Project: Combat Micromanagement with Reinforcement Learning and Curriculum Transfer Learning

ECS170, Spring 2018

Group Members

Robert Park, rbpark@ucdavis.edu, 913033164
Xingwei Ji, xwji@ucdavis.edu, 914401326
Adrian Wang, atwang@ucdavis.edu, 914167916
Jordan Palmer, jspalmer@ucdavis.edu, 914959425
Rishi Subramanian, risubramanian@ucdavis.edu, 999327988

Problem and Contribution Statement

Playing games such as Starcraft II involves performing several different tasks. Creating separate models to perform each of these is difficult and time consuming, and is a major obstacle in the creation of such programs. However, much of this work could be saved since many of the tasks performed share some similarities. We propose to exploit these similarities by applying transfer learning to the problem of unit micromanagement. The choice of a machine learning approach allows for the automatic transfer of skills since the algorithm can learn different scenarios with minimal manual intervention.

The application of transfer learning to Starcraft was recently attempted by Shao et al. in April 2018. They trained a deep reinforcement learning model with a custom training algorithm, PS-MAGDS, and applied it to several small combat scenarios in Starcraft: Brood War. They then increased the size of the scenarios to transfer the knowledge gained. In both cases, the models performed favorably compared to human players and other AI approaches.

Our first step is to reimplement the aforementioned algorithm for use with Python (the original is in C++) and Starcraft 2. This in itself will be a contribution to the Starcraft AI team at UC Davis, as well as the larger Starcraft 2 community. This will allow easier inspection and modification of the specific techniques used in the future. We hope that the use of the newer game will also increase interest in the community.

While the original work was focused on combat micromanagement, we also plan to extend it by applying it to a wider range of scenarios. For example, we intend to see how combat skills can be transferred to resource collection and production optimization rather than the original goal of applying the same combat skills to larger variations of the same task.

Since we will be working with Starcraft 2 rather than Brood War, we see the need to adapt and extend parts of the original paper to potentially increase performance. These include the state representation and the analysis of the strategies used. The open-ended nature of this portion will allow us to adjust for the time we have remaining. This work can be integrated with the efforts of the Starcraft AI Team even after the completion of this project.

Shao, K., Zhu, Y., & Zhao, D. (2018). StarCraft Micromanagement With Reinforcement Learning and Curriculum Transfer Learning. IEEE Transactions on Emerging Topics in Computational Intelligence, 1–12. https://doi.org/10.1109/TETCI.2018.2823329

Design and Technical Approach

Broadly, the work by Shao et al. can be split into 3 parts: state representation, reinforcement learning, and curriculum transfer. Each of these can be implemented separately for modularity and separation of labor.

The input to the neural network is a 93-dimensional tensor containing information such as weapon cooldown, unit health, and unit position. The output is the action for each unit. For simplicity, the movement direction has been reduced to one of eight directions and a distance. We recognize the need to modify the representation for Starcraft 2. This portion can also be expanded by experimenting with different representations for input and output.

Implementing the reinforcement learning will involve creating the actual neural network, the PS-MAGDS training algorithm, and the reward function. These algorithms, as well as the model hyperparameters, are included in the paper, but these will likely need to be modified for optimal performance.

Implementing transfer learning requires training and testing the algorithm on various scenarios. These include the minigames included with PySC2, but additional scenarios will have to be constructed.

We will use Python 3 with the SciPy stack (NumPy, SciPy, Scikit-learn, etc.) and PyTorch for neural network support. Interaction with Starcraft 2 and visualizations

will be handled by PySC2 and the Starcraft 2 Learning Environment. Python is an appropriate choice because it is the primary language for machine learning research. The ergonomics of the language will allow us to implement ideas quickly.

We will take advantage of the computers in the CSIF for much of the training and testing. The specifications on these computers exceed those of the computer used for testing in the source material. In addition, the CSIF PCs contain GPUs for accelerating neural networks.

The choice of development environment is personal, but the majority of development will be done using the Jetbrains Pycharm IDE. Pycharm is the preferred choice for Python development due to its powerful analysis and quality assurance tools.

We will use Github to host our code and track development progress and issues.

Scope and Timeline

Phase I: Reimplementation of Paper

Tentative Due Date: May 11 (2 weeks)

Deliverable: Code base on Github

This part will reimplement the source material for Python and Starcraft 2. The original paper includes detailed descriptions of the specific processes used, so we are optimistic about meeting the deadline.

Phase II: Training, Testing, and Transferring

Tentative Due Date: May 25 (2 weeks)

Deliverable: Quantitative results and visualizations

This part is dedicated to training and testing the algorithm and transferring the knowledge to other scenarios. This process will be limited by the resources available to us in the CSIF.

Phase III: Extension and Adjustment

Tentative Due Date: June 8 (2 weeks)

Deliverable: Additional results and final report

This part will be dedicated to optimizing and extending the model. The flexible nature of this allows it to grow or shrink as time allows.

Documentation and Access

We will use Sphinx to generate documentation, due to the ease of integrating equations and technical diagrams. The documentation will be hosted on Github.

Evaluation

Part of the testing procedure will involve the built in PySC2 minigames. Since these are popular in the Starcraft AI community, we have several baselines to compare to. Since we are trying to transfer knowledge rather than learn scenarios from scratch, we do not expect to match or exceed the baselines. Instead, we are aiming to score within 80% of them. There are no existing results for our custom scenarios, but we are willing to collaborate with other teams to generate a database of results. If time permits, we can also test the performance of related work in the field.

In addition to quantitative measures of performance, we will also analyze footage of gameplay to identify the “human” strategies used by the AI.

Plan for Deliverables

In addition to the code, we will deliver a paper detailing the approach and results. The final presentation will include either recordings or a live demonstration of the system’s performance.

Separation of Tasks for Team

These roles are tentative and will be adjusted as the project progresses.

State representation - Robert Park

SC2 interface, Scenario Design, and Visualization - Jordan Palmer, Xingwei Ji

ANN/PS-MAGDS - Rishi Subramanian

Training and Testing/Infrastructure - Adrian Wang, Xingwei Ji

Transfer learning - Jordan Palmer