Project Proposal - Deep Reinforcement Learning on Super Mario Bros

Matthew Chen Isabel Bush

1 Abstract

For this project we will be implementing various reinforcement learning algorithms to create learning agents to effectively play a variant of the classic Super Mario Bros game. The game is challenging from an AI standpoint as the environment is stochastic with randomly generated levels, and the state space is large. To address these challenges we will use Q-learning with various Q-function approximations ranging from simplistic feature extractors to deep learning implementations.

2 Task Definition

The game is defined by a simulator that receives actions from our controller and outputs various environment attributes. Our controller will extract state and rewards from the environmental attributes and send Mario control actions back to the simulator based on the learned policy.

We will test out inputting rewards to our controller at each step as a combination of metrics (such as positive rewards from coins collected, monsters destroyed, level completion and negative rewards for collision and death) or simply providing the final fitness score as a reward at the end of the game. We may evaluate our controller by running it on multiple game levels and determining it's overall fitness score (combination of bonuses collected, monsters killed, and distance travelled), as well as a simple distance travelled metric to compare with our benchmarks, described in a later section.

Actions: { Left, Right, Jump, Crouch, Run/Fire }

State:

Status - Running, Won, Dead

Mode - Small, Large, Fire

onGround - True, False

isAbletoJump - True, False

hasShell - True, False

ableToShoot - True, False

CreaturePositions - Array of [x,y] coordinates

Obstacles - 22x22 array of obstacle type

3 Challenges

The main challenge is to find an optimal policy for Mario to follow given that the map of the environment is not known in advance, and the actions which are taken will lead to rewards in future time steps. Since the state transition function is unknown to the controller, it would be difficult to use offline methods – the controller must learn as it plays.

Another challenge is that we would like our algorithm to run in real time, which means we need to complete all calculations in at most 42 ms given the simulation frame rate. Finally, it is not apparent what the optimal feature extraction function is for this specific problem and thus we will experiment with handcrafted feature selection as well as deep learning techniques.

4 Approaches

To address the aforementioned challenges, we will implement reinforcement learning algorithms, which are well-suited to Markov Decision Processes with unknown transition probabilities. We will use Q-learning so that we may calculate the optimal Q-value (and thus optimal policy) directly, without needing to first calculate estimations of transitions and rewards.

Since the state-space is large and our controller is unlikely to experience all possible states, we will use function approximation to generalize to unseen states. We will begin by defining our own features based on simplifications and combinations of the provided environment attributes, and then we will explore learning the feature vector as well through a neural network representation.

5 Benchmarking

Our baseline method will be a random policy in which Mario picks a random action with a forward moving bias. We chose this as our baseline method since the controller chooses actions regardless of state or reward. We use as our oracle the winning scores for the 2009 Mario AI competition [1]. The difference between the two bounds should provide a good prospective on the improvement that can be made with our learning controller. The base metric that we will use to start off is the distance the agent was able to progress at a given level summed across all levels. Our baseline agent averaged 13,713 while the score for our oracle was 46,565.

6 Previous Work

The Mario AI competition ran for a few years, but few submissions used reinforcement learning. The winning controllers used A* search since the initial metric only took into account distance travelled and the controllers were not required to learn from scratch. Next-highest ranking controllers used rule-based heuristics [1]. In the years since competition, a group has implemented a reinforcement learning algorithm that managed to outperform the rule-based controllers, but still fell short of the A* scores [2]. We hope to explore different state and feature definitions than were used in that study, and to extend the algorithms to use deep learning as was done for the Atari games [3].

References

[1] S. Karakovskiy, J. Togelius *The Mario AI Benchmark and Competitions* 2012.

[2] J. Tsay, C. Chen, J. Hsu *Evolving Intelligent Mario Controller by Reinforcement Learning* 2011.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller *Playing Atari with Deep Reinforcement Learning* 2013.