COMPUTATIONAL RESEARCH PROGRESS IN APPLIED SCIENCE & ENGINEERING (CRPASE)

CRPASE: TRANSACTIONS OF INDUSTRIAL ENGINEERING

Journal homepage: http://www.crpase.com

CRPASE: Transactions of Industrial Engineering 8 (3) Article ID: 2487, 1–5, September 2022 ISSN 2423-4591

Research Article

Towards Interpretable Reinforcement Learning in Real-Time Strategy Games

Lance Bae*

Bergen County Academies, New Jersey, United States

Keywords

Explainable Reinforcement Learning, Real-Time Strategy Games.

Abstract

In recent years, Deep Reinforcement Learning (DRL) has become an increasingly popular method of creating effective intelligence agents. DRL agents have proven to be successful, especially in the realm of games, but we as humans have difficulty understanding DRL agents' behavior due to their complex structures. Uncovering game-playing DRL agents' priorities and action patterns can reap valuable insights into how humans can effectively manage real-world game-like environments. One genre of games that might be of particular interest would be Real-Time Strategy (RTS) games, which involve many real-world aspects such as simultaneous management of multiple units and real-time decision making. In this paper, we introduce the method of using Decision Tree Classifiers to better understand and visualize the behaviors of DRL agents in the RTS environment, gym-µRTS.

1. Introduction

As Deep Reinforcement Learning (DRL) methods have become an area of intense study in recent years, more and more effective DRL agents have been birthed. The realm of games in particular has been well-populated with DRL agents: a notable example would be AlphaGo, the first computer program to defeat a professional Human Go player [1].

Although these DRL agents have proven to be effective, human researchers have had difficulty uncovering their priorities and strategies due to their complex structures as Neural Networks. DRL agents have appropriately been dubbed "black boxes" due to the difficulty of being able to understand their inner workings. Explainable Reinforcement Learning is an area of study that aims to tackle this issue by studying the behaviors and action patterns of DRL agents. Doing so may yield fruitful insights into newer and more effective strategies for humans to adopt.

In this paper, we present a potential method in studying DRL agents in the Real-time Strategy (RTS) game µRTS through its Python wrapper, gym -µRTS [2]. Gym-µRTS is an environment consisting of a 16×16 grid and incorporates the two key aspects of an RTS game: 1) the simultaneous management of multiple units and 2) real-time decision making. Figure 1 displays a sample gym-µRTS game state.

The agent we study in this paper is COAC, the 2020 µRTS AI Competition champion microRTS AI Competition [3]. We look to uncover COAC's strategies as it plays µRTS, as RTS games' features of simultaneous management and real-time decision making are crucial in real-world situations. The ability to model how world- champion DRL agents play µRTS may lead to strategies applicable to the real world, such as in the controlling of drones.

* Corresponding Author: Lance Bae
E-mail address: lance.y.bae@gmail.com, ORCID: https://orcid.org/0000-0001-7735-3435

Received: 15 August 2022; Revised: 23 August 2022; Accepted: 5 September 2022

https://doi.org/10.52547/crpase.8.3.2487

Academic Editor: **Mohammad Mahdi Ahmadi**

Please cite this article as: L. Bae, Towards Interpretable Reinforcement Learning in Real-Time Strategy Games, Computational Research Progress in Applied Science & Engineering, CRPASE: Transactions of Industrial Engineering 8 (2022) 1–5, Article ID: 2487.

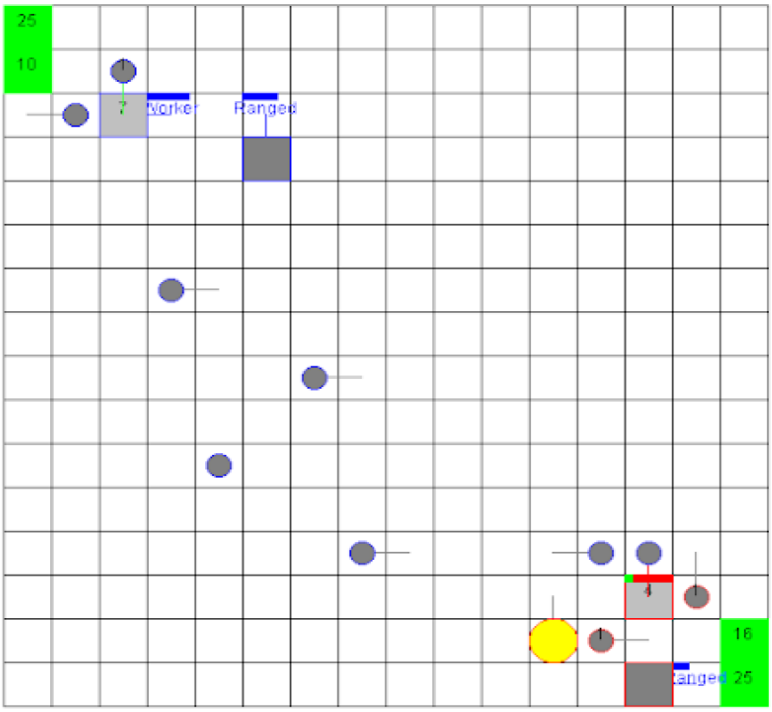

Figure 1. Sample μRTS Game State

2. Related Work

Many methods have been previously proposed to advance the ability to understand DRL agents [4] such as:

- Programmatically Interpretable Reinforcement Learning: The policies of DRL agents are recreated but in a high-level human-readable programming language [5].
- Hierarchical and Interpretable Skill Acquisition: Agents are designed to create higher-level policies that consist of lower-level actions; the higher-level policies are described by a human instruction [6].
- Model U-Trees: Linear Model U-Trees (LMUTs) are essentially the same as Continuous U-Trees, but each leaf node represents a linear model instead of a constant; LMUTs are able to efficiently represent continuous functions, which allow them to effectively imitate the Q functions of DRL models [7].
- Representation via Causal Graphs: Causal graphs are directed acyclic graphs constructed using regression learners and help explain why a certain action are taken by considering the counterfactual [8].

3. Methods

In order to model how COAC played μRTS, we trained a Decision Tree Classifier (DTC) for each cell on the 16×16 game board. Given the state of the game board, each DTC would then predict what the cell's action would be.

3.1. Decision Tree Classifiers

Throughout this project, we choose to use DTCs to predict a cell's action given the state of the board. DTCs are full binary trees where each inner node has an inequality condition based on a single feature of the input data: if the condition is true, the train of logic falls to the left child, and the train of logic falls to the right otherwise.

Each leaf node represents a prediction.
DTCs are advantageous because they have a structure that is easy for humans to understand, the process of following their logic is rather simple. DTCs can also be easily visualized for human interpretation, as shown in Figure 4. The DTCs we used, as provided by scikit-learn [9], can also be easily limited in size to prevent overstating and too complex of a model.

However, a downside of DTCs is their volatility. The structures of DTCs can be completely altered due to a few outlier data points, which is certainly not favorable.

3.2. Collecting and Reshaping Data Points

While COAC played against various other agents, we collected, for every action, 1) the current game state (16 × 16×27) and 2) an action array detailing the action taken and at which cell (1×8). A total of 59,335 data points were collected, with Figure 2 displaying a heatmap of the number of data points collected for each cell.

Each cell in a game state was described by a length-27 array based on the formatting detailed in Table 1.

So, for example, a cell containing a moving worker owned by Player 1 that has Hit Point and 2 Resources would be described as the concatenation of its one-hot encoded features [2]: [0, 1, 0, 0, 0] [0, 0, 1, 0, 0] [0, 1, 0]
[0, 0, 0, 0, 1, 0, 0, 0] [0, 1, 0, 0, 0, 0] → [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0].
Although one-hot encodings are ideal for DTCs, there was no need for the Hit Points and Resources features to be one-hot encoded since - for these two particular features - the notion of comparing via inequalities is reasonable.

Additionally, compressing these one-hot encodings eliminates extraneous features which may have resulted in extra layers for our DTCs. Thus, our one-hot encoded Hit Points and Resources were adjusted into single integers, yielding a new cell feature array of length 19.

As for the action arrays, as detailed in Figure 2, we only care for the Source Unit and Action Type features for now, so our action arrays were shortened to be of length 2.

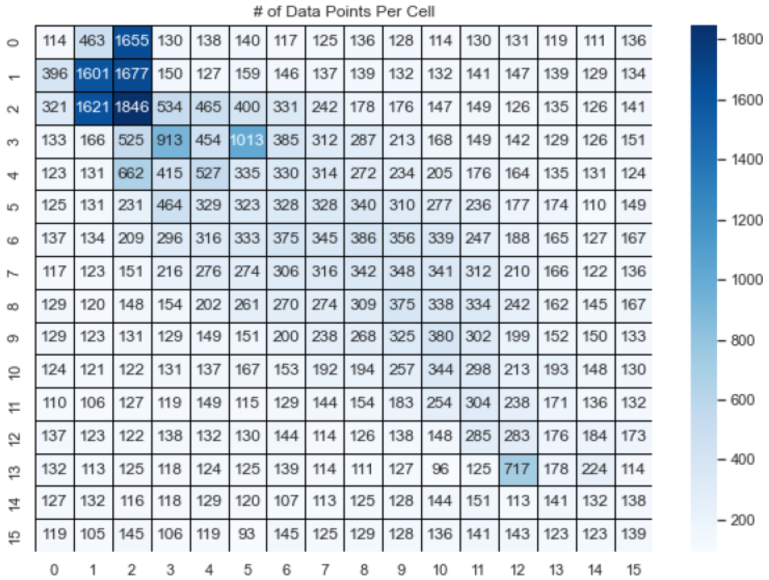

Figure 2. # of Data Points per Cell

Table 1. Cell Observation Features

Observation Features	Planes	Description
Hit Points	5	0, 1, 2, 3, ≥ 4
Resources	5	0, 1, 2, 3, ≥4
Owner	3	-, player 1, player 2
Unit Types	8	-, resources, base, barrack worker, light, heavy, ranged
Current Action	6	-, move, harvest, return, produce, attack

3.3. DTC Training

With all of the data now collected, the next step was to train our DTCs for each cell. This was done by first assigning each data point to its appropriate cell, then hypertuning each cell's DTC's max depth from 3 to 15 using a grid search cross validation.

4. Results and Discussion

Figure 3 displays each of our cells' DTCs' test accuracies. Although our test accuracies appear to be relatively low for the most part, there is a clear special relation between the number of data points collected (Figure 2) and the DTC test accuracy (Figure 3) along the grid's diagonal. Thus, we are hopeful that the collection of more data points especially in the areas that seem to be less populated-will lend itself to higher overall test accuracies.

Table 2. Action Components

Action Components	Range	Description
Source Unit	[0, h × w - 1]	the location of the unit selected to perform an action
Action Type	[0, 5]	NOOP, move, harvest, return, produce, attack
Move Parameter	[0, 3]	north, east, south, west
Harvest Parameter	[0, 3]	north, east, south, west
Return Parameter	[0, 3]	north, east, south, west
Produce Direction	[0, 3]	north, east, south, west
Parameter Produce Type	[0, 6]	resource, base, barrack, worker, light, heavy ranged
Relative Attack Position	[0, a_r^2 - 1]	the relative location of the Position unit that will be attacked

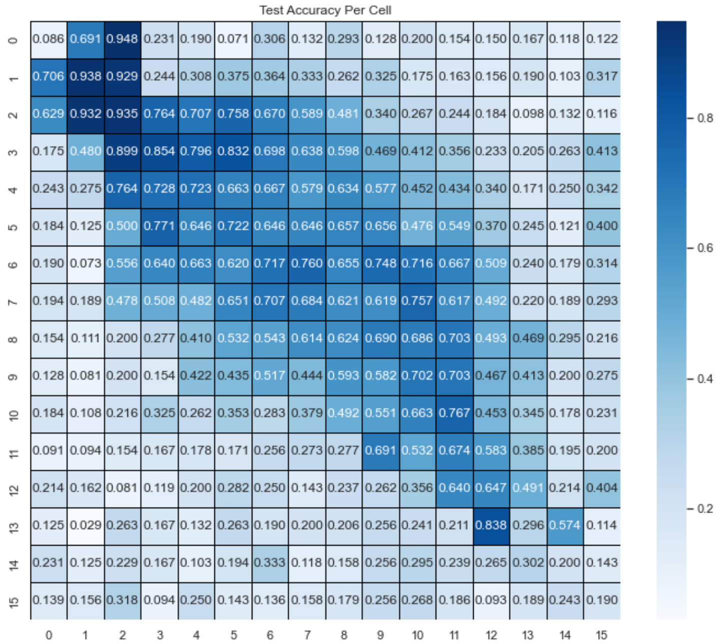

Figure 3. Test Accuracy per Cell

4.1. Sample DTC

Figure 4 displays the DTC used for the cell in row 2, column 2, which had 1,846 data points collected (the most of any cell) and a test accuracy of 93.5%.

Looking at this tree, it's very easy to follow the model's decision process: if the condition at the top of each box is true, the train of logic falls to the left, and if it's false, the train goes to the right. Once the train of logic reaches a leaf node, a prediction is made. This sample DTC's visual simplicity and straightforwardness are precisely why we propose the use of DTCs in our future pursuits of modelling µRTS agents' decision-making processes.

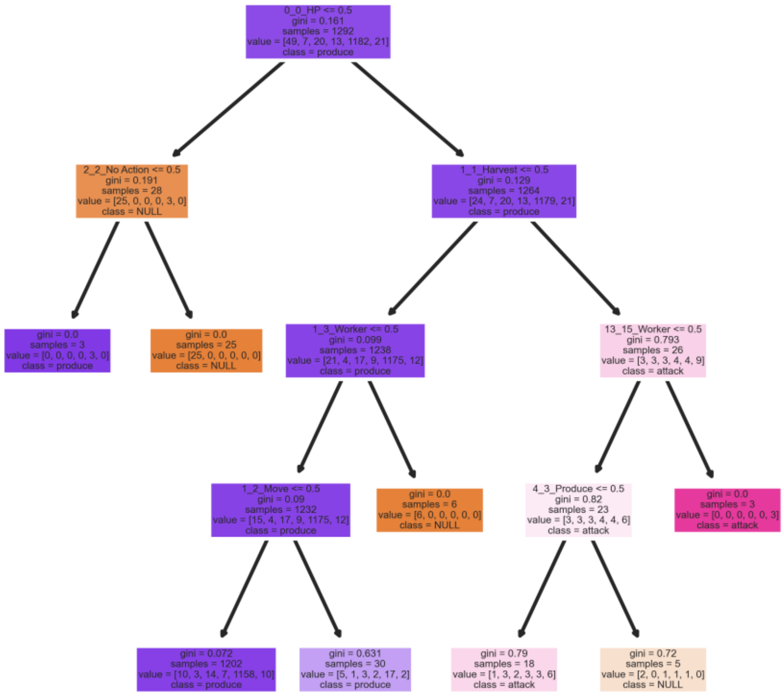

Figure 4. Decision Tree Classifier for Cell (2, 2)

5. Conclusion and Future Work

If it is not already apparent, this research project is still young, and there is a lot of work to be done. However, these preliminary results are enough to get us excited and assured that we are on the right track. As of right now, our (chronological) future plans consist of:

- Collecting more data from COAC. As previously stated, there appears to be a clear correlation between each cell's number of data points and its DTC's test accuracy. Thus, it is only natural for us to aim to collect more data points. However, there are more data points in certain regions for a reason: the game starts with COAC's agents in the top-left corner, and it appears that COAC tends to move its units along the diagonal of the board. Therefore, we will work towards manually placing COAC in uncommon situations by manipulating the game state: this will allow us to collect more data points from less popular cells.
- Hard-coding rules into our DTCs. Right now, our DTCs do not appear to be taking a cell's unit into account. However, it would make sense for a DTC, for example, to automatically eliminate some action choices for a “Barrack” unit since they can only stay idle or produce. We aim to hard-code these decisions into the DTCs so that they are able to immediately make “obvious” choices.
- Creating a GUI. Once our DTCs are all able to perform at a sufficient level, we will aim to create an interactive GUI for outsiders to easily understand how the COAC agent behaves, thus ultimately achieving our goal of creating a human-understandable model that is able to present how the COAC agent behaves.

Acknowledgments

I thank Yanming Zhang (Stony Brook University) and Dr. Klaus Mueller (Stony Brook University) for mentoring me and allowing me to partake in this research project.

References

[1] DeepMind, Alphago. (2022). https://www.deepmind.com/research/highlighted-research/alphago.
[2] vwxyzjn. gym-microrts-paper, (2022). https://github.com/vwxyzjn/gym-microrts-paper. GitHub.
[3] microRTS AI Competition. 2020 cog results, (2020). https://sites.google.com/site/micrortsaicompetition/microrts.
[4] E. Puiutta, E. M. S. P. Veith, Explainable reinforcement learning: A survey. In A. Holzinger, P. Kieseberg, A. M. Tjoa, E. Weippl (Eds.), Machine learning and knowledge extraction. Springer International Publishing, (2020).
[5] A. Verma, V. Murali, R. Singh, P. Kohli, S. Chaudhuri, Programmatically interpretable reinforcement learning, (2018). DOI: 10.48550/ARXIV.1804.02477
[6] T. Shu, C. Xiong, R. Socher, Hierarchical and interpretable skill acquisition in multi-task reinforcement learning, (2017). DOI: 10.48550/ARXIV.1712.07294
[7] G. Liu, O. Schulte, W. Zhu, Q. Li, Toward interpretable deep reinforcement learning with linear model utrees, (2018). DOI: 10.48550/ARXIV.1807.05887
[8] Madumal, P., Miller, T., Sonenberg, L., & Vetere, F. (2019). Explainable reinforcement learning through a causal lens. DOI: 10.48550/ARXIV.1905.10958
[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, E. Duchesnay, Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12 (2011) 2825–2830.