training better than KL (Nielsen (2020)).

Our work thus explores the potential of JS Divergence in enhancing both the performance (rewards) and safety (costs) of multi-agent systems.

3 Experiments



(a) Rewards: MA - 2x4 Ant (b) Rewards: MA - 2x3 Cheetah (c) Rewards: MA - 3x1 Hopper



(d) Cost: MA - 2x4 Ant (e) Cost: MA - 2x3 Cheetah (f) Cost: MA - 3x1 Hopper

Figure 1: The graphs in the top line compares the rewards over epochs with the Baseline of MACPO and the bottom line shows the cost over epochs, where MA stands for Multi-Agent.

We have tested our approach in 3 different environments of Safety2x4AntVelocity-v0, Safety2x3HalfCheetahVelocity-v0 and Safety3x1HopperVelocity-v0 from Safety-gym by Ray et al. (2019). We can see in 1 our approach (JS) gives better or comparable rewards to MACPO in 3 different environments but with a significantly lower cost which shows that our methodology performs well on task objective while being safer in a multi-agent setup.

4 Conclusion

In conclusion, our work presents an alternative approach to KL divergence to enhance safety and performance in multi-agent systems by using Jensen-Shannon Divergence in MACPO. Our findings indicate that this symmetric divergence measure leads to improved rewards and reduced costs. Thus we believe that this work can be further explored in similar paradigms.

URM STATEMENT

We acknowledge that all the authors of this work meet the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pp. 1259–1277. PMLR, 2020.

Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, and Yaodong Yang. Multi-agent constrained policy optimisation. *arXiv preprint arXiv:2110.02793*, 2021.

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

Santiago Miret, Somdeb Majumdar, and Carroll Wainwright. Safety aware reinforcement learning (sarl). *arXiv preprint arXiv:2010.02846*, 2020.

Frank Nielsen. On a generalization of the jensen–shannon divergence and the jensen–shannon centroid. *Entropy*, 22(2):221, 2020.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

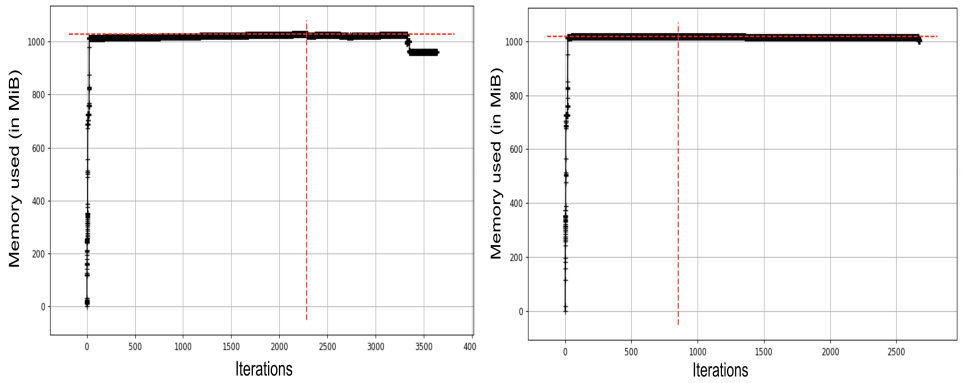
Jonathon Shlens. Notes on kullback-leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*, 2014.

Lilian Weng. From gan to wgan. *arXiv preprint arXiv:1904.08994*, 2019.

A APPENDIX

A.1 COMPUTE COMPLEXITY

This section demonstrates the computational complexity associated with the JS and KL Divergence methodologies. We present an analysis of memory utilization, as monitored via memory-profiler[1], accompanied by the average computational time incurred per iteration by each method. As seen in figure 2, we can see that both the JS and KL Divergence methods use around 1000 MiB memory for 1 environment run.



(a) Memory Profile KL (b) Memory Profile JS

Figure 2: The plot shows the amount of memory usage across the iterations. The red line denotes the peak memory usage.

Following are the details of the time required for both the methodologies:
Macpo:
Mean time of 1 iteration: 12.491843749773922
Standard Deviation of time: 0.6253396397894774

Macpo JS:
Mean time of 1 iteration: 44.2813022100951
Standard Deviation of time intervals: 7.711076688983893

We can see JS Divergence method requires a higher time during the training phase, which can be attributed to its additional calculations as seen in 1 when compared with 2. Despite this increased time the JS based method leads to improved performance in terms of rewards and cost.

[1]Memory Profiler, Python Package Index - PyPI. Available at: https://pypi.org/project/memory-profiler/