Schutt & Fetterolf

12SEP2019

MV4025 Final Project Write Up

**Early Stages**:

We started off by running several iterations of the starter code. As expected, the results were poor, yielding a running avg reward of 0.0189.

Our baseline settings were as follows:

Run time/training duration = 48000

Discount factor = .99

Hidden layer type = tanh

Test duration = 100

Learning rate = .05

Reward timeout = 20

Seed = 557935545

Num hidden layers = 80

Respawn with = 15

Loss factor = .2

**Approach**:

We noticed that the only reward function was passing rewards based solely on the entity’s distance to target. We modified the RewardDistToTarget method to incorporate other methods of rewards like the kills minus losses and also a reward for distance from a friendly entity. We set up our code in such a way that the kills-loses factor was more effective than the distance from a friendly entity.

Our thinking behind the incorporation of these two reward factors was to provide a method of measuring our status of “winning”. A higher number of kills correlates to a closer status to winning. The concept of rewarding entities for fighting closely to other friendlies was based on the idea that two entities attacking an enemy together would be more efficient and effective than one. While this may not be doctrinally sound, it is our belief that studying this metric will hopefully shed some light on better approaches.

While being close to each other is not guaranteed to increase effectiveness nor is it something that always makes sense tactically, the fact that the terrain is not varied in this simulation (it is actually flat in this case) should yield positive results.

Next Iteration Params:

Run time/training duration = 48000

Discount factor = .99

Hidden layer type = relu

Test duration = 100

Learning rate = .05

Reward timeout = 20

Seed = 557935545

Num hidden layers = 80

Respawn with = 15

Loss factor = 1.1

We had some major issues when trying to implement Relu as our hidden layer type. For some reason, Relu produced various index out of range exceptions. This is something we attempted to troubleshoot further, but ultimately ended up abandoning Relu for tanh because it was a proven algorithm. After making this adjustment we continued our testing.

**Best Results:**

We implemented the RewardCloseToTarget() method into our reward function. When viewing previous iterations without it, we noticed entities straying away from the target. By implementing a reward for distance from target, the idea was that this would prevent straying away from the target. We conducted multiple iterations with RewardCloseToTarget() implemented. Our best results came from the following parameters:

Run time/training duration = 24000

Discount factor = .99

Hidden layer type = tanh

Test duration = 100

Learning rate = .03

Reward timeout = 40

Seed = 38072947

Num hidden layers = 60

Respawn with = 15

Loss factor = .8

The parameters above yielded a running average reward rate of .0036. While more testing might yield higher results, we feel that this is a decent increase in the reward rate from the starter code. When viewing the simulation from a top down view, the friendly entities appear to work effectively when projecting fires on the target.

**Conclusions**:

We found this project to be extremely difficult due to sheer amount of parameters that need to be modified to yield meaningful results. Much like our Introduction to artificial intelligence (AI) class projects, finding a starting point was the most difficult. The ability to bash and test multiple parameters at one times was very useful, however very computationally expensive and sometimes unreliable. While we did face some adversity in this project, we felt that we were able to takeaway some valuable insights about the use of the Unity3d game engine, as well as the AI process.