Formal Verification Methods for Solving Spatial Games

Authors:

Boaz Gurevich & Erel Dekel

Supervisors:

Hillel Kogler & Avraham Raviv

Date:

Thursday, October 10th 2024

Abstract

Reinforcement Learning (RL) has emerged as a cornerstone of modern technology, driving advancements in areas such as autonomous systems, robotics, and adaptive control.
However, a critical challenge in deploying RL algorithms lies in their tendency to suffer from convergence issues, particularly in complex, high-dimensional environments where instability can lead to suboptimal or unsafe policies.
This project aims to explore the impact of formal verification techniques on RL, specifically focusing on how these methods can address the convergence challenges that plague RL systems.
We will empirically investigate this by applying formal verification to a model-free RL algorithm designed to solve the Sokoban puzzle game, a notoriously difficult problem requiring complex planning and decision-making.

Contents

1. Background

In this chapter, we introduce the fundamental concepts that critical for the reader to understand. We will provide deep understanding of Neural Networks (NNs), Reinforcement Learning (RL), Deep Q-Learning (DQL), and Fomal Verification.

1.1 Neural Networks

Neural Networks (NNs) are machine learning models inspired by the brain structure which used for approximating complex functions. They consist of layers of interconnected neurons (nodes) through which information flows to learn how to map input data to outputs.

1.1.1 Neural Network Structure

A neural network is devided into 3 types of layers:

1. **Input Layer:** Represents the input features of the network.
2. **Hidden Layers:** Layers between the input and output layers, where most of the computations and transformation of the features happens.
3. **Output Layer:** Represents the output of the network. Usually represent a classification probability vector.

Each adjacent layers are connected weights and bias. Each neuron process information using a activation function, which adds non-linearity to the model which allows it to approximate more complex functions.

1.1.2 TBA

TBA

1.2 Reinforcement Learning

Reinforcement Learning is a type of machine learning where an agent learns to make decision (actions) based on interactions with the environment.
Unlike supervised and unsupervised learning, reinforcement learning doesn't learn through given training dataset but through the feedback that agent get when interacting with the environment which come in the form of rewards. Therefore, reinforcement learning ofter considered as separate paradigm of machine learning that focuses on decision-making and sequential actions in an environment.

1.2.1 Markov Decision Process (MDP)

MDPs provide a mathematical description of the environment, defined as the following tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of all possible states, $\mathcal{A}$ is the set of all possible actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the probability transition function which defines the probability of transitioning to state s_{t+1} taking action a_t from state s_t, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function which gives the expected reward for taking action a_t at state s_t, and $\gamma \in [0,1)$ is the discout factor.
In MDPs $\mathcal{P}(s_{t+1}|s_t, a_t) = \mathcal{P}(s_{t+1}|s_t, a_t, \ldots, s_0, a_0)$ which means that the following state depends only on the current state and action and not on the past states and actions. In reinforcement learning the environment often can be modeled as an MDP.

1.2.2 Key Componants of Reinforcement Learning

- **Agent:** Interacts with the environment and decides which action to take in each state.
- **Environment:** Responds to the agent's actions and provide the new states, usually can be modeled by a probability transition function $\mathcal{P}(s'|s, a)$.
- **State $s \in S$:** The current situation of the environment. The state space, $\mathcal{S}$, is the set of all possible states.
- **Action $a \in A$:** The decision the agent made in a given state. The action space, $\mathcal{A}$, is the set of all possible actions the agent can take, potentially depending on the current state.

- **Policy** $\pi(a|s)$**:** A mapping from states to probabilities of selecting each possible action. A policy can be deterministic ($\pi(s) = a$) or stochastic ($\pi(a|s)$ gives a probability distribution over actions).
- **Reward** $R(s, a)$**:** The immidiate reutrn value after preforming action a in state s. A function the maps pairs of action and state to a real scalar $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.
- **Value Function** $V^{\pi}(s)$**:** Function $V : \mathcal{S} \to \mathbb{R}$ estimates the cumulative reward (future reward) starting from state s and following policy π.
- **Q-Function** $Q^{\pi}(s, a)$**:** Also know as Action-Value Function. Function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ estimates the cumulative reward starting from state s, taking action a, and following policy π.

1.2.3 Types of Reinforcement Learning

There are 2 primary types of reinforcement learning approaches:

Model-Free: In this approach, the agent does not have a model of the environment but learn only from interactions with the environment.
There are 2 main methods of making a model-free RL:

- **Value Based:** The agent learns a value function which estimates the value of the expected reward in a given state and/or taking an action.
- **Policy Based:** The agent learn a policy that maps states to actions without learning a value function.

Model-Based: In this approach, the agent builds a model that mimics the environment and tries to predict the next state given the current state and action. Model-Based are more data efficient but more complex to build.

1.2.4 The Reinforcement Learning Problem

On each step t, the agent observe a state $s_t \in \mathcal{S}$ in the environment, selects an action $a_t \in \mathcal{A}(s_t)$ according to its policy π an recives a reward r_{t+1}, and the environment transition to a new state s_{t+1}. The environment dont have to be deterministic which mean the next state can be difined by the probability function $P(s_{t+1}|s_t, a_t)$.
The goal of the agent is to maximaze the future reward, also known as return. At time t

looking forward T steps in the future, and considering a discout factor $\gamma \in [0, 1)$ (to ensure convergence as $T \to \infty$) one can represent the return as:

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-1} r_{t+T} = \sum_{k=0}^{T-1} \gamma^k r_{t+k+1}$$

Notice earlier description of the Value Function and Q-Function can be expressed as:

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] \quad Q^\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$$

The objective is to find a policy π^* which maximize the expected return form any state s:

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi[G_t | s_t = s]$$

1.2.5 Exploration vs. Exploitation

One main challenge in reinforcement learning is the trade-off between exploration and exploitation.

- **Exploration:** The agent tries new actions in order to discover better rewards and avoid getting stuck in suboptimal policy.
- **Exploitation:** The agent select actions the based on his current knowledge yeilds the highest reward.

Both of them are crucial for the success of the agent. One strategy of balancing them is called ϵ-greedy which the agent explore each step at a probability of ϵ.

1.3 Deep Q-Learning

TBA

1.4 Formal Verification

TBA