Defeated Zombies Through Deep Q-Learning in Minecraft

Hiroto Udagawa, Tarun Narasimhan, Shim-Young Lee

11/20/2016

1 Introduction

Before reinforcement learning was widely recognized as one of the most effective methods for training an agent to perform various tasks, most approaches heavily relied on engineering based on specific knowledge about the agent and the environment in which the agent operates. Specifying appropriate actions the agent needs to execute for every situation it can possibly encounter is not only unfeasible but also ineffective because the agent cannot react appropriately when faced by the new environment. With Reinforcement learning, however, we only define the reward and set of actions available for the agent, and it gradually learns to make decisions that will maximize the aggregate future reward.

In this project, our agent plays the video game Minecraft. The game does not have a predefined objective, but it allows the player to navigate the world, build and destroy structures, and fight an enemy in the first person point of view. The task we chose for this project is to train, with reinforcement learning, an agent to kill as many monsters — which can damage and ultimately kill the player — as possible in a confined space. Our agent will only be equipped with a sword to defend itself from the monsters. The ultimate goal is to have the agent learn what policies to execute based on the information it receives about the surroundings and the raw pixels from the game play screen.

2 Malmo and Previous Research

Project Malmo is an open source platform that offers an interface for researchers to experiment with different approaches to artificial intelligence via the online game Minecraft, which is a popular sandbox video game that was released in 2011. It is an ideal environment for artificial intelligence research because it allows users to test endless situations within a relatively simple framework. In the game, users control agents which can move around a map, fight computer-generated opponents, and build structures. Project Malmo allows researchers to control agents' movements and environment using Python scripts. By combining different states, actions, and spaces, researchers can create different environments and define various tasks in which and for which the agent trains with reinforcement learning.

Other applications of reinforcement learning in video games include DeepMind famously training a convolutional network with Q-learning to play Atari games using raw pixels as inputs and a value function estimating future rewards as output, which ultimately outplayed human experts [**?**].

The main inspiration for our project is "Application of a Deep Learning Algorithm on Minecraft" [**?**]. The author tested the ability of an agent to adapt to the task of destroying as many animals as possible. The author set up a scenario where the agent was set in the middle of room, with only three possible actions: turn left, turn right, and attack. Based on the visual input from the agent, the author was able to use deep Q-learning to train the agent to effectively destroy as many animals as a human. We plan to add on to this work by giving the agent the ability to move while attacking. Furthermore, animals in the game do not have the ability to attack and damage the agent. We hope to use monsters instead so that our agent will have to figure out not only how to kill monsters but also defend himself.

Another machine learning application of Malmo was "Exploratory Gradient Boosting for Reinforcement Learning in Complex Domains", in which the authors sought to train an agent to climb as high as possible up the given terrain [**?**]. They used visual input from an agent to train a reinforcement learning algorithm, and based the reward system on the agent's altitude.

3 Method

So far our work has mostly involved getting familiar with the Malmo environment and completing simple reinforcement learning tasks such as having our agent navigate a path of good and bad blocks.

However, our real learning problem will be much more complex. Thus in this section we will lay out the different parameters of our project that we will need to implement next.

3.1 MDP Formulation

3.1.1 Actions

The actions the agent can execute are the continuous movements of the agent as well as the ability to attack with a weapon. We will model the agent like a car, where the agent has the ability to move forward and back and turn left and right. We allow a single attack action that will allow the agent to kill monsters that it faces.

3.1.2 States

The states will be composed of the image of the player's vision represented by the pixels of the screen. In this way, the agent will be able to react to its environment like a human would. Fortunately, Project Malmo provides the functionality to retrieve the image from the screen and feed this back into our algorithm.

3.1.3 Reward

Currently, our plan is to give a reward of +1 for a successful hit onto a monster and a reward of +5 for a successful kill. Also, there will be a negative reward of -1 when the player gets hit.

3.1.4 Deep Q-Learning

We will use Q-learning to have our agent learn policies to best fight these monsters. We will hopefully utilize the previous work done by Google DeepMind to also use a convolutional neural network to approximate the Q-function iteration after iteration. With the ability to extract the images from the game, we believe we will be able to turn our learning problem into one that is similar to those worked on by DeepMind.

3.1.5 Possible Extension: Deterministic vs Stochastic

In general applications of reinforcement learning, researchers incorporate uncertainty over which action an agent will take; this is done using a probability distribution over all the possible actions an agent could take from a particular state. In Minecraft, an agent performs a given command with 100% probability.

We plan to incorporate this uncertainty by having an agent take a different action than the one we give it with some probability greater than zero. In the discrete action space, this could be modeled by having the agent take an action other than the one we gave it (e.g., move left or right rather than move forward) with some probability defined by a probability mass function. In the continuous action space, we could have the agent take an action other than the one we gave it with some probability defined by a probability density function.

In addition, we could further complicate matters by having the algorithm perform without knowing these probability functions beforehand. Thus, rather than choosing the optimal action based on its knowledge of the probabilities, it would have to estimate the parameters of these probability functions.

3.2 Scenario

The agent will begin in the middle of a completely flat room with randomly generated zombies. The room will be indoors and a uniform color so that the algorithm will not have to account for differences in lighting and colors. The zombies will automatically approach the agent and damage it if they get too close. The agent will be equipped only with a sword to fend off the zombies. Over time, new zombies will continue to

spawn, and the game will continue to run until the agent loses all of its health. At this point, the score will be calculated based on the total number of zombies that the character was able to kill.

References

[1] Van de Goor, Elodie. *Application of DeepLearning Algorithm on Minecraft*, Master's Thesis. September 2016.

[2] Abel, David; Agarwal, Alekh; Diaz, Fernando; Krishnamurthy, Akshay; Schapire, Robert E. *Exploratory Gradient Boosting for Reinforcement Learning in Complex Domains*, March 2016.

[3] Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Graves, Alex; Antonoglou, Ioannis; Wierstra, Daan; Riedmiller, Martin *Playing Atari with Deep Reinforcement Learning*, December 2013.