Deep Reinforcement Learning for StarCraft II Battles

Jerome Kafrouni jk4100
Roop Pal rmp2191
Connor Hargus cjh2209

Abstract

We lay out a brief history of the prior work in reinforcement learning for real-time strategy games and propose our own project working with StarCraft II's SC2LE interface. Specifically, we intend to develop and improve on a policy gradient algorithm such as COMA used for two SC2LE minigames involving battles between groups of agents.

1. Introduction

Deep reinforcement learning has made significant strides in recent years, with results achieved in board games such as Go. However, there are a number of obstacles preventing such methods from being applied to more real-world situations. For instance, more realistic strategic situations often involve much larger spaces of possible states and actions, an environment state which is only partially observed, multiple agents to control, and a necessity for long-term strategies involving not hundreds but thousands or tens of thousands of steps [3]. It has thus been suggested that creating learning algorithms which outperform humans in playing real-time strategy (RTS) video games would signal a more generalizable result about the ability of a computer to make decisions in the real world.

Of the current RTS games on the market, StarCraft II is one of the most popular. The recent release by Google's DeepMind of SC2LE (StarCraft II Learning Environment) presents an interface with which to train deep reinforcement learners to compete in the game, both in smaller "minigames" and on full matches [4]. While current unsupervised reinforcement learning algorithms are no where near matching human performance in playing full matches, they are able to make some progress on the minigames. The 7 minigames included in SC2LE set up environments for achieving various essential components of SC2 gameplay, such as gathering resources, fighting small battles, and building new units. For our project, we hope to make improvements on current models in the combat minigames.

The two minigames which we are concerned with for this project are "DefeatRoaches" and "DefeatZerglingsAndBanelines". In each case the player is given a set of units which it must use to defeat a set of enemy units. Each time the set of enemy units is defeated, the player is given additional units. A reward of +5 or +10 (depending on the minigame) is given for defeating each enemy unit and a reward (penalty) of -1 is given for each of the player's units which is lost.

2. State of the Art

In releasing SC2LE, Google's engineers also tested a policy gradient method with parallelization as laid out by Mnih et al.'s Asynchronous Advantage Actor Critic (A3C) in combination with a set of modern deep architectures to establish baselines for performance on the included minigames. The neural network architectures they used included Atari-Net [1], a fully convolutional network (predicting actions without reducing resolution at each layer), and a fully convolutional network with long short-term memory. While the latter of these models tended to perform the best across the set of minigames, they all fell dramatically short of StarCraft GrandMaster performance on the tasks we consider here. In "DefeatRoaches", the highest mean episodic reward achieved by any of Google's architectures was 101, while that of a GrandMaster was 215. "DefeatZerglingsAndBanelings" proved significantly harder, with the highest mean score of the RL algorithm at 96 while GrandMaster players averaged 727. The difference in performance capabilities between these two tasks likely stems from the fact that the second, in which the player must defeat two different types of units, demands the use of different strategies and ablities depending on the enemy unit being faced.

In the October 2017 paper Deep Learning for Video Game Playing, Justesen et al. describe the unique challenges of RTS games and varying approaches utilized for controlling multiple agents [3]. Some algorithms, such as Intrinsic Curiosity Module (ICM) and Independent Q-Learning (IQL), treat all units as independent individuals. This assumption allows for dramatic simplifications in the necessary complexity of the model for taking actions, but it clearly lacks much of an ability to utilize the advantages which come from group agent strategy. They also describe how Multiagent Bidirectionally-Coordinated Networks (Bi-

CNet) have shown promise for controlling large numbers of units, but that for small skirmishes of fewer than 10 units (similar to the minigames we intend to tackle) that Counterfactual Multi-Agent (COMA) methods using policy gradient are the state of the art. This method has recently been applied to the original StarCraft: Brood Wars and is described in depth in a December 2017 article by Foerster et al. [2]. The core feature of COMA methods is that a centralized critic evaluates the decentralized policies of multiple agents, striking a better balance between independence of agents and high-level strategy.

3. Our Approach

We intend to begin our project by emulating the most promising of the methods described above. To start, we want to match Google DeepMind's preliminary benchmark mean scores on the minigames "DefeatRoaches" and "DefeatZerglingsAndBanelings" by developing a policy gradient algorithm using a fully-connected convolutional network with LSTM. Because the COMA model is also a policy gradient method, we then hope to apply the COMA algorithm and test whether we are able to achieve significant gains in average reward. We suspect that COMA's decentralized policies may perform better than Google's method on the latter of these minigames, where the strategies may require higher-level coordination. We are also interested to know how COMA it will scale to a number of units larger than 10. If time permits, we may also want to draw on the large set of matches played by master StarCraft players. Our ability to use this data will rely on having some way to extract the snippets relevant to the minigames we are working on, however, so it is possible this information would only be useful were we to try to take on the full game.

4. References

References

[1] M. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An evaluation platform for general agents. J. Artif. Intell. *Res.(JAIR)*, 47:253 279, 2013.

[2] J. Foerster, T. Afouras, N. Nardelli, G. Farquhar, S. Whiteson. Counterfactual Multi-Agent Policy Gradients. *arXiv preprint arXiv:1705.08926v2*, 2017.

[3] N. Justesen, P. Bontrager, J. Togelius, R. Sebastian. Deep Learning for Video Game Playing. *arXiv preprint arXiv:1708.07902v2*, 2017.

[4] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev. et al. StarCraft II: A New Challenge for Reinforcement Learning. *Google DeepMind*, 2017.