Project: Happy Kitting

ECS170, Spring 2018

Group Members

Qing Lin,	qlilin@ucdavis.edu,	913802104
Chu-Hung Cheng,	chccheng@ucdavis.edu,	914198299
Yean Li,	yeali@ucdavis.edu,	912187273
Siyuan Yao,	jimyao@ucdavis.edu,	913199781
Oi Lam Sou,	olsou@ucdavis.edu,	914393890
Nghiem Trong Van,	nvan@ucdavis.edu,	913535853

Overview

Starcraft II is a complex real time strategy video game that combines fast paced micro-actions with the need for high level planning and execution. With PYSC2, numpy and some other powerful tools, we can explore, simulate and study machine learning. This project is deployed on the mini-maps that were created with Starcraft II editor. With an ultimate goal of creating a fully functional A.I. that learns to play in normal game mode against other players, we start with this mini-game to gain experience and get prepared for the future challenges.

By implementing different machine learning algorithms in this project, we are expecting to learn how different algorithms approach problems differently, and thus gain a better understanding of when to apply each algorithm to solve real-life problems. In a group of 6, the jobs are distributed evenly. Each member is assigned with tasks individually as well as collaborating with others as a team. With that being said, all members will get their hands onto both designing the game, coding and writing reports, etc.

In a carefully designed mini-game, our A.I. will learn how to control the marines to defeat roaches. The A.I. is expected to learn how to control the units and give complex commands to maximize the total value of rewards received from the environment in response to its actions. In our carefully designed game, the rewards would be more enemy units eliminated, less health points lost and less time spent to defeat all enemies, etc. And the penalties can be the opposites.

In approximately 6 weeks, the team members will start from learning the tools and APIs, study the 4 different algorithms - Monte Carlo Search, Q-learning algorithm, Deep Q learning algorithm and Deep Deterministic Policy Gradient. Then implement them with python, train and test the models and finally give a project demo during the last 2 weeks of the class.

Problem and Contribution Statement

Unit micro-management and tactical strategies are the two cores of RTS games. Therefore, in order to win an RTS game like Starcraft II, aside from winning the engagements, good unit micro-management is often necessary to win the battles and lead us to victory. In this project, we are going to focus on implementing the unit micro-management to maximize the use of units. Micromanagement is not limited to RTS games. It is applicable and beneficial to some business models in some occasions as well. Implementation of a good unit micromanagement can maximize profits while using minimal cost.

Although there already exists solution with different approaches to this problem, to name an example---https://github.com/simonmeister/pysc2-rl-agents, we found them not powerful enough to implement in real time since these solutions are mostly implemented with decision-makings based on a pre-planned paths, which will no longer be meaningful if the prediction goes wrong. Suppose there's a scenario where the enemies are microing their units as well. In such case, pre-planned path won't be useful because it cannot give us a good prediction. Therefore, a better approach to solve this problem is to use real time path planning. That sets the fundamental goal for us---implementing an agent that can respond in real time and make decisions correspondingly.

The reason for real time path-planning is to create agents that can micro the unit better than an average human. From what we observed in the existing solutions, average human players are still doing a better job than the AI's, which makes them not as exciting as expected.

This problem is technically interesting because it will provide us with hand-on experience on how machine learning works, how to set up the environment and utilizes pre-existing algorithms to allow the machine to achieve the task of ‘learning’. We would be working with TensorFlow, an open-source library that is widely used in the recent technology development, which will allows us to gain first-hand experience on the mechanism of AI and how companies like DeepMind and OpenAI are accomplishing difficult tasks. Other than that, we would be able to explore advanced algorithms such as Deep Deterministic Policy Gradient (DDPG), and how to apply it to solving problems. Specifics on such algorithms will be mentioned the the Design and Technical section.

To provide a better gaming experience for our players, we want the AI to behave just like another human player. There are many different approaches to this. For instance, traditional AI use cheat code to outperform human player. This approach forgoes the fairness of the game which is something that should be avoided. Other approaches such as static or dynamic game bots are not challenging enough to provide an exciting gaming environment. Therefore, our objective is to train the agent to perform better than an average human player by realizing the advantages and disadvantages of ranged units against melee units in this gameplay. Our solution will provide a nearly perfect way to micro a unit in an engagement with minimal casualty, maximum hp leftover and shortest duration of battle by rewarding the agent if it hits an enemy and penalizing it if it is attacked. This will be a better solution than the existing ones.

Design and Technical approach

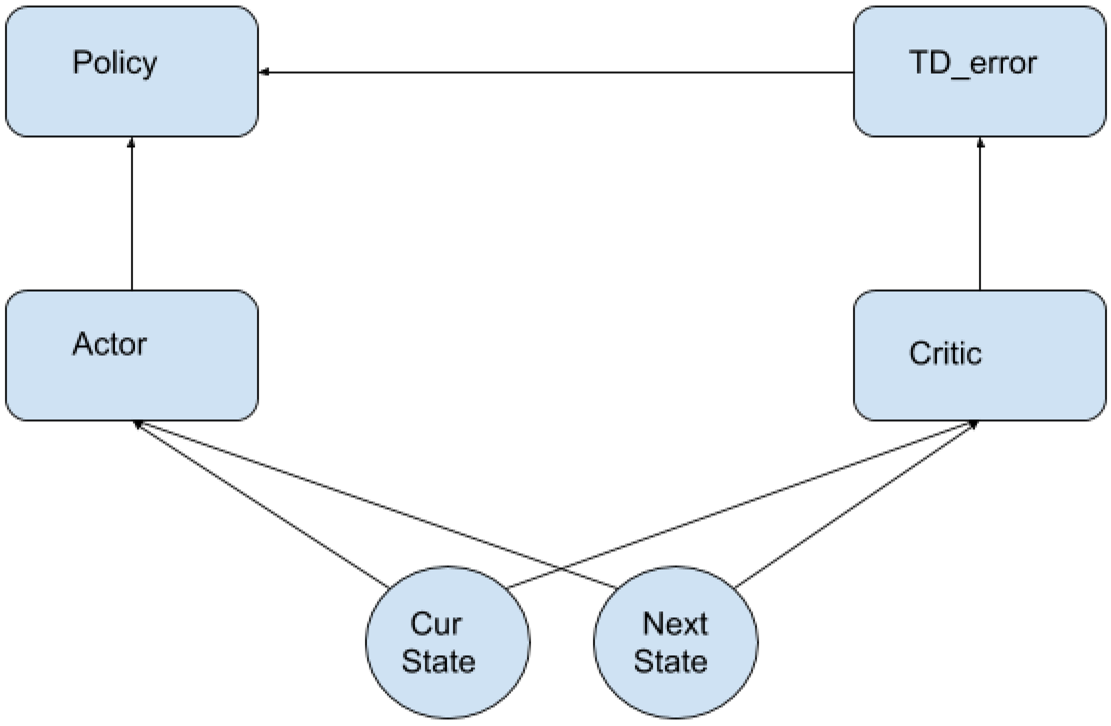
To achieve the goal of building a smart AI that is able to micro the units and respond in real time, we will be implementing some reinforcement learning algorithms. The reason that we chose reinforcement learning as our main focus is because reinforcement learning enables an agent to learn, starting from knowing nothing, and eventually approach the goal. This feature of reinforcement learning makes it a better technique than other machine learning algorithm, such as supervised learning and unsupervised learning, in a dynamic environment. Among all of the reinforcement algorithms, we are aiming to implement the Deep Deterministic Policy Gradient algorithm (DDPG) for our project.

Before explaining why we have chosen DDPG as our main approach, we would like to point out the foundation of DDPG: Actor-Critic algorithm. Actor-Critic algorithm is consisted of two parts, actor and critic. Actor is used for making an action, while critic is used to evaluate the action. In practice, actor is implemented with policy based algorithm such as policy gradient, which allows the algorithm to converge in continuous environment. On the other hand, critic is usually used with valued-based algorithm such as Q-Learning, which allows the algorithm converge faster due to the fact that valued-based algorithm updates more frequently compared with policy-based algorithm. To conclude, Actor-Critic is an algorithm combines the advantages of the two main category reinforcement learning algorithm, policy-based and value-based algorithm. Yet, DDPG is an advanced version of Actor-Critic algorithm. In short, DDPG is the combination of Actor-Critic and Deep Q Network. The introduce of the Deep Q Network not only makes the training to converge faster but also makes the training result more accurate. As a result, we have observed that DDPG is not only a proper fit for our project but also a challenging and interesting algorithm to implement. However, according to our observation, we didn't find many implementation of DDPG for pysc2 agents on Github. Rather than implementing DDPG, many have applied the A3C algorithm, which is also an advanced reinforcement learning algorithm. Therefore, we will be using a less common reinforcement technique, DDPG, and we are hoping to have a successful outcome.

In general, we will be extracting some useful information from the pysc2 api as inputs for our algorithms. So far, we haven't decide what information we will be used as inputs due to the lack of time of exploring the pysc2 api. During the first stage, we will be training our agents against the stand ground opponents, which aren't smart and will be easy to predict due to the lack of smart control. After our agent is able to take down the stand ground opponents, we then can move on to the second stage and train our agents by playing against itself. Some policies that we have came up with to guide the agent are the most overall leftover hp, least amount of time used, and most leftover unit count etc. For instance, we will be given higher reward to the agent when there are 5 units leftover than 3 units leftover. The result which we are expecting is that our agents will be able to micro the units and respond to the opponent's action in real time in a level where it surpasses average Starcraft 2 human player's micro ability.

Since the DDPG is a complicated algorithm, we will have at least four people working on the algorithm. In addition, we are aiming to complete at least an Actor-Critic version agent if unfortunately we failed to come up with a good training result using DDPG. In contrast, we will try to implement other algorithm and compared the performance with DDPG if we have more available time.

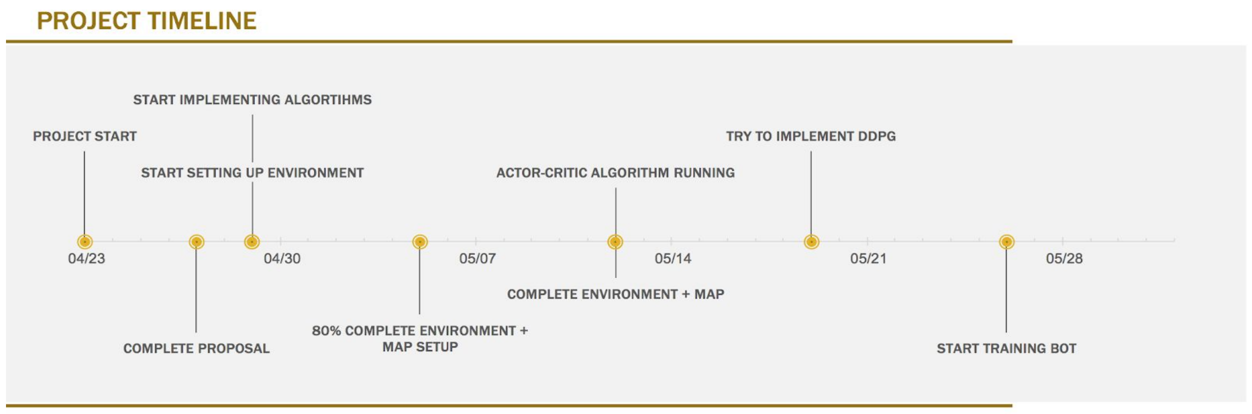
We will be using python for our main programming language because python is one of the most common language used with some reinforcement learning tools. Moreover, python is comparatively an easier language to learn and also an ideal language for high level programming. To bring in variables from Starcraft 2, we will be using the pysc2 API. We will also be using numpy and pandas for basic vector and matrix implementation; matplotlib to generate graphs while training our agents; tensorflow for implementing the reinforcement learning algorithms, and potentially Keras later on to build the neural network. Our agent will be fully run and tested under python 3.6. Since python3 can be run on a wide variety of platform, we will be running our agent on different operating system including Windows, Mac Os, and Ubuntu. In addition, each of us is likely to be using different IDE to code. For instance, jupyter notebook, pycharm, and visual studio code. Github will be the place where we will share our code by using effective version control and commit. For other shared information such as proposal, additional research paper and reading, we will be sharing all these resources using a google team drive. Lastly, we will be relying on peer review for code check.



Scope and Timeline

Provide a justification for the feasibility and scope of your project. Provide a timeline covering the major deliverables and technical problems.

We will separate the projects into two parts, one part is setting up the environment and making sure we have a map scenario that we can train on, and other part is to understand and implement the algorithms.


PROJECT TIMELINE
PROJECT START
START IMPLEMENTING ALGORTIHMS
START SETTING UP ENVIRONMENT
ACTOR-CRITIC ALGORITHM RUNNING
TRY TO IMPLEMENT DDPG
04/23
04/30
05/07
05/14
05/21
05/28
COMPLETE PROPOSAL
80% COMPLETE ENVIRONMENT + MAP SETUP
COMPLETE ENVIRONMENT + MAP
START TRAINING BOT

For week 5, All the group members will start on either setting up on the map and environment or getting familiar with the algorithms

For week 6, Ideally, members will be close to finish setting up the environment and the maps.

For week 7, Member should be done setting up the maps, and jump into implementing the algorithms. And we should have a dumb algorithms up and runnings

For week 8, we are training the algorithm and if we have more time we would be implementing a more difficult algorithms and every one in the group is working on it?

For week 9, we should be training the more difficult algorithms at the same time, starting on preparing for the deliverable.

Evaluation

Describe your criteria for success. How will you gather data to evaluate your solution? How will it be analyzed?

The AI bot can play against itself to get better in the game, but in reinforcement learning, the learning will never stop, but the skills will keep growing, the goal is to the aim that the AI acquire the skills that is enough to beat professional players?

Separation of Tasks for Team

Environment setup:	Lin, Yean, Siyuan
Algorithms (Actor-Critic -> DDPG)	Nick, James, Oi
Training & Testing:	Siyuan, James, Nick
Inline Documentation:	Yean, Oi
Reports:	James, Lin, Nick