Survey and Analysis

Name: Junjin Chen

Supervisor: Eleni Vasilaki

Module: COM3610

Date:

This report is submitted in partial fulfilment of the requirement for the degree of BSc by Junjin Chen

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Junjin Chen

Signature:

Date:

Abstract

Inspired by Deep reinforcement learning of Atari games done by Deep Mind which in many games outperforms than human player, this project will demonstrate the learning algorithm on a Real Time Strategy (RTS) game Starcraft2. Applying learning method in RTS games are generally considered challenging since there are usually large action space and large state space in the game as well as large amount of possible strategies for future states, especially in this case where Starcraft2 is a typical RTS game. However, open source from the cooperation between deep mind company and Blizzard entertainment has provided useful and convenient platform for researchers. It includes convolutional layering from the raw data, simplified maps and some example base-agents. Learning methods and the policy-reward functions are only needed to be concern from the researchers. Other researches about other games online also provide the theory of how to apply the learning algorithm into the game. In this article, important terminologies and relevant literature reviews are presented.

Content

1. Introduction ……………………………………………………… 5
2. Literature Survey ……………………………………………………….
3. Requirements and Analysis ………………………………………….
4. Progress ………………………………………………………………..
5. Conclusions and Plan…………………………………………………

Reference…………………………………………………………………..

Appendices…………………………………………………………………

1. Introduction (700)

Reinforcement learning is a powerful technique to perform intelligent state space search. Some problems have the enormous amount of possible states that may require ages using brute force to compute the best solution. RL method learns from feedback of actions and improves gradually. In some problem, RL method can outperform the human.

Deep Mind company has done many researches on Artificial Intelligence (AI). It applied deep reinforcement learning in Atari games. The agent only received high dimensional sensory input (i.e. Pixels on the screen, scores in the game), which set the agent in the similar condition as human when playing the game. Initially the agent has no knowledge about the game or the result of each control-key. It automatically plays the game and learning and improving itself by analyzing the outcome of the game(score). This method is known as Deep reinforcement learning, where deep addresses that it learns from raw data, without pre-programmed input. Thus it can play across wide range of games and it is considered as world’s first ‘General purpose machine’. Result was that in some games it outperforms than human player. Coo-founder of this company Demis said that he was surprised after space invader agent was able to make the predict shot at the end after training for one night. A video shows that the agent in the beginning loss very often, however at the end it barely missed a single shot. Another example is after it played the Breakout 500 times, it discovered the optimal strategy which is to dig a tunnel round the left-hand and then send the ball round the back. But in some games it performs poorly since that game requires pre-information such as English Language Montezuma's Revenge. Another famous example of this algorithm is Alpha Go, who beat the human professional Go player with 9 dan (highest rank in Go) Lee Sedol with the score of 4-1. AlphaGo was initially given 100,000 games to mimic human player, after that it was ordered to play the games itself 30,000,000 times, which made the Alpha Go stronger than human. The difference between Atari game and AlphaGo is former one is online and alphaGo trained with offline data. Those examples show that deep reinforcement learning algorithm that DeepMind has been using is pretty strong.

[10] Deep Mind company now cooperate with Blizzard entertainment and provides the open source of Starcraft2 learning environment and agent for researchers. Starcraft2 is a typical RTS game with rich culture and. There are many elements in the game such as different units with different abilities, collecting resources over the map, different structures produce different units, micro-control and long-term planning the strategies to defeat enemies. It is fast paced game meaning that one 1v1 game may require 10min-1hr. …the challenge in this games are: firstly in multi-player game (2v2 or 3v3, 4v4) agents’ actions influence the overall situation. Secondly, in the agent’s vision, the map is cover by fog where it doesn’t explore. Which means the agent will not get full information about the whole map. Thirdly, the action space and state space is large since the game includes spatial action. Deep Mind company has also done some work in this research. They used Atari agent, full convolutional agent and fullconv LSTM agent. Result is that Atari agent outperforms in combat game such as FindAndDefeatZerlings, DefeatRoaches and DefeatZerlingsAndBanelings. While only full convolutional network agent learned to produce the worker to increase the income in game. In full game, the best agent was only able to avoid constant losses by using the ability of lifting the structure and move them out of the attack range from enemies.

Other researches about other games like bubble shooters provides it methods of how to applying the learning algorithm to the games… the terminology will be explained in the following chapters. However some results are not as strong as the deep mind experiment of atari games, i.e. not as good as human player. that could also be my problems of not able to play as good as human since there are many things to be concerned about, thus it is challenging. Another problem could be time taken for the learning process. It could be too long until get an unsure result…

Task is to apply the learning algorithms to starcraft2, while thinking about the policy and reward function according to the game structure. Next chapter will explain some terminologies and methods used in other researches. Then analysis about which function should be used in the game will be show and plan as well.

1. Literature Survey (2300)

[1] Reinforcement learning is defined as to let the agent take an action which will place the agent to a situation, through that there will be reward signals and agent need to be able to maximize the reward signal by choosing relevant action. In some cases, actions from the agent may affect future rewards. Thus that the agent should be able to search action space while considering future rewards. The way to let the agent exploring and exploiting the action space is using stochastic method to using weighted actions that agent can choose. Reinforcement learning consist of a policy, a reward function, a value function, and if you like, a model of the environment.

Policy of the RL tells the agent what to do in a particular situation.

Reward function tells the agent whether current situation is good or bad.

Value function tells that whether the current situation is good for future states, basically it roughly calculates the expectations of the future reward function.

Model roughly tells what the environment is generally like in the agent’s sense. It helps the agent to do the planning.

Action value methods calculates the mean value of the rewards of the situation that an action that being chosen by many times.

Greedy function is to choose the action that has the most action value. This method has the disadvantages that it only does the exploitation.

ᵋ-greedy function defines that while choosing greedy actions there is small chance ᵋ that the agent completely ignores the action value methods. This enables the exploration. However, if in its exploration, the probabilities of choosing all actions are equal, the agent might choose the worst-action.

Softmax action selection deal with this case where it ranks the actions to be explored with the estimated value.

Implementation of action value can be convert so that it doesn’t store all r values.

For nonstationary problems, it makes sense to weight recent rewards heavier than long-past ones.

Instruction tells the agent what the right action would have been no matter what action the agent selects. Few of them would be useful to direct agent’s action in selection rule.

Optimistic Initial action value can be used as a way of encourage exploration if set to larger than 0.

Reinforcement comparison method uses a reference reward to indicate whether the reward gain by an action is good or bad. These kinds of method are sometimes more efficient then action value method.

Returns sums the rewards from now to the final step, it is used in testing (rather than training).

Agent-environment interaction can be break into subsequences naturally; they are called episodes. Episodic tasks are easier to compute than continuous task since in episodic task, the actions affects shorter number of rewards.

Rewrite the return function to include the discounting factor among future states results the same while it can save computes.

Markov property is obtained if an environment’s state signal compactly sumerize the past without degrading the ability to predict future.

Markov Decision Processes builds on Markov property and use a function to determine an action to do. There is a function to calculate the possibilities of next states given the current state and action. Another function of MDP calculates the expectation of rewards given current state and action along with next state.

Value function calculates the expected return for state (denote V) or state-action pair (denote Q) in a policy. It tells how good is the state that agent wants to get.

Optimal value functions calculate expected return for optimal state-action pair. (i.e. optimal policy)

Bellman equation expresses a relationship between the value of a state and the values of its successor states.

Bellman optimality equation is the bellman equation for optimal value functions.

Dynamic programming are some algorithms that can find the optimal policy given a model that has Markov property. Since the Markov property is impossible in real world, there will be some limits when using DP in RL.

Policy evaluation is the process in DP that uses the value function.

Policy improvement can be done by update the policy iteratively in every step with little change about action for a state that has higher Q value, this process will continue until the Q value stabilize.

Monte Carlo Methods in agent that it learns value functions and optimal policies from experience in the form of sample episodes. In other words, it calculates the average state-action values over random samples. First-visit MC only calculates the average returns but the first visit of the state. They can be used to learn optimal behavior directly from interaction with the environment even without models of environment.

A problem of this method is that if the policy is breath first search, there will be no average of the state-action values. So policy should be some stochastics of weighted action.

On policy is that the algorithm is able to evaluate and improve the policy that determine the action.

Off policy is that the policy that determines the action may not be evaluated or improved, but a hidden policy can be evaluated and improved by observing. Advantage is that episodes can be generated and sampled continuously of the same policy.

Temporal-Difference combines MC method and DP method. Different from MC, TD method updates the state-action value immediately after each time step, where MC must wait until the end of an episode. It is a bootstrap method, meaning that it learns its estimates from its other estimates. Its advantage over DP is that it does not require a model.

TD(0) improves the agent after a batch of episodes, this method is called batch update.

Sarsa is one method that use five parameters to evaluate the current Q value using TD method prediction of future Q value. The difference between on-policy Sarsa and Off-policy Sarsa is that on-policy using a policy to determine the action in next state, where off-policy using a method called Q-learning. It choose the maximum Q value of the future state, irrelevant from the policy.

Eligibility trace is some temporally record of an event and it marks the event as eligible if the result is expected and otherwise not eligible. This method can be used to alternating different methods as they have gone extreme.

Forward view of TD(lambda) means to decide how to update each state by looking in the future state. This could be not implementable since looking in the future to determine the current state is acausal.

Backward view of TD(lambda) accumulates the eligible trace with some discounted factor. It then assigned the current TD error to the previous state.

Eligibility trace in Sarsa evaluates the action-state pair instead of state value.

One problem of reinforcement algorithm is that there could be large action space and state space. If we assume that our value functions are represented as table of state-action space, it could be huge. Thus we need to generalize some good approximation from small limited subset of state-action pair in a large space.

Function approximation is to construct an approximation of entire function from some examples of desired function. This is an instance of supervised-learning.

[2] Starcraft2 is a classic real time strategy (RTS) game, where players control their units to construct base and build armies and fight for resources. The player needs to send his workers to collect the resources, using the resources to build some buildings, produce armies, develop technology to enable some special ability of the units, explore the map, produce different units and combine them to perform efficiently in compact. And the player needs to make balance among those.

Collect resources: player controls the workers to collect mineral or gas, workers will store those resources in the base building. Thus worker needs to travel between mineral field or gas field and base building. And thus the base building should be built as close to the resource field as possible. Resources are the economy of the game, it can be transfer to other things in the game, hence effectively collect the resources will have good position in the game.

Buildings: some buildings produce units; some buildings enable the player to develop some abilities of the units or be able to produce different units; some buildings can be used in defense to attack offence enemy. Different buildings may cost different resources.

Units: works are used to build base, some units output damages to enemy, some units transport other units, some units assist the fighting units. Player needs to combine different units to perform an effective compact. There are loans of combinations and weights of the army which contributes to loans of the strategies in the game.

Support: it limits the number of units that can be produce. Different units may require different amount of support. Additional support will be required when player wants to produce a unit that may cause exceed of the support. Depends on species, support can be increased by build some buildings or produce a certain kind of units (that unit will not cost any support). The maximum support is 200.

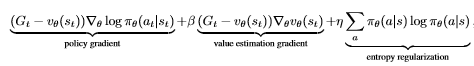
Technology: technologies can be developed in certain buildings. They make the army stronger and efficient and substantial. Some technologies increase the units’ damage or armor; others enable the units’ special. They cost certain amount of resources and time to be developed. The player needs to balance the time to develop a technology.

Explore the map: the map is covered by fog until the player explores it. The enemy can be anywhere of the map, player should regularly visit some place of the map in order to observe the enemy’s actions and predict its motivation.

Expand the base: resources field will be exhausted in the future, and player needs to find another resources field to collect the resources in order to keep good economy. When doing the expansion and migration of those workers, player should also consider about the position of the resources in the map, whether it is good in defense or attack based on the situation the player is in.

convolutional neuron network transforms the raw input data into useful data structure.

[10] The previous work done by DeepMind in Starcraft2 game was that using a method called Asynchronous Advantage Actor Critic (A3C) to learn the policy parameters. A3C looks like this:



Where is the return that the agent aims to maximize, and it is expressed as:

There are 2 architectures that DeepMind used in the experiment. One is the Atari-Net architecture, which process the non-spatial features to a linear layer with a tanh non-linearity. The screen and mini map are processed to convolutional layers, then combine with the result from non-spatial features. Then they are processed into a linear layer with ReLU activation before inputting to the agent. The second one is the Full-Convolutional Network architecture, where it concatenates the result of convolutional layered screen and minimap, with the non-spatial features as state representation. By this means the agent predicts spatial action directly through the state representation.

The result was that, overall, the full-convolutional network performed better than Atari-Net. In particular, Atari-Net performed strongly in combat task, where full-convolutional network was able to learn to produce more workers in the task of CollectMineralsAndGas. Compare with human player, both networks in some task only able to outperforms a casual human player, but seems very weak comparing to Grand Master player.

[3] There is some other research of RTS game in Glest. The game has many different elements such as Technology and Magic. Players in this game need to collect resources, build different buildings and armies and fight the other team. The problem of current AI in this game is that they are not intelligent and once human player finds it weakness, it is easy to be defeated.

Their research was using LSPI and SARSA in their reinforcement learning methods. Firstly, they create their own score scheme, then they developed formulas for both SARSA and LSPI, they implement those methods in C++ as that was the language the game using. Next they test both methods on every 10 games training. And play games for 20 tests, in other words, totally 200 games are played. Since RTS games are usually required a long-time to play, 200 is a number enough in the experiment. They finally get a conclusion that using LSPI with SARSA sample performs best among using SARSA method or LSPI with random samples. They did not test the game with human player.

Other researches may not apply the reinforcement learning in RTS game, however, their method could be useful as they can be regarded as each component of RTS game. for instance, bubble shooter can be seen as the agent has the ability to analyze the state and make optimal selection of an action. Racing game the agent response fast according to the situation. Etc.

[4] Bubble shooter is the game that has different colors of bubbles that needs to be clean on the screen, and the way to clean them is to shoot the bubbles with same colors so that bubbles will be eliminate in a sequence of a least 3 same color bubbles. The method they used was Deep reinforcement learning. However, the environment for the agent is simplified such that makes it easy for agent to learn. They have their own reward schemes too. The result is that the agent wins 95% of the game. However, when competing to human player, the agent performs poorer than human player. The author blamed the time of training.

[5] Racing game is another research that using deep reinforcement learning. It is a game that player needs to control his car to race. Author firstly processed the raw image on the screen then the signal was transfer into the learning network. The result was that the agent performed excellent actions in early game, while in later game entering the desert where the color of the road and the side was too similar, the agent began to make mistakes. Finally, agent was able to learn keeping the car in the center of the road.

[6] Flappy bird is a simple game where player only needs to click to make the bird flap in order to pass barriers (pipe). The input was passed into two convolutional networks including the ReLu. Q-learning method was used in their deep reinforcement learning. So there will be no policy. Replay memory was used to store the experience, which will finally convert to gradient decent to update parameters. The result turns out that with both loss functions the agent does not outperform than human player.

[7] FPS game as Doom also used Deep-Q-Networks in their experiments. FPS games like Doom player has 3D experience as the game allows player to rotate the camera and point to an object and shoot. The architecture they used was 2 convolutional networks to process the image through multiple layers, some will go through LSTM and the others go to game features. Frame skip technique was used for the agent where the skip step was 4. The result turns out that it outperforms human players in both single-player and multiplayer situations.

[8] Deep mind has done some experiment with Atari games. Where it uses 2 convolutional networks process the input then feed through the deep reinforcement learning. TD-Gammon architecture was used in the learning process to update the parameters by estimating value function. Experience replay method was used in deep Q learning to utilize the agent as well. This architecture was testing across 7 Atari games without scripted the program for different games. The result shows that in some games the agent outperforms human but other games the agent does not, however, DQN agent across all games performs better than SARSA agent.

[9] Battleship use deep reinforcement learning in scripted input. And it used 2 layer convolutional network. After 2500 training process, the agent was able to find out the ship within average 4.6 moves.

1. Requirements and Analysis (1000)

The objectives of this project are to apply the Reinforcement Learning to Starcraft2, and make the agent to be able to defeat human players.

The first task requires to design the architecture of the reinforcement learning agent in the game, then determine the learning methods to use in the reinforcement learning, design the code to perform the learning process. Analyze those methods and predict its outcome (e.g. what will be the score the agent will get after training for one day). Train the agent using the replay data and let it play the games itself. The result of the game play will be recorded to analysis and evaluate the agent. There could be some debugging sessions if the code runs wrongly, which will be noticed if the result is not as expected. Some unexpected result may not be the fault of the code, it may be something new and it could lead to discover some new concept of reinforcement learning.

The second task requires to evaluate the agent. Apart from judging the agent from the score it gets in the end, competing the agent with human player is a more convincing way. One way to do that is to compare the score the agent gets with the one that human can ever achieve, this way is cheap, we could simply get the data from the internet. Another way to evaluate the agent is to let it play with professor human player. This way is much expensive since it requires to find such human and it has risk that unable to get volunteers or get a volunteer that is not a professor after sending the invitations.

Fortunately, the open source has provided some really useful tools for the reinforcement learning environment. Thus, the implementation of the architecture of the agent is done.

To determine the learning method for the agent, we are firstly required to be familiar with all available methods in reinforcement learning. Then we need to analyze the game structure, rules, timing and score scheme. Finally, we choose some methods that fits the game.

To write the implement code, we should know what is available, in this case, the environment of the reinforcement learning in StarCraft2 which has built the convolutional layers and transform the game state into spatial features (e.g. position of the mineral) and non-spatial features (which technology enemy develops) as the input to the agent. There are also scripted actions for the agent to pick (e.g. select\_rect() to select an area of units or buildings.). Then we use the learning methods we picked to coding the agent. This requires some knowledge of python coding; we can search some functions online if we need it.

To analyze outcomes for those methods, we need to be able to calculate the values in the structure, which requires the understanding of each term. We can also get some sample data when running the program and predict the next output by calculate the relevant values. Another easy way to predict the outcomes is to do some research about similar projects that use the same method, then we can transfer the outcome of other project to this project accordingly.

To train the agent, this may require additional step. First step is using the analyze technique to test the agent in a small game (e.g. DefeatRoaches). Then see the result and decide whether it is good for training. After that, if the agent passes the test, we then allow it to be trained in a super computer.

The agent may not perform well as expected…

After that, it is time to see the fruit of the training. We will initially let it play some games and see how it performs. We can use both evaluation methods mentioned above to evaluate the agent. When we are doing the second method, which is to find a professional human volunteer to play the game, we will firstly do some research about the culture of starcraft2, to see who play Starcraft2 very well. Then we send invitations to them and make some terms and offer. We will ask them to send some feedback to us by some files as output of the game. (determine how good is the player, what is the number of minerals that the player collects, what is the kills…)

1. Progress (500)

The project is now currently on the theory part where all the theories of reinforcement learning and the rules of Starcraft2 is understood. Other similar project is searched. The architecture of the agent in the Starcraft2 is designed. The plan of future working is designed.

Theory of reinforcement learning is explained above as well as the rules of the Starcraft2.

Here is the architecture of the whole game.

Screen

Mini-map

Non spatial features

State, Available actions

agent

game

reward

The game consists a screen, minimap and some non-spatial features.

Screen: the current screen of the game that player stays on.

Mini-map: a summary of the whole map, indicates resources position, enemy position.

Non spatial features: such as the amount of the mineral and gas, total support, etc. which might limit the available actions.

Then the agent using reinforcement learning technique to choose an action in the game.

The game then produces the reward of the action taken by the agent.

Here is the diagram that shows the detail of the agent.

agent

reward

State

Available actions

Game

policy

evaluation

model

improvement

Where the arrows indicate the input to destination or dependencies.

In PYSC2 environment, all the work of the input and connection to the game as well as a base agent has been provided. Thus we only need to transfer some reinforcement learning technique to the program of the agent.

Policy: at the beginning, we can use greedy function in the policy with some exploitation parameters. It will be improved by playing the game.

Evaluation: we can use Q values of state-action pair or V values to evaluate the current policy, SARSA can also be used to evaluate the policy.

Improvement: we can use TD method here. Because Starcraft2 is Real Time Strategy game meaning that it is continuous, it is hard to define a long is each episodes. The disadvantages can be that it would be hard for the agent to learn to predict the future. One solution can be using off-policy method, such that after the agent using the acting policy to complete a task, hidden policy then learns from the acting policy.

Model: action space is large because it consists spatial features, previous solution was to sample some typical space in the action, which reduce the action space to a size of 300. Another solution can be that using the relations between the buildings and units, thus it stores the relative position, and the target position. This model can reduce the action space and enables the fast computing.

1. Conclusions and Plan (500)

We have research what is reinforcement learning and how can it be apply to Starcraft2. We know reinforcement learning is a powerful tool in artificial intelligence. It learns from rewards and improve itself. After doing loans of training, RL may be able to outperform human level. We have walked through the rules of the Starcraft2 and consider the problem that we might face in the implementation stage. We know Starcraft2 is a typical RTS game which has lots of elements in the game. The challenge for applying RL in SC2 is that SC2 has lots of states and available actions as well as possible strategies. It might require a super computer large amount of time to train the agent. We have also designed a structure of the agent and the methods that we can used in RL. We filled our agent with e-greedy policy, SARSA method to evaluate the policy. We have analyzed the pros and cons among DP, MC and TD method. We have also designed a possible model that can be used to reduce the action space.

Below is the Gantt chart of this project.





Reference

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning An Introduction.* Cambridge London, England: The MIT Press, 1998.

[2] Blizzard Entertainment. “Starcraft2 Official Game Site” in *Game Overview,* 2017. [Online]. Available: https://starcraft2.com/en-gb/game. Accessed on: November 11, 2017.

[3] V. K. Dimitriadis, *Reinforcement Learning in Real Time Strategy Games Case Study on the Free Software Game Glest.* Chania: V. K. Dimitriadis, September 2009. [online] Available: <http://www.intelligence.tuc.gr/lib/downloadfile.php?id=347>

[4] L. Samson, *Deep Reinforcement Learning applied to the game Bubble Shooter.* Amsterdam: Science Park, June 24, 2016. [online] Available: <https://staff.fnwi.uva.nl/b.bredeweg/pdf/BSc/20152016/Samson.pdf>

[5] P. Lopes, "Deep Reinforcement Learning: Playing a Racing Game - Byte Tank", *Byte Tank*, 2017. [Online]. Available: <https://lopespm.github.io/machine_learning/2016/10/06/deep-reinforcement-learning-racing-game.html>.

[6] N. Appiah and S. Vare. *Playing FlappyBird with Deep Reinforcement Learning,* n.d. [online] Available: <http://cs231n.stanford.edu/reports/2016/pdfs/111_Report.pdf>.

[7] G. Lample and D. S. Chaplot, *Playing FPS Games with Deep Reinforcement Learning,* n.d. [online] Available: <http://www.cs.cmu.edu/~dchaplot/papers/aaai17_fps_games.pdf>.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, *Playing Atari with Deep Reinforcement Learning,* December 19, 2013. [online] Available: <https://arxiv.org/pdf/1312.5602v1.pdf>

[9] J. Landy, "Deep reinforcement learning, battleship – EFavDB", *Efavdb.com*, 2017. [Online]. Available: http://efavdb.com/battleship/.

[[10]](file:///U:/Dissertation%20Project/sc2le.pdf) O. Vinyals, T. Ewalds, K. Calderone, P. Keet, et al. *StarCraftII: A New Challenge for Reinforcement Learning,* n.d. [online] Available: <https://deepmind.com/documents/110/sc2le.pdf>.

Appendices