CSCI-4800/5800-003: AI with Reinforcement Learning

Term Project Report

Project Title:

DeepRacer: The Blind Leads the Blind

Team members:

David Darling
Ivy Truong

Problem Statement / Tasks of the project:

Our goal is to optimize a reward system for the DeepRacer that will improve the driving system to reduce the number of times it goes off the course and the overall lap time through reinforcement learning. We will benefit from this project because we will learn the basics of reinforcement learning through the Deep Racer's learning process of trial and error. We will also learn if the greedy approach or the far-sighted approach is the most effective in developing the best reward system.

Methodology:

1. Keep DeepRacer driving on track
    a. Used reward function sample that made sure the DeepRacer followed the centerline
    b. Added in reward value to make sure that the DeepRacer had all wheels on track
2. Reward DeepRacer for making progress to complete a lap on the track
    a. Used progress parameter to give the DeepRacer a reward based on how much of a lap it has completed
3. Find optimal speed for DeepRacer / maintain minimum speed
    a. Added in speed threshold and comparing the speed parameter to make sure that the DeepRacer was driving at a minimal speed and reward the DeepRacer for maintaining speed threshold
4. Continue testing model if reward function produces a graph that has the DeepRacer improving over time
    a. Tested the model every 15 minutes to make sure that the model wasn't overfitted
5. Testing DeepRacer on a different track and training on it for 30 minutes to see if the DeepRacer is able to complete a lap, wanted to allow DeepRacer to develop general behavior and decision making that can complete a lap for any track. (Focusing on accuracy and generalization)

Evaluation:

Our metrics for success with our final project related to completion of laps and the speed at which the laps were completed. Our first measure of success was to have our model complete at least 1 lap consistently. The next measure of success was to have our model complete 2 to 3 laps consistently. After attaining this level of success we wanted to improve our lap times by increasing the speed of the model.

Experiment Results and Discussion:

We used the AWS console as the main environment to test our DeepRacer using our reward function. The AWS console runs using a proximal policy optimization algorithm and the TensorFlow platform.

Below is a general outline of the experiments that we did when testing and determining the optimal reward function.

1. Choosing which base reward function framework to work with
    a. Option 1: following the centerline
    b. Option 2: staying within the track borders
2. Design reward function based on methodologies above
    a. After choosing aspect to focus on, train the model for 15 minutes and evaluate the model
        i. If the training model is not good, there will be no evaluation
    b. If the training model and evaluation is good the first round, choose to train the model again or add in another functionality

Discussion, Observations, and Results:

Please read the readme on the github repository using the link below to get a more in depth description of the experiment observations and results. The results of the training and evaluations for the models are in the Workday Results folder on github. The github is a private repository so please let us know if you haven't received the invitation.

DeepRacer Github: https://github.com/ivyT26/CSCI4800_AI_Final_Project

Prior/Related works around the universe:

Yes this problem has been solved by many other people, however there is the competitive aspect through racing and leaderboards, that fosters continuous adjusting of code to reduce lap times. So we will not be reinventing the wheel, but there are many different solutions to this problem using reinforcement learning and we will have to find our own solution. There might also be different

areas of success for the deep racer, based on who is working on it. Some developers might be trying to reduce the time it takes for the deep racer to learn how to go around a track (learning rate), while others are trying to get the fastest lap time. Our solution is mainly to benefit us, we want to have our deep racer complete the required laps while also having this development process improve our knowledge for Reinforcement Learning. We have come across some good information online that will help us with our journey.

[1] There is a good advanced guide that we came across. It discusses advanced techniques that were used by the author to help his university team get their deep racer to the top 1% ranking for AWS leaderboards. This guide will be helpful for us at the later stages of our project development if we want to make our deep racer more efficient. We benefited from some information that was on this website that related to multiplicative increase vs additive increase. We were originally using a multiplicative increase to modify our reward function but the website said that if your reward function has many parts that affect the reward function it can be a better approach to use an additive increase or decrease instead. We changed our reward function to reflect this new knowledge and we started noticing an improvement in our model.

[2] This website looks like it could be useful to us while we are working on keeping our deep racer on the track. It includes lots of good examples, code, thought processes, and common mistakes and how to solve them. This website contained some useful information that helped us develop our model further.

[3] Found a video of a talk given by a guy that has won 3 deep racer virtual circuit races. He gives tips and tricks that helped him with his development. This information would be beneficial to us to improve our deep racer design and help us with the thought process.

[4] We found this github that was created to help new people to AWS DeepRacer. The creator of the github wanted to have a record of his thought process as he sampled various reward functions. We looked through this guide and it helped give us some ideas of possible approaches we could take on our model.

[5] This article presented some really helpful insight on a few aspects of the reward function that could be implemented such as determining where the DeepRacer should be heading on the track, adjusting steering, and improving on the reward given to the DeepRacer based on how close the DeepRacer is to the centerline. A couple pieces of advice they have is to train the model on different tracks for a more stable behavior, observing different aspects of the reward function to know how the car will behave, and having more complex reward functions increases training time. In our model training, we focused on a few of the input parameters such as speed, progress, and all_wheels_on_track to see their effects on how the DeepRacer drives and makes its decisions.

[6] This article presented a couple other pieces of advice on how to improve and make sure to find the balance when creating an optimal model such as focusing on the accuracy at which the DeepRacer drives rather than the speed and overall lap time for the DeepRacer. He also mentions being extra cautious with training time, where too much training can make it more difficult to perform well with slight changes in the environment and not enough training can make it difficult for the DeepRacer to make correct decisions. During our model training for this project, we focused on training the model every 15 minutes, and would update the reward function if we observed any negative slopes in the reward graphs or no improvement after 30 minutes of total training after the last updated reward function.

Limitations and Future work:

Limitations:

Main limitations in the AWS console was that we weren't able to change as many variables as we would in a local environment, such as the action space and range for the input parameters. Using the AWS console was also limiting because we could only use the algorithm set in the console and were not able to implement our own algorithm. As stated in the progress report, we were initially thinking of trying out value function approximation or neural networks to enhance our understanding of machine learning and see how the DeepRacer learns the environment. Second limitation was the cost it took to train on AWS console. Another limitation encountered was the limited time in the semester to work on the project. We spent a majority of the time setting up the local environment for the DeepRacer and did not get it to work successfully.

Future work and milestones:

Our first goal for the future work for this project is to improve the accuracy and reliability of our reward function to allow the DeepRacer to make accurate turns on the track. Some ideas to satisfy this goal is to use the closest waypoints and waypoints parameters to determine where the DeepRacer should be heading next. Another possible addition is to add in a steering angle threshold so the DeepRacer does not turn too much or too little to go off track and reward the DeepRacer if it meets the threshold. Another goal for the future is to find an optimal training time. As read in one of the articles, too much training time can inhibit the DeepRacer from learning new changes and adapt to the changes in the environment and too little training time can make it difficult for the DeepRacer to make accurate and correct decisions. [6] The third goal for the future is to decide whether a simple or complex reward function would be better to train and allow the DeepRacer to have more consistent laps completed, more versatility when training on different tracks, and if it would help improve the lap times. The complexity of our reward function in the future will depend on if the DeepRacer does improve in any of the options above and how much it improves for each option. The fourth goal for the future is to improve the lap times for the DeepRacer. One way to achieve this would be to add more conditions using the speed parameter or adjusting the speed threshold throughout the training process. The fifth goal for future works is to try out more of the input parameters and see if they can improve our

current reward function. Some parameters that might help improve our reward function are using the steps parameter to enhance our progress section or using the steering angle values to find the optimal minimum steering angle. The sixth goal for future works is to optimize the hyperparameters to decrease training time or improve the quality of the data obtained from training the DeepRacer. The last goal we want to achieve in the future is to run our reward function and see the DeepRacer drive live.

List of References:

[1] https://towardsdatascience.com/an-advanced-guide-to-aws-deepracer-2b462c37eea
[2] https://codelikeamother.uk/planning-a-training
[3] https://youtu.be/rpMus-mj4fo
[4] https://github.com/scottpletcher/deepracer
[5] https://medium.com/axel-springer-tech/how-to-win-aws-deepracer-ce15454f594a
[6]https://medium.com/vaibhav-malpanis-blog/getting-started-with-deepracer-2020-edition-a7896dd07c48