Solving an agility maze using Reinforcement Learning with Unity ML-Agents and Curriculum Learning

Mario da Graca

Self-optimizing Systems
Winter Term 2022/23
University of Applied Sciences (HAW Hamburg)
Berliner Tor 5, 20999 Hamburg

Abstract. This project report examines curriculum learning in Unity ML-Agents using a 3D agility maze task as an example. The task involves tilting the board on two axes to guide a marble through a maze, filled with walls and holes, through which the agent must steer the marble in order to follow the correct path and reach the end. The agent's performance was evaluated using two different approaches: standard reinforcement learning and curriculum learning. Curriculum learning involves starting the agent with simpler sub-tasks, and gradually increasing the difficulty over time. The paper covers the technical details of the project, including setup of the environment, tuning of parameters, reward engineering and performance analysis of the trained agent.

Keywords: Reinforcement Learning · Curriculum Learning · Unity ML-Agents

1 Introduction

Machine learning (ML) is a subset of Artificial Intelligence (AI) that focuses on the development of algorithms and statistical models that allow computers to learn from data and make predictions or judgments without being explicitly programmed. Reinforcement Learning is a type of Machine Learning that focuses on teaching agents to make decisions in a given environment by maximizing a reward signal. It is a method for an agent to learn how to behave in a given situation by executing specific actions and watching the rewards that the environment provides.
ML and RL are widely used in a range of applications, such as robotics, autonomous vehicles, finance, healthcare and games. This report explores Reinforcement Learning in the latter one, by creating a complex and agility focused maze environment, that an agent has to successfully solve. With the help of Unity's ML-Agents [2] a model is trained using the standard Reinforcement Learning and Curriculum Learning [4] approach. There are several goals defined for this project:

1. Recreate the original wooden board game in Unity and set up the environment for training an agent.
2. Train the agent on the whole maze and try to find the best performing set of hyperparameters and rewards to successfully follow the game-rule given path to the end, without falling into a hole and thus restarting the game.
3. Train the agent using Curriculum Learning, where the agent starts of in an easier environment whose difficulty increases on successful completions. For this approach the environment has to be slightly modified compared to the previous task.
4. Evaluate the performance of the agent on both tasks using the best set of hyperparameters found.

The following chapter will explain basic concepts of Reinforcement Learning, Curriculum Learning and describes an overview of important principles used in developing the application. Chapter 3 goes into detail describing the environment setup, reward engineering process, agent configuration as well as metrics used to analyze the agent's performance. Finally the process and results are discussed to give an educated evaluation of the achievements and further optimizations.

2 Basics

There are several paradigms of ML, including supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the model is trained on a labeled dataset, where the correct output is provided for each input. Unsupervised learning involves training the model on an unlabeled dataset, where the goal is to find patterns or structure in the data. RL is a type of learning where an agent learns to make decisions by interacting with an environment, and receiving feedback in the form of rewards or penalties. Sutton and Barto [6] compare this approach to an infant with no explicit teacher, but with a direct connection to it's environment.

> "Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about what to do in order to achieve goals."[6]

2.1 Reinforcement Learning

In contrast to, for example supervised learning, where the model tries to learn a function that maps the input data to the specified targets, in reinforcement learning the agent's goal is to learn a policy. The policy describes a mapping from states to actions, that maximizes the expected cumulative reward over time. This process is particularly useful when the desired behavior is not explicitly provided, but can be inferred by the agent through trial and error.
The agent observes the current state of the environment and chooses an action based on its current policy. The environment then changes and the agent is rewarded or punished based on the action it took. By utilizing this feedback, the

agent updates its policy and enhances its decision-making over time. Value-based approaches, which learn the value of different states or state-action combinations, and policy-based methods, which directly learn the best policy, are two types of RL algorithms. There are also model-based techniques, which use a model of the environment to anticipate the outcomes of certain actions. For this task the policy-based method PPO was chosen.

2.2 Proximal Policy Optmization (PPO)

Proximal Policy Optimization (PPO) is a type of algorithm used to optimize a reinforcement learning (RL) model. It is an improvement on a previous algorithm called Trust Region Policy Optimization (TRPO) [5]. PPO's primary notion is to take incremental, -proximal- steps toward refining the policy (the set of behaviors the model does in different scenarios) rather than huge, possibly unreliable changes. PPO does this by employing a unique loss function (surrogate loss) that promotes the model to make adjustments that are comparable to past actions rather than making large jumps. This makes the learning process more steady and reduces the likelihood of being stuck in a bad policy. It also uses the actor-critic method [3]. In summary, the actor-critic approach is made up of two parts: the actor, who is in charge of selecting actions, and the critic, who is in charge of evaluating the quality of the actions. The actor is optimized by maximizing the surrogate loss function, while the critic offers feedback by assessing the value of the state-action pairs. This enables the agent to learn both the policy and the value function at the same time. The whole process is repeated until the agent solves the the environment.

2.3 Unity and ML-Agents

Unity is a 3D game engine developed and maintained by Unity Technologies [1].

References

1. Haas, J.K.: A history of the unity game engine (2014)
2. Juliani, A., Berges, V.P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D.: Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627 (2020)
3. Konda, V.R., Tsitsiklis, J.N.: OnActor-Critic Algorithms. SIAM Journal on Control and Optimization **42**(4), 1143–1166 (Jan 2003). https://doi.org/10.1137/S0363012901385691, http://epubs.siam.org/doi/10.1137/S0363012901385691
4. Narvekar, S., Stone, P.: Learning Curriculum Policies for Reinforcement Learning (2018). https://doi.org/10.48550/ARXIV.1812.00285, https://arxiv.org/abs/1812.00285, publisher: arXiv Version Number: 1
5. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (Aug 2017), http://arxiv.org/abs/1707.06347, arXiv:1707.06347 [cs]
6. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. Adaptive computation and machine learning series, The MIT Press, Cambridge, Massachusetts, second edition edn. (2018)