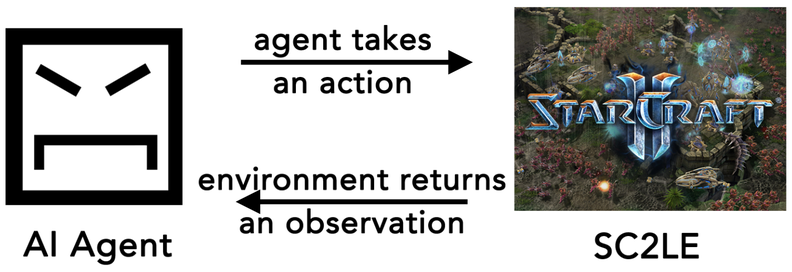
Reinforcement Learning with StarCraft II

What is Starcraft 2?

Starcraft 2 is a real-time strategy (RTS) game and the 2010 sequel to the original Starcraft released in 1998. The multiplayer aspect of the game involves players competing by building armies and controlling them to overwhelm the opponents.

Actual game dynamics can be extremely complex with several parallels to real-world battles, such as scouting, ambushes, flanks and other tactics. On top of that, players have to balance time and resources between their economy and their military, with different strategies such as 'rushing' or 'turtling'.

The SC2LE Environment



In August 2017, fresh from the victories of AlphaGo, DeepMind released the Starcraft 2 Learning Environment (SC2LE), in collaboration with Blizzard Entertainment.

The Python component of SC2LE is known as PySC2, which exposes the Starcraft 2 API as a Python environment for reinforcement learning (RL) agents.

Essentially, this allows an AI agent to take an action in the environment. The environment then returns an observation as a result of the action taken. The AI agent then takes a new action based on the observation. This is similar to a very complex game of chess or Go, where a player makes a move based on what they see and the board changes due to the move.

The Challenge

The multiplayer aspect of Starcraft 2 as played by humans is extremely complex. Players have to mine resources and build buildings, while defending against attacks and grow an army, in order to defeat their opponents. Each aspect has to be balanced and serve as parts of a bigger strategy that counters the opponents'.

In terms of reinforcement learning, the game possesses several challenges in addition to its complexity. A single game can take tens of thousands of game steps / frames, which means that reward is sparse and heavily delayed and early actions can have important delayed consequences. Moreover, the game is partially-observed, which means that RL agents must use a combination of memory, planning and inference. The action and state spaces are both tremendous, with approximately 100 million possible actions at any step, given a compressed screen size of 84x84 pixels.

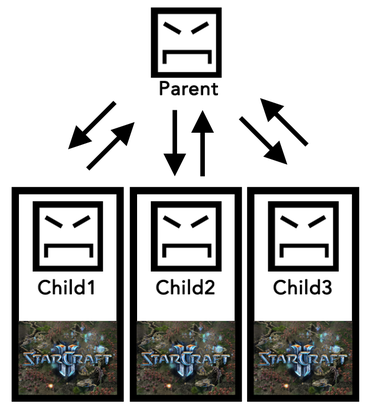
The Project

Since DeepMind did not release any of their agents, this project focuses on coding agents that replicate the performance of DeepMind's agents as reported in their paper and if successful, release these agents as open-source code.

A3C Atari-net Agent

We first attempt to implement the A3C algorithm for SC2LE. Arthur Juliani has an implementation of A3C for the VizDoom environment, which was used as reference for the A3C SC2LE agent.

The crux of the algorithm involves simultaneously running multiple child agents in the environment, with a central parent agent updating its policy and value networks based on the rollouts from the children. The children then periodically download the latest network parameters from the parent.



Subsequently, we reconstruct the Atari-net architecture described in DeepMind's papers, for the policy and value networks.

In the Atari-net architecture, the spatial feature layers (state and minimap) are passed through convolution networks and a final fully connected network, whereas nonspatial features are passed through regular fully connected networks. The outputs are concatenated to form the state representation vector, which is fed into the policy and value networks. The policy network then generates the next action to be taken, while the value network generates value predictions.

DefeatRoaches

We begin with a minigame supplied by DeepMind for training of RL agents. This minigame involves the agent controlling 9 Marines against 4 enemy Roaches. Whenever all 4 Roaches are defeated, a new batch of 4 Roaches spawn again and the agent receives 5 additional Marines, with remaining Marines retaining their existing health. The agent gains +10 score for every Roach defeated and -1 score for every Marine defeated.

Results

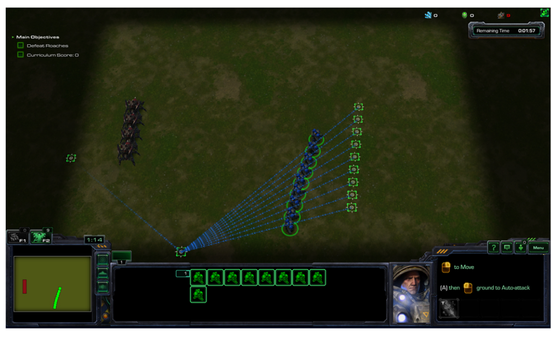
Due to limited hardware available, the results presented here is only after a limited number of game steps / episodes, which means that the agent's performance is still improving.

		DeepMind Atari-net 600 million steps	Our Agent 50 million steps
DefeatRoaches	Max Score	351	338
	Avg. Score	101	65

Observations

Angle of Attack

In the more recent episodes, the agent begins to learn how to approach the enemy Roaches in a more optimal manner. The agent learns to send Marines from the bottom instead of head on. It is observed that doing so forces the enemy Roaches to reposition themselves in order to attack the Marines. While the further Roaches are repositioning themselves, the Marines are able to focus fire on the nearest Roach.



Stalling for Time

It is interesting to see that in some cases the agent seems to try to stall for time. For example, if the agent has only a single Marine left, the agent will move it randomly around the map. While this appears to be a strange strategy, if the lone Marine cannot defeat the remaining Roaches, it makes sense to run away to preserve the Marine's life and prevent the -1 to the score, while waiting for the 2-minute timer to run out.

Future Work

The code for the agent described here is already available on the GitHub link below.

We have also recently generalized the code to work on all seven minigames and we will complete the documentation of the agent's performance on all minigames.

Work is also in progress to replicate the more successful FullyConv, FullConvLSTM and arFullyConv architectures described in DeepMind's paper.

References

Vinyals, Oriol, et al. "Starcraft II: A new challenge for reinforcement learning." arXiv preprint arXiv:1708.04782 (2017)

https://github.com/deepmind/pysc2

Credit goes to Arthur Juliani for his implementation of A3C for the VizDoom environment

https://github.com/awjuliani/DeepRL-Agents

Image credits to Blizzard Entertainment.

More Information

For more information or collaboration, feel free to contact me at my email: limsweekiat@gmail.com

To keep up to date on the progress of this project, check out the project repository at:
https://github.com/greentfrapp/pysc2-RLagents