Preparation of Papers for AIAA Technical Journals

Liam Brown * and Jeremy Crowley†
Stanford Universiry, Stanford, CA, 94305

In this paper, we discuss a reinforcement learning strategy for the video game Super Smash Bros Melee.

Nomenclature

β = basis function for global approximation

a = cylinder diameter

I. Introduction

Reinforcement learning (RL) is an area of research that focuses on finding an action that will maximize a reward function, given the state of an agent. Finding the optimal action requires large amounts of data gathered from the agent interacting with the environment. The ability to gather reliable data is crucial to the learning process because this provides the learning algorithm with an accurate representation of how the state evolves with a given action.

RL has recently become a topic of interest due to its ability to solve problems with dynamics that can be difficult, if not impossible, to model given computational limitations. The underlying problem of complex dynamics can be addressed using model-free reinforcement learning and is widely used over a large set of applications including control theory, natural language processing, image recognition, and medical diagnoses. The most common model-free reinforcement learning algorithm is call Q-learning and it relies on applying incremental estimation to the Bellman equation.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{1}$$

Model-free learning can be applied online (updating the Q values after each action) or applied as an offline strategy using batch learning (gathering data for a fixed period of time and updating the Q values). Online implementations require the runtime of the update equation to not interfere with the agents ability to perform an action at the next available time step. Batch learning, is a simple yet effective method to apply reinforcement learning to an agent in a time critical scenario.

*Graduate Student, Aeronautics and Astronautics.
†Graduate Student, Aeronautics and Astronautics.

Super Smash Bros Melee presents an environment with dynamics that are difficult to model without knowledge of the underlying engine used to build the game. In addition to complex dynamics, the game has a state space that would be infeasible to represent as discrete values. A model-free approach eliminates the need for a state transition model and generalization allows the agent to approximate the optimal policy given limit data. Given the nature of the problem we would like to solve, we chose to implement a perceptron Q-learning algorithm.

We define the action space as $\mathbb{A}$ and the state space as $\mathbb{S}$. The action space contains a set of actions that are mapped to a combination of buttons being pressed on a gamecube controller. The state space contains a set of continuous and discrete variables that represent the agent in the environment.

II. Problem Statement

It is infeasible to discretize the state space for this game if we wish the develop a learning algorithm that can be executed in a reasonable amount of time with a standard personal computer. To account for this, we must apply a learning algorithm that can generalize from limited experience.

$$Q(s, a) = \Theta_a^T \beta(s) \tag{2}$$

III. Applications to Super Smash Bros Melee

- state evolver / intractible state space / discretized controller

A. Discretized actions

The controller for the nintendo game cube can be thought of as a continuous action space for Super Smash Brothers Melee. There are two analog control sticks which can be placed at any value from -1 to 1 in both the x and y direction, two analog shoulder buttons which are functionally identical and range from 0 to 1, as well as four digital face buttons.

To reduce the number of potential actions, repetitive buttons were discarded the analog inputs were discretized. The result is the following set of potential inputs:

B. Basis Functions

Since our project is based on perceptron Q-learning

C. Reward Functions

In order to apply perceptron Q-learning to to SSBM, a set of reward and basis functions had to be defined. As described in the introduction, the goal of the game is to ultimately knock your opponent off the stage, which is increasingly possible as their damage goes up since the damage causes them to fly further. This indicates a careful balance between accumulating damage to facilitate a knock out move, and overly focusing on accumulating damage

with no regard for the ultimate goal of getting a knock out. To tackle this, we defined the reward function between states to include a decay for damage dealt when the opponent is already at higher percentages, as well as a penalty for taking damage. In order to prevent suicidal behavior of the agent, a reward was also given when the agent managed to move from being off of the side of the stage in state s, to on the stage in state s'

Denoting the opponents damage as d_o, the agents damage as d_a, and the on stage parameter of the agent as a true / false flag ON, the reward function became:

$$R = (d'_o - d_o)exp(-.01 * d_o) - (d'_a - d_a)exp(-.01 * d_a) + \delta_{0,ON}\delta_{1,ON'}$$

In addition to this reward given out at each state transition step, a novel method for assigning rewards to significant important states was formulated. To assign rewards to killing the opponent, the last action the agent took that resulted in an increase in damage to the opponent is recorded. When the opponent dies, a large reward is assigned to this "last damaging action". This is due to the long lag between hitting the opponent and them dying, and the necessity of avoiding assigning large rewards to actions that had nothing to do with the opponnent dying.

Additionally, a finite length action history is tracked for the purposes of assigning negative rewards to deaths. When the agent dies, all actions in the action history are penalized, as the agent could have taken an action in this time frame that may have prevented its death but ultimately chose not to. An example of this is the agent being slightly off the edge of the stage and not choosing to jump back on. The action selected instead of jumping back on is then penalized.

IV. Results

V. Conclusion

Appendix

An Appendix, if needed, appears **before** research funding information and other acknowledgments.

Funding Sources

We would like to thank mom and dad.

Acknowledgments

We would like to acknowledge libmelee for providing an open source solution to obtaining information about the game while it is being played.