Using reinforcement learning to learn gaming strategies

Group Name: Cautious Train

Dongmin Wu, 605308, dongmin.wu@aalto.fi
Shan Kuan, 604888, shan.kuan@aalto.fi

Abstract

Artificial Intelligence is the technology that makes the machine to work in intelligence way and then to aid our daily life more easily. However, in the research of the Artificial Intelligence industry, the amount of data and the quality of the data, which are domains the accuracy of the result in the machine learning industry. In this project, we introduce the Reinforcement learning, which is the way that researcher does not need to collect the training data before the training step. In other words, the machine learns the rule of conduct without the supervisor, and then to generate the training data during the training step.

In this study, we analyzed two leading methods from the reinforcement learning, which are Deep Q-learning and Policy Gradients, to introduce the core ideal of each method. In deep q-learning, it is a value-based method. It tries to maximal the expected future rewards via learning of a state-action q function. On the other side, policy gradients are policy-based methods. The goal of this method is to find the optimal policy by directly optimizing of the policy function.

In the experimental, this project conducts the simple grid game to elaborate the algorithms. Therefore, the study will discuss the weakness and strength between deep q-learning and policy gradients in different application circumstance.

I Introduction

In this project, we specifically introduce two well known methods in the reinforcement learning, which is q-learning and policy gradient. Q-learning is the traditional reinforcement learning algorithm, it uses the reward function to evaluate the action and then to obtain the state in the next. For the reward function, it is not the function to tell the machine how to act. In contrast, it tells the machine what outcomes are desired or undesired in the current circumstance. In 1989, a computer scientist named Chris Watkins [7], he introduces the innovation ideal to combine the Bellman equation and the law of fact to generate the new type of learning algorithm, which is Q-learning.

In the big family of reinforcement learning, most of the methods do not have convergence guarantees after numerous of regression procedure. If the testing environment is trivial, the q function would convergence in a short term of regression. However, if the testing environment is complexity, it has plenty of different state and action that would cause the time to find the optimal policy. According to the policy gradient [4], the method does not suffer convergence problem. It uses a neural network to approximate the Q function and then directly optimizes in the policy space. During the repeat process, in the end, the policy gradient would converge to the optimal policy.

Throughout this work, we will implement the two most popular methods of reinforcement learning algorithms, deep q-learning, and policy gradient, in learning to play the maze game. During the training step, the game agent can learn how to escape the ghost and the obstacle and then to reach the final destination. In section III, the report introduces the methods deep q-learning and policy gradient [2] to approximating the Q function through different based on reinforcement learning. In section IV, we will introduce the training environment, the maze game, what is the reward and penalty in the game. Therefore, how to train an agent reach the goal in the training stage. In section V, to discuss the correlation of the hyper-parameter of deep q-learning and policy gradient. In section VI, the evaluation provides further discussion between deep q-learning and policy gradient. In section VII, the report analyzes trivial and complex environment with two methods. Furthermore, regarding the test result to discuss the weakness and strength of two methods.

II Related Work

Reinforcement learning (RL) is a big family, according to the different circumstance; reinforcement learning provides the different method to address the specific problem. Except the well known method, q-learning and policy gradient, the following research will introduce the other method on several different schemes.

In 1995, Gerald Tesauro [6] proposed a new RL methods, TD-Gammon, it is a well known method to play backgammon game. As same as the q-learning, TD-gammon used the model-free RL to approximated the value function. Therefore, TD-gammon also include the method of using a multilayer perceptron with one hidden layer to advance the performance of calculate the value function. However, TD-gammon only work well in the backgammon game, but failed in the other board game, such as Go and checkers. According to the research [5], TD-gammon is not able to converge as it may get stuck in the locally optimal solution.

SARSA stands for State-Action-Reward-State-Action. In 1994, SARSA developed by Rummery and Niranjan [1], which is the on-line policy learning of RL. Therefore, SARSA takes into account of the prediction state from the control policy during the learning step. In contrast, q-learning is the off-line policy, and it does not guarantee to follow the optimal state that predict by the control policy. To compare the result of two methods, although SARSA needs to keep the action value longer in the stored before the new updated. It provides the way to escape the local optimal and received the higher reward in the end of the regression.

III Method

The method we used in this reinforcement learning project is basically comparison. We will designed an environment of the game and run different algorithm on it. The evaluation of algorithms will be based on the curve of how many reward score in average the algorithm can earn. In this section, we will explain the environment we used and elaborately describe the implement of our algorithms.

The first algorithm we use in this project is Deep Q Learning, where the agent will learn a good solution through evaluating and updating the quality of each state.

$$Q(s,a) = r + \gamma(max(Q(s\prime, a\prime)) \tag{1}$$

Q is the quality of current state, and $Q\prime$ is the quality of all the possible state that can be reached from current state. This is the Bellman Equation, which shows the relationship between the quality of current state and quality of all possible next state.

Since we already have the approach of how to get quality of one state from its future states, we can build a neural network to predict quality Q of future states and compare this with the actual quality Q-target which derived from Bellman Equation

$$Loss = \Sigma(Q_{target}(s,a)) - Q\prime(s,a))^2 \tag{2}$$

For better prediction, we used the Dueling DQN algorithm. As the Algorithm 1 shows, this approach has two neural network, the primary network will predict the quality of next states of s and choose an action from it. The target network is used to predict the quality value Q. The primary controls the direction of agent and the target network calculates the rewards in that direction

Algorithm 1 Dueling DQN

```
1: repeat
2:     $Q \leftarrow PrimaryNetwork(s)$
3:     $s\prime \leftarrow Environment(max(Q(s)))$
4:     $Q_{direction}\prime \leftarrow PrimaryNetwork(s\prime)$
5:     $Q_{value}, r\prime \leftarrow TargetNetwork(s\prime)$
6:     $Q_{target} = r + \gamma Q_{value}[max(Q_{direction})]$
7:     $\theta_{Primary} \leftarrow \theta_{Primary} + \nabla\alpha\Sigma(Q_{target}(s,a)) - Q\prime(s,a))^2$
8: until Exceed the maximal episode
```

The policy gradient we used is a base on the equation below:

$$Loss = -log(\pi) \times A \tag{3}$$

In the equation, the π presents the policy we will choose in a state, or more intuitively, it means the chance of the agent choosing certain action based on its current state.

The parameter A represents the advantage of the policy, which is an

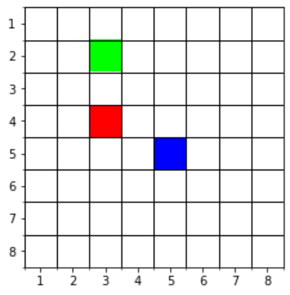

Figure 1. Example of the game we used in this project

combination with the rewards got from the game and the baseline of our expectation. The base could be changed in different scenario.

Algorithm 2 shows the pseudocode of policy gradient.

Algorithm 2 Policy Gradients

```
1: repeat
2:     for step = 1 to MAX STEPS do
3:         $a \leftarrow PolicyNetwork(s)$
4:         $s\prime, r \leftarrow Environment(a)$
5:         $Buffer.append(s, a, s\prime, r)$
6:         if terminated then
7:             $v \leftarrow r_0 + \gamma r_1 + \gamma^2 r_2 ... \gamma^i r_i (\gamma < 1, i \in [0, Buffer.size() - 1])$
8:             $\theta_{Primary} \leftarrow \theta_{Primary} + \nabla \alpha log(\pi) \times v$
9:         end if
10:    end for
11: until Exceed the maximal episode
```

IV Data

Since the goal of our project is comparing the two currently popular Reinforcement Learning algorithms, we decided to make a tiny game that we can test these two algorithms. The game we created is simple which is based on a grid map. In the map there are 3 points: Trophy, Pitfall and Hero, only the play is controllable. The goal for this game is directing the Hero reach the Trophy, avoiding the Pitfall in the the map.

Figure 1 shows an example of of the map. In addition, the policy of the environment could be changed for different experiment.

V Experiments

We first designed the task we want the algorithm and the learning goal based on these task:

- **Normal Task** Description: In each episode the map is fixed; position of pitfall is fixed; the size of map is normal (8*8). Learning Goal: Algorithm will find a path to the goal properly

- **Extreme Task** Description: In each episode the map is fixed; position of pitfall is fixed; the size of map is in extreme value (3*8 and 16*16, extremely small and large respectively). Learning Goal: Algorithm will find a path to the goal properly, with fixed episodes.

- **Unstable Task** In each episode the map is fixed; pitfall will randomly move in each step; the size of map is normal (8*8). Learning Goal: Algorithm will find a path to the goal properly

- **Survivor Task** Description: In each episode the map is fixed; pitfall will move forward to the Hero with 20% probability in each step (but the pitfall will never hit the Hero unless the Hero make a decision stepping into it); the size of map is normal (8*8). Learning Goal: Algorithm will find a path to the goal properly, and learn to bypass the pitfall.

- **Random Task** Description: In each episode the map is randomly regenerated; the size of map is normal (8*8). Learning Goal: Algorithm will learn that Hero need to approach to Trophy and avoid the Pitfall, which a more general knowledge, we think this knowledge has higher abstract level.

VI Results

In this section, we will discuss the result of our experiment respectively. For *Random Task* we tried this task for $2,000,000$ episodes, while for other task we run two algorithm for only $200,000$ episodes in order to save the computer resource.

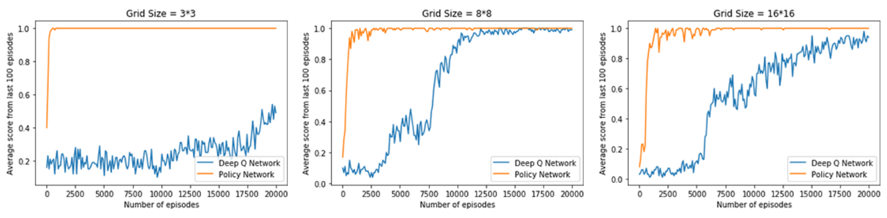

Figure 2. Normal & Extreme Task

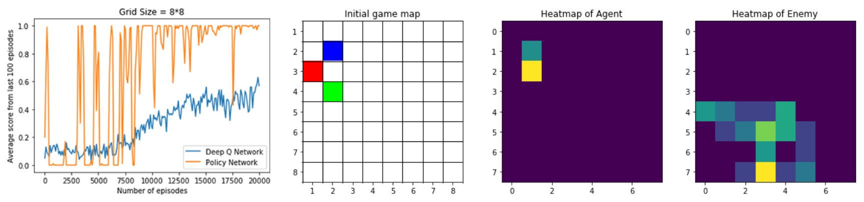

Figure 3. Unstable Task

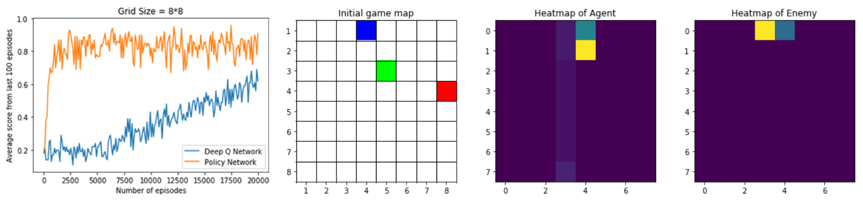

Figure 4. Survivor Task

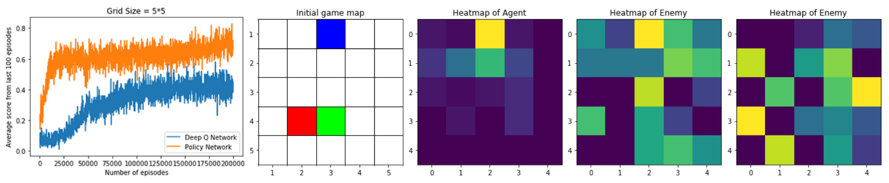

Figure 5. Random Task

- **Normal Task & Extreme Task** The normal case of the game is running with the size of $8 * 8$, the reason is we ran those algorithm on our computer. From the result of those three tests, we found following facts:
  - On average, the policy gradient will converge faster than Deep Q network.
  - In extremely small task, the Deep Q network will converge super slow because of it using random actions for *Exploreration*
  - In extremely large task, the policy gradient will converge much more faster, we think the reason is because the propagation speed of reward is faster in policy gradient.

- **Unstable Task**

In the unstable task, since the movement of the Pitfall is not predictable, when the pitfall is close to the agent (initial state), it affected a lot for the converging of both algorithms. But when it moved to further position, the policy gradient will immediately reach a good result. We think the policy gradient learned to ingore the influence from the random Pitfall.

- **Survivor Task**

  In the Survivor task, since the pitfall will be very close to the agent in the late episodes. It will affect agent a lot, but what we can see from the figures, we found the policy gradient will shake more than Deep Q network. We think the reason is the policy gradient has better global view than Deep Q network, but the history made a bad effect in this case.

- **Random Task**

  For the random task, since the problem space is much larger than a fixed map (We suppose that the computer need to learn nearly all the possible status before it can make a good prediction). We can see the policy gradient will still have better performance than Deep Q network. But from the figures, we can also derive that the dimension of our neural network might be too small to learning this knowledge. Because the rate on increasing slowed down in last few episodes.

VII Discussion

According to those researchers [3, 8], the result of the experiment indicated that policy gradient has faster convergence rate than DQN, especially in the complexity environment. In the research [8], it also mentions that policy gradient suffers the high variance in approximately the policy function; it causes the noisy during the estimation. However, we use the variance reduction method to reduce the noise. For this reason, in our experiment result, the policy gradient shows a stable performance from episode to episode.

VIII Conclusions

In this project, we have deeply understood the history of reinforcement learning. We learn how to implement the two well-known methods, q-learning and policy gradient. Furthermore, we apply those methods in the maze game, and then to training the game agent to reach the goal automatically. During this self-study journey, we also learn the evolutionary process of reinforcement learning, which is a powerful method of machine learning, and it has been utilized not only in the gaming industry but also the robot training. If we have more time, we would like to port our algorithm to different gaming, or even the difficult gaming, such as Pac-man or Mario.

To provide further evaluation of the experiment result of q-learning and policy gradient, the hyper-parameter tuning is extremely important to the accuracy, the average reward and the convergence speed. As same as the other deep learning algorithm, although the method can escape the local optimal, the same value of the parameters can cause the time consumption in each episode.

References

[1] G A. Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. 11 1994.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[3] John Schulman, Pieter Abbeel, and Xi Chen. Equivalence between policy gradients and soft q-learning. *CoRR*, abs/1704.06440, 2017.

[4] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[5] Gerald Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8(3):257–277, May 1992.

[6] Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, March 1995.

[7] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[8] Felix Yu. Deep q network vs policy gradients - an experiment on vizdoom with keras. 11 2017.

IX Roles of the Authors

We work individually on the reinforcement learning research and algorithm implement, after the research and implement, we share with each other our idea and what we have learned from the reinforcement learning. Therefore, we also help each other to figure out the confusion of the methods and algorithm. in the reporting part, we divide the work as below list.

- Dongmin Wu: Method, Data, Experiments, Result
- Shan Kuan: Abstract, introduction, Related work, Discussion, Conclusion, Reference