AA228 Final Project Status Update

Jeremy Crowley and Liam Brown

November 16, 2018

1 Introduction

For our final project we are training an agent to play and defeat the default agent in Super Smash Brothers Melee using global approximation Q-learning. To do this we are leveraging the open source python library "libmelee", which provides state information from the game as well as the Gamecube emulator Dolphin.

2 State and Action Space

talk about how we discreized controller actions, states, etc

3 Progress

In order to verify our ideas, we decided to initially train an agent to jump. To do this, we first gathered data from a completely random agent playing the game for 20 minutes. We then constructed what we believed were adequate basis functions, consisting of the agents current height off the ground taken to various powers $\beta(s) = [y, y^2, 1/y]$. The weights θ_a for different actions for the basis functions were trained using batch-learning global approximation as in algorithm 5.5 of DMU[1]. Rewards were assigned when the agent transitioned from being on the ground in state s_t to off the ground in state s_{t+1}, and gave penalties when the agent remained on the ground between states s_t and s_{t+1}. This initial attempt failed, resulting in an agent that would simply crouch, indicating there were flaws in our strategy required refinement.

First, the initial design resulted in the agent acting every gamestep, inputting 60 actions a second. This is both unrealistic in that the game is not intended to be played with this many inputs, as well as harmful in that it does not allow for state-evolution between action inputs so the bot cannot determine the impacts of a single action on the change in state. The agent was changed so that it now acts on every twelfth frame, corresponding to five times a second. This is a large improvement in that it both better simulates actual gameplay, as well as accommodates the evolution of the state between decisions. It is also beneficial for training as the impacts of a single action are much clearer.

Another flaw to be addressed was the impact of "action-lockout", in which an action taken by the bot may have no impact on its current in game movements due to previous actions taken. An example of this is when the bot inputs a move that has a long wind-up animation that cannot be canceled by other actions. To address this, we now skip over actions input during an "action-lockout" when training the weights of our basis functions so that weights of actions that did not impact the state evolution do not change.

We also believe our initial data set was far too small to train our weights, and decided to regather data. We allowed a random agent to play overnight, gathering 9.5 hours (171,000 samples) worth of of data.

Lastly, the basis functions were reworked. Now the basis function $\beta(s)$ is a flag of 0 or 1 for if the agent is on the ground, as well as 400 flags indicating the current action of the agent (i.e. if the agent is currently in animation 326, the 326th element of the basis function for the animations will be a 1 with all the rest being a 0).

4 State Space

Our state state space contains the following parameters

Non-binary:

- x_a - position of our agent
- y_a - position of our agent
- $\dot{x}_a$ - velocity of our agent
- $\dot{y}_a$ - velocity of our agent
- x_d - distance from our agent to opponent
- y_d - distance from our agent to opponent
- $\dot{x}_d$ - relative velocity from our agent to opponent
- $\dot{y}_d$ - relative velocity from our agent to opponent

Binary:

- Υ_1 - 1 if not off stage
- Υ_2 - 1 if agent has remaining jump
- Υ_3 - 1 if agent is facing the opponent
- Υ_4 - 1 if the agent in in an attacking state
- Υ_5 - 1 if the agent in in an blocking state
- Υ_6 - 1 if the opponent in in an attacking state
- Υ_7 - 1 if the opponent in in an blocking state

5 Action Space

Our Action space contains the following parameters to form a reduced set of possible inputs from a gamecube controller. We discretize the analog stick (used to control the movement of agent) and remove redundant inputs.

Non-binary:

- Ψ_1 - x position of the analog stick
- Ψ_2 - y position of the analog stick

Binary:

- Ω_1 - L
- Ω_2 - B
- Ω_3 - A
- Ω_4 - X

References

[1] Kochenderfer, Mykel, *Decision Making Under Uncertainty.* The MIT Press, 2015.