Structured Rewarding using STL

USC Viterbi School of Engineering
Department of Computer Science

Anand Balakrishnan
anandbal@usc.edu

Jyotirmoy Deshmukh
jdeshmuk@usc.edu

Introduction

- Reinforcement Learning incorporates a class of algorithms that involves training an *agent* to learn a controller (or *policy*) by iteratively acting on and observing an environment.
- Each action performed by the agent on the environment is augmented with a *reward* associated with the transition in states given the action. Thus, the agent attempts to maximize the total expected reward.
- The reward is computed from a reward function, and a well designed reward function is *critical* for training an RL agent that completes a given task efficiently. The problem of designing a good reward function is known as **reward shaping**.

Promise of (Deep) Reinforcement Learning

- Deep Reinforcement Learning involves use general function approximation techniques (specifically Deep Neural Networks) to learn policies.
- The use of DNN's for RL enables the learning of policies by observing raw data, for example, raw pixel values for video games and joint/actuator angles in robots.
- Deep RL is generally **model-free**, that is, it only assumes that the environment *can be* modeled as a Markov Decision Process (MDP) and does not assume any knowledge about the internals of the MDP.

Challenges with Deep RL

- Poorly designed reward functions can lead to the synthesis of a policy that performs unsafe actions, even if it learns to maximize the rewards.
- In the case of safety-critical systems (most Cyber-Physical Systems), this can be the difference between life and death.

Formal Methods

- Meanwhile, research on safety and verification of cyber-physical systems (CPS) has extensively used Temporal Logics to define safety specifications on the system.
- The quantitative semantics, like that for Signal Temporal Logic (STL), allow us to detect how robustly a signal satisfies a given property, that is, we can calculate how robust a signal is to perturbation against a given STL formula [1].
- Works like [2] and [3] propose the usage of Temporal Logics to design robust reinforcement learning controllers.

Partial Signal Robustness

→ We propose a method to define *locally shaped reward functions* from STL specications defined on the trajectory of the system.

→ The proposed technique provides a framework by which reward functions can be specified in an expressive manner using STL formulas.

→ This methods does not depend on the design of the RL algorithm, and can be generalized out-of-the-box.

Definition (Partial Signal)**.** Given a trajectory of a system $\mathbf{x}$, a partial signal, $\mathbf{x}[i : j]$, is defined by the slice of the trajectory from the i^{th} sample to the j^{th} sample. That is,

$$\mathbf{x}[i : j] = (s_i, \ldots, s_j)$$

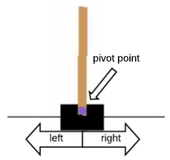
Definition (Robust Satisfaction Function)**.** The robust satisfaction function is a function $\tilde{\rho}$ that maps a STL property φ, a partial signal $\mathbf{x}[\ldots]$, and a sample step $n \in \mathbb{N}$ to a real number in $R \subseteq \mathbb{R}$.

Thus, the reward for state s_i can be written as

$$r_i = \tilde{\rho}(\varphi, \mathbf{x}[i : \tau], i) \tag{1}$$

where, τ is the length of the partial signal and φ is the STL specification defined on the system.

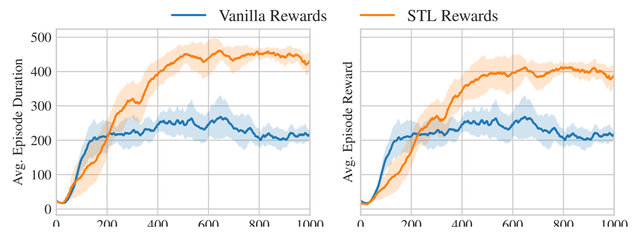
Experiment



$$\varphi = \mathbf{G}\Big(\mathbf{F}\left(|\dot{x}| < 0.01\right) \wedge \left(|\theta| < 2^{\circ}\right) \wedge \left(|x| < 0.5\right)\Big)$$

We trained a Double Deep Q-Network (DDQN) agent using the proposed framework, using the normalized **filtering semantics** [4] for STL. The DDQN policy observes the displacement of the cart from the center, the angle of the pole, and their respective velocities, and outputs a push to the left or right (bang-bang controller).

Results



- The policy trained using the proposed method keeps the cart upright for a longer duration and accumulates a larger total episodic reward than the vanilla agent.
- Thus, the traditional reward function used to train policies for the Cart-Pole problem tend make the final policy to perform worse than the new reward function.

Future Work

- While this approach works well for many cases, it doesn't guarantee convergence too a good policy when the system if very complex, like in the case of quadrotor dynamics, for example.
- The performance does not necessarily scale with the size of the STL formula, due to the discrete nature of the min and max operators in the quantitative semantics of STL. This can trivially be addressed using smoothened versions of the operators, or a more probabilistic approach can be used to combine the sub-formulas.

References

[1] A. Donzé and O. Maler, "Robust Satisfaction of Temporal Logic over Real-Valued Signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*, Springer, 2010, pp. 92–106.

[2] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-Learning for Robust Satisfaction of Signal Temporal Logic Specifications," Sep. 2016. arXiv: 1609.07409 [cs].

[3] X. Li, C. Vasile, and C. Belta, "Reinforcement Learning with Temporal Logic Rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3834–3839. DOI: 10.1109/IROS.2017.8206234.

[4] A. Rodionova, E. Bartocci, D. Nickovic, and R. Grosu, "Temporal Logic as Filtering," *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control - HSCC '16*, pp. 11–20, 2016. DOI: 10.1145/2883817.2883839. arXiv: 1510.08079.