Tim Komperda

Final Project

CSC580

Word Count:

Exploring DisCor Parameters in Hot Lap Racing in Assetto Corsa

Win on Sunday, sell on Monday is an old car manufacturing saying that described how winning in motorsports then leads to selling more cars. This led to manufacturers researching better technology which would then be proved on stages like the 24 hours of Le Mans. This led to the production of more efficient and reliable cars that could handle the harsh conditions of these chaotic conditions on the edge of grip while going as fast as possible. With such improvements happening due to the impact of racing, I think that autonomous driving should be a similar technology. There are many projects that try to solve the case of autonomous driving in a public environment, but very few that try to solve applications of autonomous driving in racing. This may be because many see self-driving in a normal case as much more valuable than obtaining an autonomous racing model. However, racing provides benefits that self-driving in a public environment does not. It provides a controlled environment that can benefit from iteration as racing is done on tracks where outside influence is low besides from track officials. It can be iterated by going from time trials which is a single car on a track going as fast as it can to grassroots wheel to wheel racing which allows for slower speed to be able to analyze how self-driving is reacting in multiple agent environments to endurance events where there are similar situations happening over long periods of time, but you have other cars that have different speeds, sizes, and cars can change in performance due to wear or change in conditions at any time.

However, track time and cars to run can get expensive, especially when using neural networks as neural networks take time to train and can perform maneuvers that can damage or even destroy the car. There is a cost-effective solution to this problem, simulation racing. Simulation racing has been around in some sense since 1982 when Namco developed Pole Position an arcade game that aimed to simulate Formula 1. This game did not have the in-depth physics simulation that modern simulators now have, such as tire wear, weather changing track conditions, weight transfer, etc. With that said modern simulations do a fairly good job of simulating reality and have been used to train autonomous driving models and one such paper has created an environment that can be used to compare and experiment: *A Simulation Benchmark for Autonomous Racing*

*with Large-Scale Human Data*. In this project, I will be exploring the different parameters of the reinforcement learning model used by the paper to see what parameters impact training and results.

**Preliminary information**

The model used in the paper is a version of a Soft Actor Critic (SAC) algorithm called Distribution Correction (DisCor). Soft Actor Critic is…

**Why do I care to do this project?**

I am performing this testing because in the homework 5 assignment that regarded reinforcement learning I struggled to understand why my Q-learning neural network did not learn well enough to produce results. This struggle really made me want to test a reinforcement learning neural network that was already trained and see what parameters effect the training process. My thought process was to test all the parameters, but that was not feasible in such a short period of time.

**Methodology of experiment**

I ended up with testing 4 different parameters of the Soft Actor Critic: discount factor, number of combined steps, target update coefficient, and error learning rate. I am using the BMW z4 gt3 on the Monza track and using the model given from the gymnasium project that was trained with the SAC with DisCor for 54 million steps. From there I trained the model for 1 million more steps using the given parameters which consisted of a discount factor of 0.992, number of previous steps of 3, target update coefficient of 0.005, and an error learning rate of 0.0003 and then tested each of the other tested parameters to see what makes a difference and what does not.

**Results**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Control** | **Gamma = 0.999** | **Gamma = 0.985** | **Nstep =5** | **Nstep = 1** | **Target Update Coefficient = 0.003** | **Target Update Coefficient = 0.007** | **Error Learning Rate =0.0006** | **Error**  **Learning Rate = 0.00015** |
| **Best Training Lap** | 1:52:789 | DNF | DNF | 1:52:872 | 1:55:773 | 1:52:565 | DNF | 1:53:195 | 1:53:743 |
| **Best Evaluation Lap** | 1:52:085 | 1:52:184 | 1:52:123 | 1:52:130 | 1:52:001 | 1:52:063 | 1:52:041 | 1:52:211 | 1:52:131 |

**Analysis**

*Discount Factor Experiment (Gamma)*

The gamma experiment was the one that I thought was the most obvious test to make this experiment also made the most sense as far as the results. The control gamma was the most optimal as expected. I will note that the experimental gammas were unable to complete a lap during the training, mainly struggling during the first chicane. But it was quite interesting to see that the lower gamma ended up performing better than the higher gamma.

*Number of Steps Experiment*

For this test, I wholly expected more steps to be better at connecting laps together. And this was indeed the case in the training example, the network managed to put up a lap that counted and was quick around the course. However, when it came to evaluation, more steps were detrimental to results. The network when trained on one step at a time managed to pull out the best result overall just 1 thousandth of a second above the 1 minute and 52 second mark. This network also seemed to perform the best around the chicanes, where other experiments lost some time.

*Target Update Coefficient Experiment*

This experiment was the most intriguing to me, the decreased target update coefficient performed the best in training and was quite quick and still improved in evaluation, but not by as much as other experiments, but it did lay down a better time than the control. The increased target update coefficient on the other hand could not get through the chicane at turn 1 during training. Every time it would just try to cut through the track and would get reset to the previous point which is what caused the DNF. In the evaluation though it put up the second-best time out of all experiments, as it was able to cleanly do that turn 1 chicane, although not as optimally as other experiments, so if it could train that chicane more, I’m certain it would break the 1 minute and 52 second barrier.

*Error Learning Rate Experiment*

For this experiment I asked ChatGPT what additional fields I should experiment with, and this was one of the responses where it recommended halving or doubling the original learning rate. And this experiment was not as exciting, as it was altering DisCor specifically, but it was interesting to see how that affected results. This ended up being the least successful in the evaluation portion, but it did manage to put novice level times in training.

Overall, the results of the experiments led that increasing the decreasing the target update coefficient in other words, slowing how much the target updates from iteration to iteration led to the best times when looking at training and evaluation combined. However, the number of steps when evaluating seemed to perform the best, I do wonder if this came down to a bit of randomness as all of the evaluation times have completed in better time then the Soft Actor Critic results described in the paper with this track and car.

**Future Ideas**

Things that could be done in the future is evaluating training with nothing but getting to higher number of steps like was done in the original paper with 70 million steps rather than the small number of steps that I completed. I also would like to see if I could get the model to learn to drive a track without a preliminary track layout like in the original paper. Other ideas I have are if I could train on a single human run and translate that data as a layout of the track even if the run is not optimal.

**Conclusion**