CS 395T: Deep Learning Seminar

Project 5 Report

Isaac Buitrago, Clara Cannon, Ty Trusty

# Objective

Our goal for this project was to use reinforcement learning to train an agent to play SuperTuxKart.

# Proximal Policy Optimization

We chose to implement proximal policy optimization (PPO) as the reinforcement learning algorithm for this project. PPO is policy gradient method that alternated between learning from interaction with the environment and optimizing an objective function.

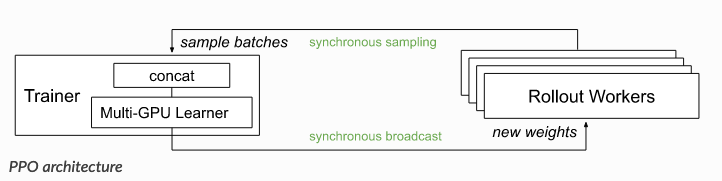


Figure I: Reinforcement learning architecture of Proximal Policy Optimization (PPO).

# Implementation

The Ray Reinforcement Learning Library package for Python offers a PPO training function. We used this function to train an agent to drive along the track and complete the race. The reward function was the negative sum of the distance from the center of the track added to the positive distance traveled in a single time step. The primary obstacle prior to training consisted of adapting SuperTuxKart to the Gym environment. Gym is a toolkit for developing and comparing reinforcement learning algorithms. The package comes with useful functionality for resetting the race, capturing the state space, and recording the sequence of actions taken and rewards received.

Once the environment configurations were set, training the agent became the next focus. The agent displayed poor learning behaviour from the outset. To bypass training time and encourage course corrections, we restarted the race once the cart traveled a certain distance away from the center of the track. The episode continued until the maximum number of steps was reached. This change contributed the most significant positive affect in agent performance. They helped initial training episodes, however agent performance quickly plateaued. Changing the step size and episode length did not resolve the problems. We discovered that increasing the learning rate helped performance marginally, but not enough to significantly affect the average reward obtained per episode.

The team decided to increase the training interval and continue adjusting environment parameters. Other changes that aided in the development of the agent included making a single prediction for a set number of time steps. In our case, we set the update horizon to seven. Updating on every time step was computationally expensive and difficult to change. Reducing the acceleration during agent training also boosted results.

We had to restart training upon the realization that we had misrepresented the state space. While the validation grader represented the current state as a pixel image, we had encoded the state as a Gym environment observation. This adjustment to the learning algorithm halted progress in agent training.

# Discussion of Results

It is important to reset the environment when the kart is stuck during training to ensure that the model has quality samples. We experimented with different reward functions and achieved the best results when the reward was the combined distance traveled in a time step and the distance from the center of the track. The only case where the agent experience extremely poor performance occurred when the cart turned around and started moving in the reverse direction. There were not enough training samples with this instance for the agent to learn how to correct its direction.

# Conclusion

The main stall in the learning of the agent was its failure to recover from extreme divergence away from the center of the course. In situations where the cart found itself off in the weeds, interaction with the environment was not sufficient to recover the correct objective function at the end of the episode. Even in cases when the race was restarted for the beginning, the learned weights for the previous trajectory may have had difficulty converging to the optimal values. While various combinations of reward functions and environment representations were tried, our experiments were not exhaustive. We were able to reach a score of 100 after ~20 minutes of training, but the results are still inconsistent. Given more time we expect that performance would be much better.