CS 4200 Project Proposal: “Super Mario Bros. AI”

Kendall Haworth

Abstract

My project proposal is to create a neural network to play the game Super Mario Bros. The neural network will be trained with reinforcement learning algorithms and will only be provided a visual representation of the game screen. The network will be evolved to accomplish several different goals and tasks within the game.

Introduction

For this project, I plan on creating a neural network to play Super Mario Bros. Neural networks have always fascinated me from the videos I have seen on YouTube and the brief explanations of them I have heard from others. I have never done anything with neural networks before, nor do I understand how they are structured or how they work, so creating a project that uses a neural network will be a great introduction for me and will allow me to get into more complicated AI algorithms in the future. Creating a neural network that plays a video game, such as Super Mario Bros., will be more entertaining to watch and code than a neural network that solves a problem that is not a video game. I chose Super Mario Bros. because I played it growing up and it is relatively simple, although there is the potential to explore more complicated algorithms with it.

To make my work unique from that of others who have made neural networks for Super Mario Bros., I plan on training my neural network with reinforcement learning algorithms such as deep Q-learning or policy gradient methods. I will also attempt to use a visual representation of the screen as the input to the network rather than physical data, as physical data network interpretations in Super Mario Bros. have already been done (Bling 2015a). The visual representation is the actual image displayed to a human player, whereas the physical data is an array that gives the exact level layout of all enemies, items, and terrain. Finally, once the algorithm can complete a single level, I will change the objective of the program to trying other things that have not been done before in Super Mario Bros., such as solving two levels simultaneously with one input for the controls.

Related Work

Programmers have already used a wide selection of AI techniques, including neural networks, to play Super Mario Bros. There are guides online on how to install and communicate with the game through python (Messier 2017) and short YouTube videos showing off finished neural networks playing the game (Hamilton 2017). The code for these algorithms is also available for download (Bling 2015b). However, this documentation is very shallow, with the videos and code only covering the very first level in the game.

The most extensive research in this field was made during the Mario AI Championship competitions hosted in association with the IEEE Games Innovation Conference and the IEEE Symposium on Computational Intelligence and Games from 2009 to 2012 (Togelius, Karakovskiy, and Baumgarten 2012). The goal of the competitions was to develop controllers, or algorithms, that could play an alternate version of Super Mario Bros. as well as possible. Controllers were rated based on distance reached, time left, and number of kills. The world generation for the competition was created using a modified version of Markus Persson’s “Infinite Mario Bros,” wherein an infinite number of levels is procedurally generated based upon content from Super Mario Bros. 3 and Super Mario World (Pedersen, Togelius, and Yannakakis 2009). Surprisingly, A* outperformed all other algorithms by a wide margin. It was followed, in order of scoring, by rule-based algorithms, genetic programming, neural networks, and state machines. While neural networks did not perform the best, they were sufficient enough that two individual contestants who used them made the top 15 in the 2009 competition.

Other work in this field includes research into making a controller for Super Mario Bros. that mimics the playing style of a human player. This project was conducted at the IT University of Copenhagen in 2013 with the intention of applying the research to making non-player characters, or NPCs, in video games more believable and making better AI that can demonstrate how to play a game in a human-like manner (Ortega et al. 2013). They concluded that methods based on neuroevolution, or methods that helped neural networks automatically evolve their internal structures and connections, produced the best controller that performed similar to a human player based upon both technical analysis and the opinions of human spectators.

Method

To create a neural network that plays Super Mario Bros., I will need to install the open-source game on my system and have a way to interact with the game via code. This can be accomplished through using Open AI Gym or following a tutorial online for the installation process (Brockman et al. 2016). Once this is done, I will have to research neural networks and reinforcement algorithms and learn to create and use them for the purposes of the game. In 2013, a company called DeepMind Technologies created a convolutional neural network trained with a variant of Q-learning and applied it to seven Atari games (Mnih et al. 2013). Using just the raw pixels of the game screen as input, they were able to beat all previously made AI for six of the seven games. Since their goal is similar to mine, I will be able to learn from their research. Once I have learned to make a neural network and train it with reinforcement learning, I can begin experimenting with the network to accomplish several goals. My goals will be the following:

- Complete a single level from start to finish
- Have the controller play two different levels simultaneously with one input for the controls
- Complete multiple levels in a row (during the same run)
- Obtain the maximum number of points
- Defeat the most enemies
- Collect the most coins
- Obtain as many hidden items as possible
- Find as many hidden paths as possible

I plan to complete a single level first, since it is the simplest of the above tasks, and evolve the neural network from that point to accomplish the other goals. To explain the goal of completing multiple levels in a single run, Super Mario Bros. features 32 levels, providing a very large testing space. Some of these levels even feature different game mechanics, such as swimming rather than jumping, where the player must repeatedly press the "A" button to stay afloat. Since competitions to create controllers for Super Mario Bros. used Infinite Mario Bros., which does not feature water levels, as the terrain generator, findings have not been published for completing all the original Super Mario Bros. levels in a single run. It would be interesting to see if a single network can complete all 32 levels or if different networks must be trained depending on the different game mechanics between levels.

The data required to create a neural network for Super Mario Bros. will be the input for every action Mario can perform (i.e., button the player can press) and a representation of the game screen as a human would see it. Indicators of fitness, or a way to gauge how successful a run is, are also required, and these may change depending upon the program objective. For example, to develop the neural network to complete a single level, the indicator of fitness would require the distance from the starting point Mario has traveled and the time that has elapsed from when Mario started the level to when he dies, completes the level, or times out. Other indicators of fitness could be the total points earned, the number of enemies defeated, or the number of coins collected, as is applicable to the program's goals.

Fortunately, all this data can be easily obtained from directly interacting with the game engine, which has several methods available to obtain information from the current game session. These methods are described in a markdown file on the emulator's download page on GitHub (Paquette 2016).

A good way to evaluate the effectiveness of the controller is if it can complete the goals described above or if it is able to successfully increase its fitness as it is trained.

Timeline

The first milestone will be to successfully install the open-source Super Mario Bros. game on my system and be able to interact with the game through code. After this, the next accomplishment will be to successfully implement and train the neural network using reinforcement learning methods. A good indication that this is working will be if the algorithm progressively gets better each time it is run and is eventually able to complete a single level. I hope to achieve this goal within the first few weeks. After this point, other tweaks and changes to the code can be made and other goals and tasks explored, such as the ones described in the method section. Since I am doing this project alone, I will be doing everything.

References

Bling, S. 2015a. Mari/o - machine learning for video games.

Bling, S. 2015b. Neatevolve.lua.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Hamilton, K. 2017. Neural network plays super mario brothers (world 1-1) *400% speed* part 2.

Messier, C. 2017. Getting mario back into the gym: Setting up super mario bros. in openai's gym.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Ortega, J.; Shaker, N.; Togelius, J.; and Yannakakis, G. N. 2013. Imitating human playing styles in super mario bros. *Entertainment Computing* 4(2):93–104.

Paquette, P. 2016. Readme.md.

Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2009. Modeling player experience in super mario bros.

Togelius, J.; Karakovskiy, S.; and Baumgarten, R. 2012. The 2009 mario ai competition.