Injecting Machine Learning into the Apprentice Learner Architecture, Project Milestone Report 3 15-400, Spring 2020

Cayden Codel (ccodel)
Project website: www.contrib.andrew.cmu.edu/∼ccodel

February 12, 2020

1 Major Changes

No major changes have occurred in the project, aside from the progress and goal-setting noted below.

2 What I've accomplished since last meeting

RumbleBlocks was last worked on in 2017. Between then and now, Unity, the game engine on which RumbleBlocks runs, has had several major updates, making several key graphical classes in the RumbleBlocks project obsolete. Therefore, I spent last week updating the game into a more modern version of Unity.

Additionally, since 2017, the Apprentice Learner architecture (AL) has undergone key changes to how it receives input. Previously, AL's input was a simple state-action interface, which allowed the RumbleBlocks game to provide the game state and the actions available to AL each time AL needed to take an action. However, AL now has fields for delayed feedback, which is integral to training AL. Therefore, I spent this week updating the RumbleBlocks interface to work with the new version of AL.

3 Milestone progress

Since two weeks ago, my milestones have become clear as I spent time with AL and RumbleBlocks. They are as follows:

- Update RumbleBlocks to a new version of Unity.
- Update RumbleBlocks to interface with the new version of AL.
- Develop an interactive trainer for AL inside RumbleBlocks.
- Get the new version of AL training on student play data.
- Develop and train a reinforcement learning agent in RumbleBlocks. To do so, the interface used to communicate with RumbleBlocks may need to be modified.

Currently, I have completed the first and am almost done with the second. For the third, RumbleBlocks used to have a kind of interactive feature, but it is outdated, as the feedback given in it is much too granular (think feedback per placement of block instead of for an entire structure), and so it will need to be reworked.

RumbleBlocks has been used in a few human studies, where play data was collected. Training AL on the play data speeds up training. RumbleBlocks has a replay feature, but tying that into AL may be an engineering effort.

To see engineering results for updating RumbleBlocks to Unity, see the **cayden/unity_update** and **cayden/www_refactor** branches on the RumbleBlocks GitHub (click here for a link).

4 Surprises

No surprises to report.

5 Looking ahead

Touched upon in the milestone progress section, my next two weeks will be spent revising RumbleBlocks to work in several ways with the new AL, including having an interactive trainer for realtime training with AL and a replayer that lets AL train on previously collected play data. Both Erik Harpstead and Danny Weitekamp have experience developing such software, so if I get stuck, I can schedule a meeting with either of them to point me in the right direction.

Looking even further ahead: once I get both AL and RL agents working in RumbleBlocks, I will need to compare the performance of the two. However, AL is evaluated based on errors, while reinforcement learning is evaluated based on score or fitness. Thus, some mapping between the two will have to be made in order to determine which model is performing better on any particular task in RumbleBlocks. The good news is that error type analysis is a major subproject underway in the Apprentice Learner group. I could consult them to see if I could take some common error types and convert that to a "score" to compare against RL's score.

6 Revisions to future milestones

Since last report, I have solidified my milestones, so see above.

7 Resources needed

No additional resources are required at this time.