Literature Review

Overview

In this literature review I will be looking at Multi Agent Reinforcement Leaning (MARL) and how it can be used in the context of strategy optimization for the matched sprint race. MARL is where we have multiple agents interacting with a common environment.

Competitive self-play has showed the most practical promise in terms of self-play algorithms so for this literature review I will limit my scope to self-play.

Examples of MARL

These are some of the popular examples of self-play. They can mostly be split into 2 versions, model based and model free. Since our case is model free, (in the sense we can’t do a tree search), we will focus on the model free applications. These are listed in chronological order.

The first example is from (Samuel, 1959)where they develop an agent that performs at an expert level in checkers. The agent could choose the action based on a value function trained on samples, a tree search and a manual look up table. The samples were gathered through human games and self-play. This is an example of a model-based approach.

Then (Tesauro, 1994) developed a backgammon agent that achieved superhuman ability. A neural network evaluated the value function of different board position. The agent was trained purely using self-play, using the TD() algorithm (Sutton, 1988). This was combined with a tree search to choose the optimum action, again this is a model-based approach.

Deep mind has had a lot of success using self-play develop agents that can perform at superhuman levels at games like Go, Chess and Shogi (Silver, Hubert, et al., 2017). Initially AlphaGo mastered Go (Silver et al., 2016), however required initial supervised learning. This constraint was removed with AlphaGo Zero (Silver, Schrittwieser, et al., 2017)which was trained purely from self-play. Finally, AlphaZero (Silver, Hubert, et al., 2017) is a general algorithm that can work on many different domains. All these algorithms are model based and rely on a very sophisticated Monte Carlo Tree Search, which looks ahead to find the optimal actions.

INDEPENDENT RL OpenAI trained agents that performed in a 3D environment with simulated physics, with tasks like taking and saving penalties. They used the Proximal Policy Optimization algorithm (Schulman et al., 2017) with self-play and an initial exploration reward.

OpenAI also trained an agent to play Dota 2 (OpenAI et al., 2019)which highly dimensional online game using a very similar approach (self-play and PPO) but did require 10 months to train the algorithm and lots of hand-crafted rewards.

Finally, AlphaStar was developed which mastered the game of StarCraft and uses supervised learning for the initial policy and then self-play. The novel self-play opposition selection approach is developed. This is by far the most complex system.

Out of these examples the two OpenAI provide the most applicable techniques that could be used for my project. Firstly, they are both model-free techniques and secondly, the approaches they take are simple and so be realistic for me to implement. The 3D environment is most similar in terms of dimensionality so hopefully a similar approach will work for my environment. The Dota 2 system also includes lots of techniques that could be useful if I have issues. The Deep mind systems are too large and complex for my environment, and it is unrealistic to implement anything similar, however, the self-play techniques could prove useful.

Theoretical Basis

MARL can be split up into 3 catagories, cooperative, mixed and competitive. Our environment is primarily competitive although more realistically would be mixed as both cyclists want to prevent crashes.

In single agent RL the environment can be modelled at a Markov Decision process (MDP) and this is the key assumption that most RL algorithms rely on. In the multi-agent case we have non-stationarity so the agent is faced with a moving target learning problem, the best action changes as the other agent changes. Another challenge with MARL, is that the learning goal is not clearly defined. Nash Equilibrium is commonly used, that characterizes a point that, given the algorithm converges, the different agents will not deviate from. Hower, the assupitions that this is based on are often not applicable in practilcal MARL algorithns. (Zhang et al., 2019)

There are many algorithms that come up with solutions to MARL ((Bus¸oniu et al., 2010) ,( Zhang et al., 2019). Most of these algorithms presented in the paper are theoretically based and attempt to find this NE, however, many of these attempts provide limited results in very specific environments.

Using single agent mehtodes can be used for MARL. In Dota 2, both agents are being trained simultansioly, and this dos manage to produce good results, despite this method clearly breaking the stationarity assumptions on which single RL algorithms rely.

Fictious self play ###insert reference is an idea form game theory. If you have 2 players who play against the best stategy to the average strategy of the oppentent iterativly, then this will converge to the NE, given a strict set of critieria. this is the idea behind competitive slef-play, . The current agent plays against a randomly selected previous agent. How the opponenent is selected differs a lot between examples and is mentioned more later.

Another way to explain self-play is that the aim is to find the policy that will maximise the expected return against the entire set of other policys. This intergral can be approximated by self-play in which it plays against policys that dominate the policy space. ##ref

The idea of competitive self-play is a method that has shown good empirical results and makes intuitive sense and it is only recently that there has ben an attempt to provide a theroretical basis for it ##ref.

Features of a MARL system

Self-Play algorithm

In the framework se out by (Hernandez et al., n.d.), it refers to a menagerie of potential opponent agents, a probability function over that menagerie that decides which opponent will be played next and a function that decides which agents will be added/removed from the menagerie. Here are a few examples that I have come across:

Naive self-play: Opponent is always the most recent agent

δ-Uniform Self-Play/ Fictious self-play: All previous agents are in the menagerie and have a random change of being selected.

There is lots of room of different alorithms that could be used and would have to be tested so find the most suitable solution. These are the fundamentals of what has been used in previous MARL examples.

Dota2 (OpenAI et al., 2019): Uses 80% naïve self-play then 20% uniform.

AlphaStar (Vinyals et al., 2019): Uses many leagues but in main league uses novel algorithm, Prioritized fictitious self-play where stronger opponents have a higher chance of competing.

AlphaGo Zero(Silver, Schrittwieser, et al., 2017): Uses niave self-play but has a avaluator on the current policy so is only used as the new policy if it beats the previous best.

Action Space, Observation space

For high action space dimensionality games, there is the option of adding “scripted actions” in for action sequences as a way to reduce complexity. This is the approach (OpenAI et al., 2019) took. Intitally, scripting most actions and then removing some as the program was further developed. This simplifies the program as the agent does not have to learn the action sequence but could cap the agent’s potential. AlphaStar (Vinyals et al., 2019) included no scripted actions. Our scripted actions could be related to steering.

Simplifications could be made in the observations space and then loosened with further development.

Exploration

Exploration is needed to ensure that the full state space is explored, and the policy does not converge on a sub-optimal minimum. There are three main approaches that are used: random initialisation, using an entropy factor and exploration rewards. The random initialisation was commonly used by most of the algorithms (Bansal et al., 2017; OpenAI et al., 2019)(even though the random initialisation might not be random in the actual game.) The entropy factor in the loss function encourages actions which are less common.

Exploration rewards are used by (Bansal et al., 2017; OpenAI et al., 2019; Vinyals et al., 2019). In the AlphaStar case they are from supervised learning. These rewards were used as the sparse reward at the end makes it hard for the agent to learn. However, these introduce a bias into the agent policy so should be avoided if possible. Even with rewards for small actions the policies still managed to find novel long term strategies, so the length of the exploration also seems important.

Neural Networks

There is a choice wether to use a MLP or a LSTM for the NN. LSTM works well for time series data but is more complex. (Bansal et al., 2017) uses a mixture of the two for different applications.

Type of RL algorithm

The most used RL algorithm is PPO and this, or a variation of it, has been used in all the examples mentioned. There are other options though that could be used.

Software

OpenAI’s gym environments (*Gym*, n.d.) are the most used environments and easy to use. However, they do not support MARL, however, there have been modifications that have been made that converts ordinary gym environments so they are compatible with MARL (*Making a Self-Play Environment for OpenAI Gym | by Isamu Isozaki | Analytics Vidhya | Medium*, n.d.).

Raw Notes