A Framework for Formal Verification to Correct Actions in Reinforcement Learning: A Proposal

Ethan Hobbs Vikas Nataraja

April 11, 2020

1 Project Overview

Reinforcement Learning has risen to prominence in recent years especially in the area of robotics and autonomous vehicles. However, much like deep learning models, reinforcement learning algorithms can often be a blackbox in terms of the policy the agent learns in a given situation leading to an inability to generalize to new environments and constraints. We propose to use formal verification methods in conjunction with an agent's actions to serve as a check on the policies the agent learns. Our implementation will synthesize a deterministic approximation of the agent's policy which is inductive invariant to "shield" the neural policy from making an invalid action which violates the inductive invariant. Using this framework, the agent can learn to abandon an incorrect action in favor of taking a sub-optimal but valid action. The inductive invariant will act in tandem with the neural policy with minimal overhead. We expand on previous work done in the field of inductive synthesis frameworks applied to reinforcement learning seen in [1] specifically focusing on the verification portions of their study. Our goal will be to prove that the verification algorithm in [1] will always produce a safe state. Currently in the algorithm, there is a condition that if there are not any safe states that the state space will be reduced by half, and the algorithm will repeat. This method seems unlikely to work in all cases so we want to assess its validity. Our goal will be to try to prove this algorithm will always function as intended and if it cannot be proven, investigate other conditions. One possible technique we would like to explore is *backtracking* which is when the agent returns to a previously encountered safe state when the verification suggests the next state is unsafe or invalid. To go along with this, we would like to explore expanding our state space instead of limiting ourselves to a reduced space.

2 Proposed Approach

Our approach contains two different components: an evaluation phase and a solution phase. First we will be evaluating the CEGIS (Counter Example-Guided Inductive Synthesis) verification algorithm to make sure that the agent always finds a solution. If that fails, we will then create new conditions and retry the same proof process on them. We propose to use back propagation through the state space and choosing a random previous safe state. The goal with these new false condition solutions is to prevent edge cases from causing the algorithm to not be able to complete. We believe they will solve the problem (assuming the original algorithm cannot be proven) because we are moving in the safe state space which will eliminate the possibility of the algorithm from getting stuck in an isolated region of the state space. We will be basing our work heavily on [1] which has extensive information about their algorithm in the supplementary information section. Our analysis will be a verification of the verification algorithm presented in that paper.

3 Evaluation

We will be developing an evaluation test along the lines of [1] which used scenarios from OpenAI gym including inverted pendulum and cart pole as examples that their method worked. We will train our agent on either the same or similar scenarios. If we do not produce a new algorithm, we want to present the proof and proof strategy through an example that will be more easily understood. We see two possible options for evaluation based on the conditional nature of our project. The first is a successful proof of the verification while the second will be a proof of any other method or an explanation of why the proof cannot be constructed. In either case, we also expect to have a basic implementation of a reinforcement learning system with the shielding described in [1].

References

[1] He Zhu, Zikang Xiong, Stephen Magill, and Suresh Jagannathan. An inductive synthesis framework for verifiable reinforcement learning. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 686–701, New York, NY, USA, 2019. Association for Computing Machinery.