An entry-level *StarCraft II* AI program

Zhong Mengyan [1]

Abstract

In recent years, the RTS game *StarCraft II* has gradually become a focus of AI research due to its complexity. This project aims to combine the knowledge of artificial intelligence and machine learning to implement an entry-level AI program *StarCraft II*.

1. Introduction

1.1. About *StarCraft II*

StarCraft II is a Real-Time Strategy (RTS) game by *Blizzard*, released in 2010. The main objective of the game is to build a military and defeat the enemy. The fog of war mechanism and huge strategic space of *StarCraft II* are extremely challenging for AI.

There are three races (Terran, Protoss and Zerg) in the game, and players will choose one to play. Each race is wildly different from the others, and each of them have completely different units and play styles.

1.2. Intention

In this project, we want to implement an entry-level *StarCraft II* AI using supervised learning. Since there are built-in bots in *StarCraft II*, we will use them for training data generation and program evaluation.

2. Training data

2.1. Choice of training data

In *StarCraft II*, not only are there numerous variables, but the number of variables is also variable. So we need a fixed dimension quantity as input data. In addition, the input data should also be able to reflect the current battlefield situation.

We found that the size of the map remains the same during a game, so we visualized own units and enemy units based on the map. And to reflect certain data such as population and resources in current time, we drew some thin lines of different colors in the lower left corner of the image, each of which reflects the size of the corresponding data in its length.

It should be noted that the training effect of gray-scale images is not good, which is why we choose color images.

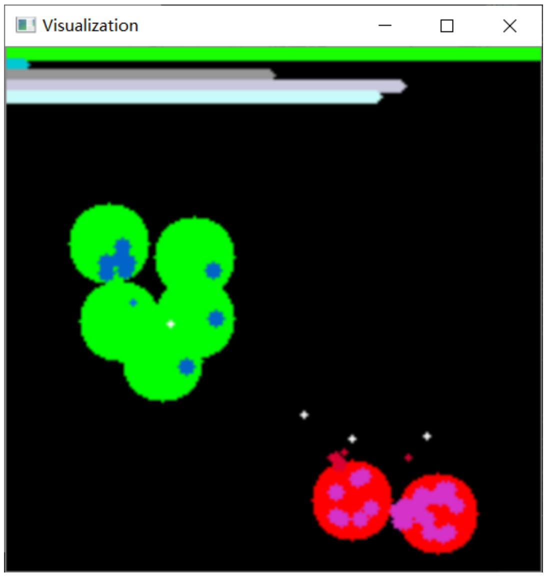

Figure 1. The visualization window

Because each map has different terrain and resource locations, the visualization and neural networks of one map are only suitable for this map. Therefore, all work below is done on a single map named *Abyssal Reef LE*, which is in the map pool of 2017 Season1 Ladder.

Taking the right action consistently is the key to win for AI. Therefore, we pre-prepared several actions of combat units (like attack one of visible enemy units or stay to protect the base) for AI to choose at certain moments.

The choices made by AI are the labels of the neural network below.

2.2. Get training data

At a certain moment in game, if all combat units are idle, then let AI choose an action randomly.

[1]ShanghaiTech University. Correspondence to: Zhong Mengyan <zhongmy@shanghaitech.edu.cn>.

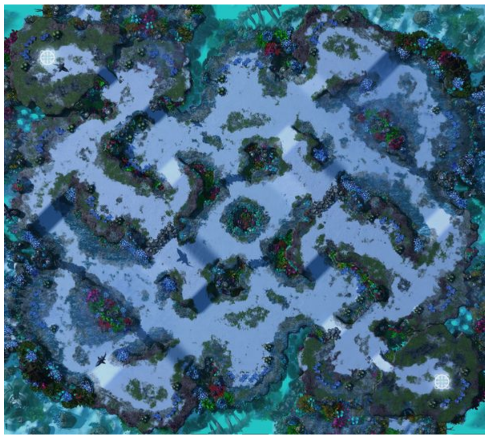

Figure 2. Abyssal Reef LE

In fact, in order to simplify the process, we only provide AI with 4 options for actions of combat units, the specific options are as follows:

1. Attack enemy unit which is closest to base.
2. Attack a known enemy unit.
3. Attack a known enemy structure.
4. Attack the enemy's start point.

It should be noted that in the hard-code part, we only had the AI train only one type combat unit called Voidray, which means that the AI used only one tactic from start to finish, and our job was just to use machine learning to adjust the relevant variables to improve this tactical details.

Each choice will get a pair of corresponding matrices:

1. A $p \times q \times 3$ matrix ($p \times q$ is the size of map) with data type uint8 to represent the resulting image.
2. A 4×1 matrix to represent the choice made by AI. The value corresponding to the chosen action is 1, and the rest are 0.

After a game, store all the image-choice pairs obtained in the game in one .npy file. According to the evolution strategy, we only store the data of the winning games.

3. Method

3.1. Choice of method

Due to feature of *StarCraft II*, it seems to be too complicated using reinforcement learning like Q-learning. Therefore, we finally choose Convolutional Neural Network (CNN) as the method of AI learning.

WIN RATE (%)	VS. TERRAN	VS. PROTOSS	VS. ZERG
EASY	96.5	100.0	86.7
MEDIUM	45.3	40.3	61.2
HARD	0.0	9.5	25.0

Table 1. Win rate of randomly choosing

WIN RATE (%)	VS. TERRAN	VS. PROTOSS	VS. ZERG
EASY	100.0	100.0	97.6
MEDIUM	81.0	78.9	86.8
HARD	21.4	33.8	40.2

Table 2. Win rate of using model

3.2. CNN model

We chose CNN as the machine learning model for this AI because of its simplicity and relatively small amount of computation.

Considering that linear rectification function (ReLU) can effectively avoid the situation of overfitting and the disappearance of gradient compared to other functions like Sigmoid function, so we chose ReLU as the activation function.

4. Result

4.1. Process

We ran 560 games of AI vs. built-in bots and got 270 data from winning games.

After training our CNN model using these training data, we let the trained CNN model make choices for AI instead of randomly choose.

4.2. Evaluation

We counted the winning rate of AI against bots with different difficulties and races under the two strategies.

Because of the high win rate of games being against with Easy bot, we only use Medium bot and Hard bot to generate training data for the effectiveness of training data.

It can be seen that using the trained model to choose can greatly improve the win rate compared to randomly choosing, especially when playing against medium bots.

In addition, we also find that the win rate of against Terran bots are lower than bots of other races. It may be because the tactics of AI are restrained by this race.

5. Conclusion

5.1. Achievement

We increase the win rate of AI which only used a single tactic against the computer by using a convolutional neural network.

Finally, we get an AI with approximately 80% win rate against Medium bots.

5.2. Follow-up work

In fact, this project still has a lot of aspects for improvement. Some of them are as follows:

1. Improve actions: mainly in increase the number of actions and adding other types of actions.
2. Improve visualization: as input, visualization is very important for model training. We can adjust parameters like the circle size of a certain type unit, or color, to improve training effect.
3. Micro manipulation: There is no limit about Actions Per Minute (APM) for AI. The average APM of top professional players is in the hundreds, while AI can easily reach tens of thousands. So AI can easily do some things unimaginable by human players, for example, finely manipulate every unit.
4. Add other tactics: Use different tactics to deal with bots of different races.

We will continue to improve this project if there is time in the future.

Reference

The open-source Python libraries of *StarCraft II* provided by *Blizzard*:

1. https://github.com/Blizzard/s2client-api
2. https://github.com/Blizzard/s2client-proto
3. https://github.com/Blizzard/s2protocol

The open-source Python libraries about learning environment of *StarCraft II*:

1. https://github.com/Dentosal/python-sc2

Unit names and Tech trees of *StarCraft II*:

1. https://liquipedia.net/starcraft2/Main_Page