CPSC 440 Project - UBC WT2 2023

Frederick Shpilevskiy
15617962
fshpil@students.cs.ubc.ca

Mauricio Soroco
60785250

Joel Vandervalk
98820277

1 Introduction

Reinforcement learning (RL) is an area of machine learning (ML) that investigates how an agent learns from its interactions with the environment and consequences of these interactions [1]. In a typical RL model, an agent is able to observe its environment and take actions. The agent receives an input as the current state of the environment and makes a decision about which action to perform. The action will lead to a new state for which there may be an associated reward or penalty. The reward value is used to determine the value of an agent's decision. RL involves learning an optimal strategy for making sequential decisions where both immediate and future consequences of actions are considered. This learning emerges from a trade-off between exploring the environment's state space and exploiting the information about the environment that has already been gathered. Typically, an agent will start by exploring its environment by making random decisions; as it learns more, it will take actions based on decisions informed by earlier exploration. Finally, an agent can generalize their knowledge of the environment to un-visited states to make informed decisions.

We will use RL to train an agent to use information from its neighbourhood to navigate a dynamically changing environment. A spaceship will learn to navigate a changing two-dimensional star system by travelling to checkpoints within the system. Both the bodies within the star system and the spaceship will be affected by gravitational forces. The environment will change over time as bodies move according to the affects of the gravitational forces on one another. The spaceship's trajectory will be similarly affected. The spaceship will be able to observe its surroundings and move to avoid obstacles and approach checkpoints. Using only proximity sensors, the agent will not have access to any hidden variables that might reveal the rules of the dynamics of the system.

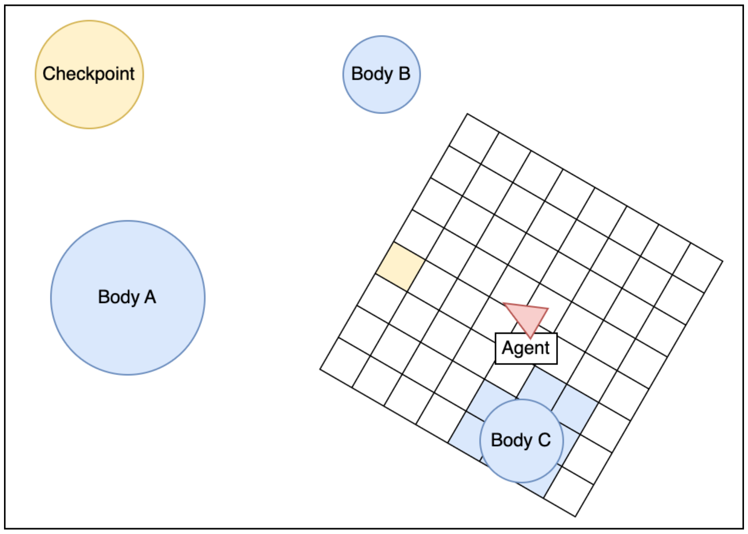

Figure 1: A diagram showing how the agent perceives the environment. We create a grid centered at the agent's position where covered boxes contain obstacles and the box closest to the goal is filled in.

The problem is interesting because it requires an agent to explore a dynamically changing environment. The problem is flexible; there are several way that it can be made more complicated. We will start with a simple discrete set of actions: throttle up, down, left, and right, or none. We can also make the agent's actions have a delayed affect by limiting the agent to only be able to control its acceleration. In the same way, we can replace the left and right actions with a rotate action such that the agent will need to rotate in order to travel in a different direction. As a result, the agent will only experience a change in position in the future, so any rewards from their action will not be immediate. The speed of obstacles can be changed such that the agent must behave more reactively. Similarly, we can reduce the power of the agent's thrusters to incentivize it to use the gravity of the system to its advantage. We will decide whether to add these complications to the problem once we have explored it more.

Our goal is to learn whether the agent is able to understand the logic underlying the dynamics of their environment and use this understanding to navigate the changing system. We will train the agent within a physics simulation to find checkpoints. We will test the agent's understanding of the system's physics with hand-designed test environments. For example, we can investigate whether the agent is aware of the gravitational pull of massive bodies by requiring the agent to travel around a body with extreme gravitational pull. One of the problems we will encounter is balancing the ability of the agent to overcome gravity. If the agent is able to easily move away from the bodies, then the problem is a trivial search problem.

The input state to the agent will be a grid of their neighbourhood. Bodies within the agent's neighbourhood will have their respective boxes filled in as obstacles. The agent will also know in which direction their goal is; the box closest to the goal will be filled in as the goal. All other boxes will be empty. We hope that this kind of state will provide the agent with information about the direction and distance of goals and obstacles within its vicinity. We can control the size of the boxes and the grid to give the agent a higher resolution representation of its vicinity or a larger area of sight. We can also provide the agent with multiple grids at different time-steps to provide it with past information. Past information may be necessary so that the agent can perceive the effects of the forces and its own movement. Since we are starting with discrete actions and a large state space, Deep Q-Networks (DQN) will be an excellent fit to this problem.

Our contribution involves employing an existing reinforcement learning method, DQN, to train an agent to navigate a dynamic environment, the logic of which the agent is not aware. We will investigate an agent's ability to navigate physical systems and learn to leverage the laws that govern them.

2 Background

Specifics of DQN (pseudocodes and stuff)

References

[1] M. van de Panne. [Online]. Available: https://www.cs.ubc.ca/~van/cpsc533V/index.html?

Wikipedia - Reinforcment Learning
Dr van de Panne’s Slides